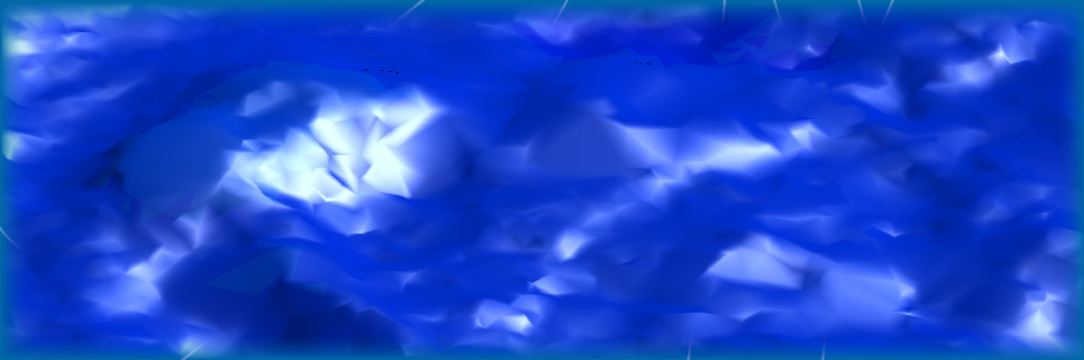
Paavo Nevalainen

Geometric Data Understanding

Deriving Case-Specific Features



Turku Centre *for* Computer Science

TUCS Dissertations
No 233, August 2018

Geometric Data Understanding: Deriving Case Specific Features

Paavo Nevalainen

To be presented, with the permission of the Faculty of Science and Engineering of the University of Turku, for public criticism in Auditorium Agora XXI on Aug 20, 2018, at 12 noon.

University of Turku
Department of Future Technologies
Vesilinnantie 5, 20500 TURKU, FINLAND

2018

Supervisors

Professor Jukka Heikkonen
Department of Future Technologies
University of Turku
Finland

Assistant Professor Tapio Pahikkala
Department of Future Technologies
University of Turku
Finland

Adjunct Professor Mikko-Jussi Laakso
Department of Future Technologies
University of Turku
Finland

Reviewers

Paola Magillo
DIBRIS - Dipartimento di Informatica, Bioingegneria, Robotica e ingegneria dei Sistemi
Università degli studi di Genova
Via Balbi 5, 16126 Genova
Italy

Csaba Beleznai
Department of Vision, Automation & Control
AIT Austrian Institute of Technology
Giefinggasse 4, 1210 Vienna
Austria

Opponent

Timo Tokola
Faculty of Science and Forestry
University of Eastern Finland
PL 111, FI-80100 Joensuu
Finland

ISBN 978-952-12-3723-2
ISSN 1239-1883

To all sports researchers and GIS workers of the world!

Abstract

There exists a tradition using precise geometric modeling, where uncertainties in data can be considered noise. Another tradition relies on statistical nature of vast quantity of data, where geometric regularity is intrinsic to data and statistical models usually grasp this level only indirectly. This work focuses on point cloud data of natural resources and the silhouette recognition from video input as two real world examples of problems having geometric content which is intangible at the raw data presentation.

This content could be discovered and modeled to some degree by such machine learning (ML) approaches like deep learning, but either a direct coverage of geometry in samples or addition of special geometry invariant layer is necessary. Geometric content is central when there is a need for direct observations of spatial variables, or one needs to gain a mapping to a geometrically consistent data representation, where e.g. outliers or noise can be easily discerned.

In this thesis we consider transformation of original input data to a geometric feature space in two example problems. The first example is curvature of surfaces, which has met renewed interest since the introduction of ubiquitous point cloud data and the maturation of the discrete differential geometry. Curvature spectra can characterize a spatial sample rather well, and provide useful features for ML purposes. The second example involves projective methods used to video stereo-signal analysis in swimming analytics.

The aim is to find meaningful local geometric representations for feature generation, which also facilitate additional analysis based on geometric understanding of the model. The features are associated directly to some geometric quantity, and this makes it easier to express the geometric constraints in a natural way, as shown in the thesis. Also, the visualization and further feature generation is much easier. Third, the approach provides sound baseline methods to more traditional ML approaches, e.g. neural network methods. Fourth, most of the ML methods can utilize the geometric features presented in this work as additional features.

Keywords: triangularized irregular networks, curvature, point clouds, microtopography, silhouette capture, camera calibration.

Tiivistelmä

Geometriassa käytetään perinteisesti tarkkoja malleja, jolloin datassa esiintyvät epätarkkuudet edustavat melua. Toisessa perinteessä nojataan suuren datamäärän tilastolliseen luonteeseen, jolloin geometrinen säännönmukaisuus on datan sisäsyntyinen ominaisuus, joka hahmotetaan tilastollisilla malleilla ainoastaan epäsuorasti. Tämä työ keskittyy kahteen esimerkkiin: luonnonvaroja kuvaaviin pistepilviin ja videohahmontunnistukseen. Nämä ovat todellisia ongelmia, joissa geometrinen sisältö on tavoittamattomissa raakadatan tasolla.

Tämä sisältö voitaisiin jossain määrin löytää ja mallintaa koneoppimisen keinoin, esim. syväoppimisen avulla, mutta joko geometria pitää kattaa suoraan näytteistämällä tai tarvitaan neuronien lisäkerros geometrisia invariansseja varten. Geometrinen sisältö on keskeinen, kun tarvitaan suoraa avaruudellisten suureiden havainnointia, tai kun tarvitaan kuvaus geometrisesti yhtenäiseen dataesitykseen, jossa poikkeavat näytteet tai melu voidaan helposti erottaa.

Tässä työssä tarkastellaan datan muuntamista geometriseen piirreavaruuteen kahden esimerkkiohjelman suhteen. Ensimmäinen esimerkki on pintakaarevuus, joka on uudelleen virinneen kiinnostuksen kohde kaikkialle saatavissa olevan datan ja diskreetin geometrian kypsymisen takia. Kaarevuusspektrit voivat luonnehtia avaruudellista kohdetta melko hyvin ja tarjota koneoppimisessa hyödyllisiä piirteitä. Toinen esimerkki koskee projektiivisia menetelmiä käytettäessä stereovideosignaalia uinnin analytiikkaan.

Tavoite on löytää merkityksellisiä paikallisen geometrian esityksiä, jotka samalla mahdollistavat muun geometrian ymmärrykseen perustuvan analyysin. Piirteet liittyvät suoraan johonkin geometriseen suureeseen, ja tämä helpottaa luonnollisella tavalla geometristen rajoitteiden käsittelyä, kuten väitöstyössä osoitetaan. Myös visualisointi ja lisäpiirteiden luonti muuttuu helpommaksi. Kolmanneksi, lähestymistapa suo selkeän vertailumenetelmän perinteisemmille koneoppimisen lähestymistavoille, esim. hermoverkkomenetelmille. Neljänneksi, useimmat koneoppimismenetelmät voivat hyödyntää tässä työssä esitettyjä geometrisia piirteitä lisäämällä ne muiden piirteiden joukkoon.

Avainsanat: kolmioverkkomallit, kaarevuus, pistepilvet, mikrotopografia, hahmoääriviivojen kaappaus, kamerakalibrointi.

Acknowledgements

I would like to express my special appreciation and thanks to my advisors Prof. Jukka Heikkonen and Assistant Prof. Tapio Pahikkala, you have given both long leash, some whip and monetary support in suitable proportions. Special thanks to reviewers Prof. Paola Magillo and Dr. Csaba Seleznyi for encouraging my research and for pointing out the path of the science adherent.

There are several esteemed researchers, who served as good role models. These are: Doctors Maarit Middleton (GTK), Pekka Naula (UTU), Samuli Launiainen (Luke), Aura Salmivaara (Luke) and MSc Juval Cohen (IL). The national olympic swimming team coach, MSc Antti Kauhanen demonstrated by his very being how the well-maintained physical condition helps cognition and staying energetic.

I would also like to thank secretaries and support persons like Sami Nuutila and Pia Lehmussaari in my laboratory. Their determination, ability to stay focused and tolerance to my often repetitive questions is one of the leading forces to awaken a desire in me to become a little better person. All of you have been there to support me when I stumbled around with the daily practicalities.

A special thanks to colleagues in the room 452B. MSc's Ileana Montoya-Perez, Jussi Toivonen, Parisa Movahedi, Markus Viljanen and Jonne Pohjankukka have shown a lot of tolerance to my daily trumpeting about my projects and thesis. DSC (Tech) Anne-Maarit Majanoja proved that achieving a doctoral degree does not necessarily prevent perceiving the reality as it is. I would also like to thank all of my friends who supported me during these years and incited me both not to strive towards my goal (for two decades) and to strive towards it on this decade. At the end I would like to express appreciation to my beloved wife Piritta, who spent rare free-time moments with me and was always there to remind me about how all things will come to an end sooner or later.

Turku, May 2018
Paavo Nevalainen

List of original publications

Papers are cited as P1,R2,P3,R4,P5 and P6.

- P1 Paavo Nevalainen, Maarit Middleton, Ilkka Kaate, Tapio Pahikkala, Raimo Sutinen, Jukka Heikkonen, Detecting stony areas based on ground surface curvature distribution. In: Rachid Jennane (Ed.), I*PTA 2015 The 5th International Conference on Image Processing Theory, Tools and Applications*, 581–587, IPTA, 2015.
- R2 Paavo Nevalainen, Antti Kauhanen,Csaba Raduly-Baka, Mikko-Jussi Laakso, Jukka Heikkonen, Video based Swimming Analysis for Fast Feedback. In: Maria De Marsico, Gabriella Sanniti di Baja, Ana L. N. Fred (Eds.), *Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2016*, 457–466, ScitePress, 2016.
- P3 Paavo Nevalainen, Maarit Middleton, Raimo Sutinen, Jukka Heikkonen, and Tapio Pahikkala, Detecting terrain stoniness from airborne laser scanning data. *Remote Sensing*, 8, 720–739, 2016.
- R4 Paavo Nevalainen, Muhammad Hashem Haghbayan, Antti Kauhanen, Jonne Pohjankukka, Mikko-Jussi Laakso, Jukka Heikkonen, Real-Time Swimmer Tracking on Sparse Camera Array. In Ana L. N. Fred, Maria De Marsico, Gabriella Sanniti di Baja, *Pattern Recognition Applications and Methods ICPRAM 2016, Revised Selected Papers. Lecture Notes in Computer Science, 10163*, 156–174, 2017.
- P5 Paavo Nevalainen, Ivan Jambor, Jonne Pohjankukka, Jukka Heikkonen, Tapio Pahikkala, Triangular Curvature Approximation of Surfaces - Filtering the Spurious Mode. In*: Maria De Marsico, Gabriella Sanniti di Baja, Ana L. N. Fred (Eds.), Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2017*, 457–466, ScitePress, 2017.
- P6 Paavo Nevalainen, Aura Salmivaara, Jari Ala-Ilomäki, Samuli Launiainen, Juuso Hiedanpää, Leena Finér, Tapio Pahikkala, Jukka Heikkonen. Estimating the rut depth by UAV photogrammetry. *Remote Sensing.* 2017, 9(12), 1279.

Contents

List of abbreviations

(P1),(R2),(P3),(R4),(P5),(P6)	publications presented in Ch. 6.2.
ALS	aerial laser scan
CAD	computer aided design (or 'drafting' at earlier times)
CCV	complete cross-validation
CV	cross-validation
DDG	discrete differential geometry
DEM	digital elevation model
DG	differential geometry (of continuous surfaces)
DSA	differential of the slope angle
GA	Geometric algebra
GIS	geographic information system
HOC	histogram of oriented curvatures
HOG	histogram of oriented gradients
k-CV	k-fold cross-validation
L-B	Laplace-Beltrami operator
MCF	mean curvature flow, a well-known TIN smoothing method
MDL	minimum description length

ML	machine learning
NN	natural neighbors in a Delaunay triangulation
NSF	National Survey of Finland
PCL	point cloud library, a popular c++ library for processing point clouds
SAF	spatial angle filtering
SCV	spatial cross-validation
TIN	triangularized irregular network
UAV	unmanned aerial vehicle

Nomenclature focusing mainly to point cloud analysis

$(a, b, c) = t \in T$	three cloud points associated as one triangle t
$\mathrm{acos}(.,.)$	The angle defined by a vector pair
α	angle of the directional curvature
$\bar{\phi}_{te}$	tip angle fraction associated to an edge $e \in t$ of a triangle t.
β_e	the edge angle of an edge e
$\beta_{tt'}$	edge tilt angle fraction
χ	Euler number of a topological entity (e.g. a polytope)
δ	the disk width for local height transform.
$\epsilon, \mathcal{O}(\epsilon), t(\epsilon)$	the perturbation factor of a polytope volume, a perturbed polytope and a triangle
γ	optical radius
$\gamma_{min}, \gamma_{max}$	view cone radius limits, see $\omega_{min}, \omega_{max}$
κ	average curvature on a 2D

κ_1, κ_2	principal curvatures $\kappa_1 \leq \kappa_2$
ψ_i	l'Huillier angle coefficients
$\mathbb{E}_S(.)$	expected value of (.) within a set S
$\mathbf{H}(\alpha)$	directional curvatures of triangles T at a direction α
$\mathbf{h}(p)$	A harmonic part of a general vector field
$\mathbf{N}$	a matrix of all vertex normals in a sample
$\mathbf{N}_p$	$= (n_{t_1}\, n_{t_2}\, n_{t_3})$, a transform matrix to ensure planar expansion in case of a vertex $p \in t_1 \cap t_2 \cap t_3$ common to triangles t_1, t_2, t_3
$\mathbf{W}$	the assembly matrix to compute directional curvatures $\mathbf{H}(\alpha)$ of a sample
$\mathcal{O}$	a generic geometric object
meas(S)	A Lebesque measure of S, either length, area or volume depending on the set S.
$\mathcal{O} \subset \mathbb{R}^3$	A (solid) spatial object
$\omega(v_1, v_2, v_3)$	solid angle defined by a vector blade v_1, v_2, v_3
$\omega, \omega_p, \omega_t$	solid angle,solid angle associated to a vertex p and a triangle t
ω_{2D}	cone vertex angle used with a 2D analogue of TIN
$\omega_{min}, \omega_{max}$	spatial angle limits of SAF. The lower limit describes the sharpest acceptable 'pikes', the upper limit is for the most penetrating acceptable 'holes'
ω_{tp}	contribution of a spatial compartment of a triangle $t \in T(p)$ and the vector $-e_3$ to a solid angle ω_p
$\partial T \subset P$	border points. ∂T equals the convex hull of P when no elimination of the large boundary triangles have been done.
$\partial \mathcal{O}$	the surface of an object
ϕ	central half-angle of triangle edges of a regular 2D polygon

$\Phi(p) = z(p)$ a potential function at a point p

$\phi_p, \phi_{tp}, \phi_{pt}$ sum of projected tip angles, tip angle of a triangle t at a vertex p, projected tip angle of t at p.

ϕ_{tp} a tip angle of triangle t at the vertex p

$\tilde{A}_t(\epsilon)$ area of a triangle t from the vertex expansion with perturbation ϵ

$\tilde{G}_t$ Gaussian curvature of a triangle t using the vertex expansion principle

$\tilde{H}_t$ mean curvature of a triangle t using the vertex expansion principle

$\underline{n}(\underline{p})$ direction of the local slope

$\underline{p} \in \mathbb{R}^2$ a cartographic location in 2D

ω_{2D} a 2D analogy of the solid angle of a vertex

$\vec{e} = q - p$ an edge vector, when the edge is $e = (p, q)$

A, A_t area, of a triangle t

A_p TIN surface area associated to a vertex p. Several policies exist in choosing the actual value.

A_{pt} ingredient from a triangle t to the vertex area A_p.

$E(p) \subset E \subset P^2$ a set of edges (edge star) joined to a a vertex p, set of all edges of a TIN

$e = t \cap t' = (p, q) \in E$ an edge of a TIN. e is between triangles $t, t' \in T$, between edge end points p, q

$e_{tp} \in E$ an edge (a, b) of a triangle $t = (p, a, b)$ opposite to a point p

f, f_p, f_t any feature definable by the local point cloud analysis, at vertices, on triangles.

G, G_t, G_p Gaussian curvature, at a triangle t, at a vertex p

H, H_t, H_p, H_e mean curvature, at a triangle t, at a vertex p, at an edge e

$k(\underline{p})$ geographic slope at a point p

L	nominal length of a polytope
$l_e, l_{pq}, l_{tp}, l_{t\cap t'}$	a length of an edge
$N(p) \subset P$	natural neighbors of a vertex point p
N_p	a matrix for scaling vertex normals. Used in a motivational numerical experiment.
$n_p, n(q)$	a unit normal vector at a vertex point p and at a general point q.
n_t	a triangle unit normal
$n_{pt}(\alpha)$	directional vertex normals for the surface oriented curvature
$P(p) \subset P$	natural neighbors of a vertex point p
$p \in P$	a cloud point
$P \subset \mathbb{R}^3$	a point cloud achieved by LiDAR, photogrammetry etc. methods
$P\_(v_1, v_2)$	projection to the subspace spanned by v_1, v_2
$P_\perp(v)$	projection to the subspace perpendicular to v
$p_{te} \in P$	a point of a triangle t opposite to an edge e
$q \in t$	a cloud point or a point within a triangle t
r	radius
$ring_r(\underline{p}, \delta)$	a horizontal ring with radii $r - \delta/2$ and $r + \delta/2$ centered at $\underline{p}$.
$sgn(e)$	signum of an edge e (positive on 'ridges' and negative on 'ravines')
$T(t), T(p) \subset T$	adjoined triangles of a triangle t and of a vertex point p
$T = Delaunay(\underline{P})$	horizontal Delaunay triangulation of a point set $P \in \mathbb{R}^3$.
$t \in T$	a triangle
$T \subset P^3$	set of all triangles
$V, V(\mathcal{M}), V_t$	volume, of a polytope $\mathcal{M}$, of a space discretization associated to a triangle t

$v, v_1, v_2, v_3 \in \mathbb{R}^3$ generic vectors for occasional usage

$Voronoi_P(p) \subset \mathbb{R}^3$ the Voronoi cell around a point p

$w_1(u), w_2(R)$ the fitting weight, radial weight of the local height formula

$z = (p, x, y) \in D$ spatial data sample: p is a geo-location, x are features and y is the label

$z = (x, y) \in D$ data sample: x are features and y is the label

Special symbols

$.^0$	unit vector operator
$\mathbf{1}$	$= (1, 1, 1)^T$
$\nabla .$	gradient operator
$\partial .$	boundary of a discrete or continuous set.
$\underline{.}$	horizontal projection of a 3D entity (point or vector)
$\{.\}_.$	an indexed set, also a matrix enumerated by its elements
$\{\}$	empty set
$\_$	missing value, especially with geographic samples, e.g. $z = (x\ \_)$

Chapter 1

Introduction

Machine learning (ML) is a discipline residing in the intersection of three disciplines: computer science, mathematics and domain or target field (Conway, 2010). Geometry is a rather commonly occurring mathematical field, which is sometimes used abusively on the domain specification, and maybe sometimes left underrepresented in the intersection of ML and mathematics. There is a question: how much one should understand about the domain field? Perhaps good results follow already by a routine application of the most standard methods? Of course, certain amount of data understanding is always needed e.g. for imputation of the missing data, choosing correct scaling etc. Sometimes elaborate techniques are being developed to produce helpful features for specific problems. Development of such a new feature can be a tedious task. One of the main arguments of this Thesis is that the geometric data understanding is helpful in the completion of the ML projects.

Data understanding requires sometimes a lengthy analysis of the data sources, measurement methods and data format alternatives. The cross-industry standard process (CRISP-DM) for data mining (Berthold et al., 2010), has six stages in a ML project:
1) project understanding, 2) data understanding, 3) data preparation, 4) modeling, 5) evaluation, 6) deployment. In reality, first three stages have diffuse boundaries, and evaluation is a crucial step. One could say that process understanding is about defining the evaluation criteria and data understanding is about what to model. This work focuses on stages 2) and 3) from the geometric point of view using the ground curvature analysis and geometric swimming video rectification as examples.

Geometry is an aspect of mathematics which pervades a lot of ML theory starting from the concrete application fields of this work (micro-topographic registration from point clouds and underwater stereo-imaging) to such abstract heights as kernel spaces, Lie algebraic image processing (Kondor, 2008) and manifold projections (Sober and Levin, 2016). Usually there is a don't touch - don't get hurt approach to geometry. The time it takes to get to the end of the matter could have been used in learning strong parameterless or self-adaptive methods perhaps leading to similar results.

The actual research of this work is focused on three cases: 1) registration of an area stoniness without specifying individual stones with publications (P1) and (P3), 2) micro-topographic registration of forest machine trails (P5,P6) and 3) the underwater camera view rectification (R2,R4). Two main cases are depicted in the upper row of the Fig. 1.1. The thesis concerns geometrically justifiable representations, which can be utilized by some ML methods, see Fig. 1.1. The related publications (P1,R2,...,P6) can be found in Ch. 6.2.

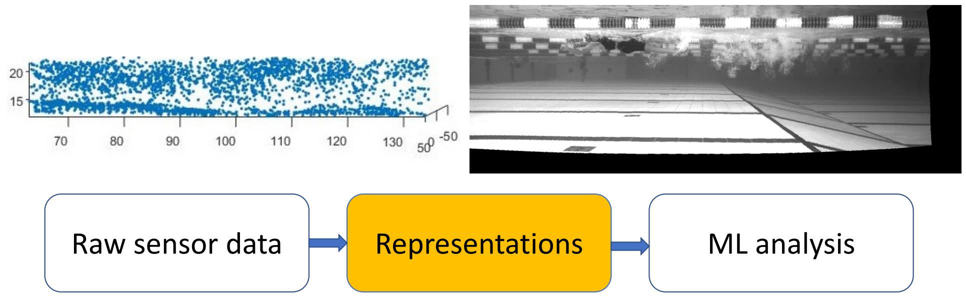


Figure 1.1: The main input domains (data types) covered in this work. *Above left*: A slice of a 3D point cloud at Turku Harakkakallio. Measures are in meters and data from NFL open data site. *Above right*: An athlete swimming at 25 m pool of the Impivaara public swimming center. The image has 1022 × 355 pixels, each pixel is 4 mm wide in the geometric projection. *Below*: The process flow, where raw data is turned to representations for analysis by ML methods. The Thesis scope is in yellow.

The point clouds are created by aerial and ground-based light detection and ranging (LiDAR) and photogrammetric measurements. Point clouds were originally used to measure and model built environments (indoors, built infrastructure), but the natural resource modeling and evaluation has become common during last two decades. This development shows in the available point cloud analysis methods, too. There is a wide variety of methods e.g. for curvature analysis (see Ch. 3). They suite well to densely sampled, smooth surfaces with the point cloud position noise level close to the natural coarseness of the surface. Only few of the available methods suit to the natural resource analysis, where samples are sparse and noise level is high, and surfaces have no ideal smoothness.

The best swimming analysis sites have a dense array of cameras with a stereo-camera calibration. Cameras are either temporarily installed or permanent, and even the physical aspect of these systems requires high accuracy and is expensive. In case of P2 and P4, the number of cameras was limited by the available computational capacity and camera places by the architecture.

Existing situation on these two fields (natural resource analysis and swimming analysis) seems to leave room for some more research. Point clouds on natural resources exhibit a wide range of spatial distributions, and e.g. the terrain surface still has no formal multi-scale definition. There was also need for a computationally efficient swimming analysis arrangement.

1.1 Motivation to geometrically expressive features

An approach among the ML practitioners is to add features to the data model whenever new data sources or new feature extraction methods become available. This is a justifiable practice as long as the total amount of features stays low (e.g. under 200), since most of the methods are noise-tolerant and produce about the same results with or without useless features. The meaning or even the source of a feature is of less importance. This encourages considering the method as a black box, and data as a primal input only.

Geometric properties have always been a part of the image processing techniques. There are also various ways to produce invariance to Euclidean (rigid) transformations (translation, rotation). There is a recent general purpose phi-descriptor (Matsakis, 2016), which may prove to be useful e.g. in geomorphology classifications. Geometry seems to pervade the deep learning, too. The recent capsule networks (Sabour et al., 2017) implements a voting system, where a higher order feature will be accepted only if it is geometrically sensible with respect to the lower order features.

Three motivational points for having geometric representations either as features and feature sets or as a separate date structure, are outlined in the following.

1. Some geometric entities are interesting or useful by themselves, even the concrete measurement of them may be cumbersome or impossible. There is a need for **an indirect model-based geometric measurement**.

2. Geometry often provides additional constraints reducing the degrees of freedom of the problem, or at least giving better probabilistic structure to the noise component. This argument is loosely related to the minimum description length principle (MDL), which states that the best models are such which enable the most efficient transmission of the data (Rissanen, 1978). Using dimensions of the geometric primitives involved in data may lead to a separation of the noise and the characteristic structure. This case is of **efficient geometric data modeling**.

3. Sometimes there is no easy way to give a probabilistic model to a phenomenon (e.g. the cross-terrain ground model). The model varies over different scales, and to capture the properties of the surface one needs to register the model on a spectrum of physical scales, and use the acquired continuum of models as a basis for further ML operations. This motivational item is termed as the **multi-scale geometry aspect**.

A need for indirect measurements based on a geometric representation, (see Fig. 1.1) is relatively easy to perceive in cases, where it occurs. An example of this need is swimming analysis (P2), where a generic metric view to the body dynamics is essential. The second motivational factor (structuring data geometrically) is exemplified e.g. in the micro-topography analysis (study of the ground contour in small scale) where e.g. the derived features should be independent of the slope (P1).

The swimming research is dominated by the need for videometric measurements e.g. the speed of the swimmer as a function of time. In this respect, it was considered essential to get a geometrically rectified projection of the swimming action before any silhouette capture or vectorization etc.

1.2 Research goal and methods

The research goal was to find helpful geometric representations and methods for the micro-topography and swimming research, which would be an improvement from the currently known methods. The point cloud study had a focus on the curvature aspect, which may prove useful with or without the neural network methods, which are gaining popularity. The aim of the swimming research was to develop an applicable and verifiable video metrology approach to serve as a basis and a benchmark for future ML attempts in the biometrics analysis.

The results are summarized in Ch. 7 on p. 89.

Research data consists of aerial point clouds from Sodankylä and photogrammetric point clouds from Vihti, Finland. The swimming video records are from Turku Impivaara Sport Center. The point cloud analysis is focused only on curvature aspects and the video material was subjected to camera calibration.

1.3 Proposed novelties

The following is a list of the proposed novelties in papers P1,...,P6.

1. A point cloud thinning algorithm (spatial angle filtering (SAF)), which applies to ground and canopy modeling, and can be tuned in a ML training phase to specific tasks. (P6)

2. An efficient equation to estimate mean curvature of a triangle. (P5)
3. A numerically efficient histogram of gradients (HOC) method based on the properties of the above equation. (P6)
4. A practical and economical procedure for the projective geometry capture using a sparse underwater camera array. (R2,R4)

1.4 Organization of the thesis

This thesis consists of two parts that both together form the contribution of the research. The thesis has 7 chapters. The second part presents the six original research publications that are conducted in various research projects. Gaining the data understanding as an early part of a machine learning project and geometric data have been discussed in this Chapter. Chapter 2 gives an introduction to supervised machine learning and its relation to primal and derived features.

Chapter 3 outlines concepts and theory of triangulated irregular network (TIN) curvature and briefly presents some characteristics of topography and micro-topography such as local height, slope and curvature. Chapter 3 also provides the state-of-the-art approach to curvature in the extent needed to relate current research and the new results of the thesis. The Chapter 4 contains three algorithms or methods, which are among the deliverables of the research done. The Chapter 4 also has a summary of the computational complexity of TIN curvature methods. Some numerical examples are also provided.

Chapter 5 provides an overview to video-based athletics performance analysis using swimming as the target sport. The camera calibration problem and issues with inaccuracies in camera positioning are covered briefly.

A summary of the research papers of the second part is provided in Chapter 6. Each problem domain is briefly introduced, the approach and outcome are presented in the context of the research goals of this thesis. Chapter 7 summarizes the contributions and limitations of this thesis. Also, possible future extensions to the research are discussed.

Chapter 2

Machine learning

Several methods presented in this thesis have method parameters which have been or can be tuned and utilized by some Machine Learning (ML) approaches. As Samuel (1959) states, ML brings some sort of autonomy into the decisions ML systems make. This autonomy emerges from specific algorithms, which may meet wider conditions than the programmer specifically enforces. One way to measure this autonomy (or resiliency to details of new cases) is called the generalization capability. It means how well an algorithm performs when facing new but similar data.

ML learning algorithms can be divided to supervised and unsupervised ones. The former have a specific data set used to tune a set of algorithm parameters and to estimate the generalization performance. Data has to be divided or augmented to features and a target variable (labels). Unsupervised methods form the necessary classification from the features only and may provide insight to data in general. Semi-supervised algorithms need a source of extra input (reinforcement learning) or a user feedback to improve the performance (recommender systems).

Fig. 2.1 outlines the thesis scope in terms of general ML concepts, which are briefly introduced in the following text. A vivid spectrum of algorithms and methods is forming from supervised to unsupervised methods. This work employs only the **supervised learning**, which requires pre-defined expert input or valid measurements in order to be applicable. This is because of the type of the project cases were suitable for supervised learning. The sample sizes in this work are small and in that case the **cross-validation** (CV) is a natural tool to measure the classification or prediction performance when facing the new data (generalization performance). CV is able to find an approximate optimum between the state of underfitting and overfitting. The former occurs when the algorithm performs poorly with all data, and the latter occurs when the algorithm performs poorly with the new data. There are also information criteria assessing the model complexity choice

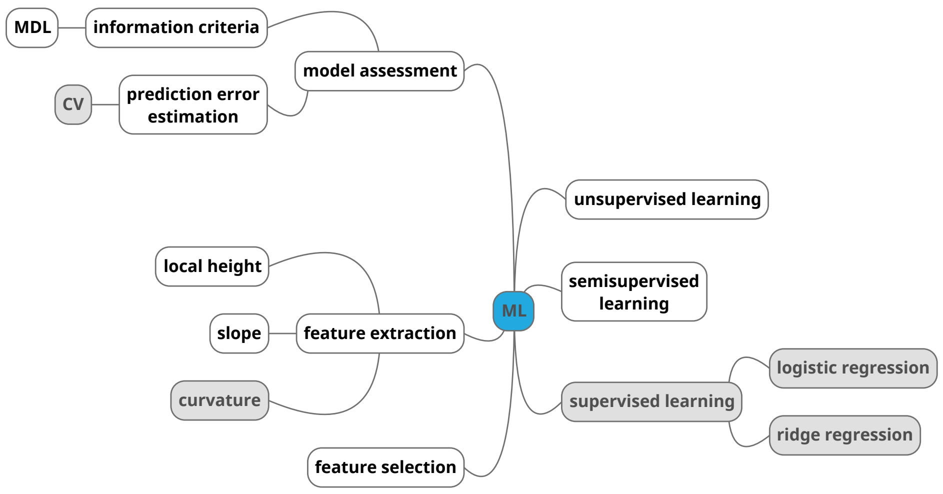


Figure 2.1: ML concepts central to the thesis (in grey). The mind map is adapted from Amazon.com (2018).

e.g. Akaike (AIC) and Bayesian information criteria (BIC) (**?**) and the minimum description length approach (MDL) (Rissanen, 1978). To work well, AIC requires the true model to be infinite dimensional and BIC requires the model to be finite dimensional (**?**). The point cloud data sets are heavily censored in this work and no model families with controllable complexity were used. Also, the nature of the true model has not been taken into consideration and the usage of AIC and BIC have been omitted. All the above criteria require the samples to be i.i.d, which is probably not the case with the point clouds.

The **logistic regression** fits a hyperplane which is most likely separator between two classes, this in the sense of a likelihood function totally definable by data only. The optimization problem involved in logistic regression can also be understood purely from a geometric perspective. There exists a sigmoidal weight function for the linear regression, which leads to the exactly same solution. The **ridge regression** is an ordinary linear regression but with a penalty subjected to the linear regression coefficients.

Sec. 2.1 introduces the supervised learning from a type theoretical and software modularization point of view, Sec. 2.2 brings up the specifics when applying supervised learning to the spatial prediction and Sec. 2.3 brings very briefly up the feature extraction needed to deal with various geometrically inspired features, some of which will be presented in Ch. 3 and Ch. 4.

2.1 Supervised learning

A data sample $z = (x\,y)$ consists of a feature vector $x \in \mathcal{X} \subset \mathbb{R}^d$ and a label $y \in \mathcal{Y} \subset \mathbb{R}$, where $d \in \mathbb{N}$ is the data dimension and $\mathcal{X}$ and $\mathcal{Y}$ are the domains of the feature vectors and labels, respectively. The label y is often called the tag, class, label or target variable or class of a sample z. A set of samples z can be considered as well a matrix $D = (X\,Y)$ with feature block X and the label column Y, if we just associate each row z of D to the sample membership relation: $z \in D$. To shorten the presentation the set aspect and the matrix aspect of D have been denoted as the same, even this is not exactly so. The function application to sets of samples is being used. Inevitable variations like multi-target prediction $D = (X\,Y_1\,Y_2)$ are excluded from the scope of the presentation, since the cases covered are single-target $D = (X\,Y)$ ones.

The relation between labels Y and features X can be characterized by an approximate function $\hat{f}$ such that $D \approx (X\;\hat{f}(X))$. The function $\hat{f}$ is called a predictor. The problem of finding an ideal function $f : X \to Y$ is usually ill-posed meaning that there are: a) some missing values $(x\;\_)$, b) duplicate labels $(x\;y_1) \neq (x\;y_2)$, or c) sensitivity of resulting labeling $\hat{Y}$ over a small perturbation ϵ of input $(x\;y) \mapsto (x + \epsilon e_i\;y)$, where e_i is the i'th Euclidean base vector. A large scale example of a binary classification with two partially overlapping clusters can be considered as approximately of type b) with strong regularization, and of type c) with weak regularization.

The ill-posedness can be fought by regularization or by CV or a combination of both. CV actually 'regularizes' models with e.g. a polynomial base with a variable degree. The resemblance of supervised learning techniques and regularization of ill-posed inverse problems is widely noted (Sever, 2015).

The maximum likelihood principle (Hastie et al., 2001) prefers a predictor $\hat{y} = \hat{f}^*(x)$ that its value is the most probable given the data D i.e. $\hat{f}^*(x) = \mathrm{argmax}_{y \in \mathbb{R}}(p(y|x_+, D))$, where $\hat{y}$ is the predicted value, $p(y|x_+, D)$ is the conditional probability for y given a new feature vector x_+ and a data set D. Several geometrically or intuitively justified methods have been developed to approximate $\hat{f}^*$ in cases where the direct or approximative addressing of the probability $p(y|x, D)$ is not practical. The varying mixture of intuition, geometric reasoning and concepts of probability are seen in methods like k-nearest neighbors, k-means and logistic regression, and procedures like nested cross-validation, which can be seen also as a re-sampling method.

2.1.1 Nested complete cross-validation

The general idea of nested CV is to produce both a predictor and an unbiased estimate of its prediction performance (Varma and Simon, 2006). The CV is based on some sort of search over subsets of data D, and the complete cross-validation (CCV) (Mullin and Sukthankar, 2000) has the most exhaustive search possible. The rest of the CV methods can be formulated with various restrictions on the enumeration of the data subsets of the CCV. The CV process itself is a minimization problem, as depicted in Fig. 2.2. An ML algorithm $\mathcal{A}$ bridges the processing of the data D to a series of optimization problems, where the performance (loss) of various subproblems is being estimated.

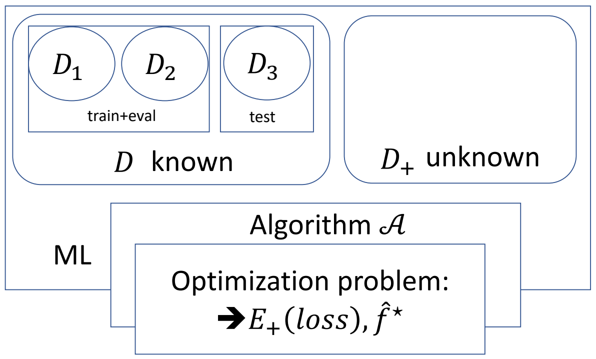


Figure 2.2: The nested CV as a performance optimization problem of a ML algorithm. The sample distinct subsets $D_1, D_2, D_3 \subset D$ act as a hierarchical iteration structure. $E_+(loss)$ is the estimate of the expected loss with a new data D_+, and $\hat{f}^\star$ is the deliverable predictor.

The treatise has a procedural and structural focus and the constituents of the CCV algorithm (**hyperparameters**, **training**, **prediction**, **internal parameters**, **enumeration**, **loss measure**, **evaluation**, **testing** and **deliverable predictor**) will be presented in a functional manner.

All supervised learning algorithms $\mathcal{A}$ produce a prediction function $\hat{f}$ specific to the application of $\mathcal{A}$. The functional type of the supervised learning algorithm is:

$$\mathcal{A}: \ \Theta \times 2^D \to \mathcal{Y}^\mathcal{X},$$

where $\Theta \subset \mathbb{R}^{|\theta|}$ is the domain of the hyperparameters $\theta \in \Theta$ of the algorithm $\mathcal{A}$. The power set 2^D is the set of all subsets of data D and $\mathcal{Y}^\mathcal{X}$ is a set of all surjective functions from feature vector values $\mathcal{X}$ to label values $\mathcal{Y}$. A specific single event of learning by an algorithm $\mathcal{A}$ produces a prediction function $\hat{f}: \ \mathcal{X} \to \mathcal{Y}$ from chosen **hyperparameters** θ and the data $D_0 \subseteq D$:

$$\mathcal{A}(\theta, D_0) = \hat{f} \tag{2.1}$$

with $\hat{f}$ possessing a usual potential for function application, i.e. $\hat{f}(x) = \hat{y}$. A clarification of the three roles of $\hat{f}$ in nested CV is in place. $\hat{f}$ is an individual iterate in the training phase, $\hat{f}^*$ is the best possible predictor to be tested, and $\hat{f}^\star$ is the delivered predictor.

Applying an algorithm $\mathcal{A}$ is called **training**, whereas applying $\hat{f}$ is called **prediction**. Note that training may involve some **internal parameters**. E.g. a ridge regression has a single hyperparameter $\theta = \lambda$ and internal parameters $\beta \in \mathbb{R}^d$ (in the case of zero mean normalized data), so that: $\hat{f} = x \cdot \beta$, where $\beta = \text{argmin}_{\lambda \in \mathbb{R}^+} \|\hat{Y} - Y\|^2 + \lambda \|\beta\|^2$. If the hyperparameters are missing, there is no need for CV, and the problem reduces to an ordinary (possibly still hard) minimization problem.

The CCV algorithm is an exhaustive in the sense that it uses a set of all possible tuples of three subsets *triplets* of data D. The definition of the search space utilizes a set split operator *split*, which is needed later and which produces all the pairs of the distinct subsets of D:

$$split(D) = \{(D_1, D_2) | D_1, D_2 \subset D,\ D_1 \cap D_2 = \{\}\} \tag{2.2}$$

The **enumeration** part consists of producing a set of all triplets of all possible distinct subsets of D: $triplets = \{(D_1, D_2, D_3) |\, D_3 \subset D, (D_1, D_2) \in split(D \backslash D_3\}$. The subsets D_1, D_2 and D_3 are named as the training, evaluation and test set, respectively. One can now create a model $\hat{f}$ by D_1, evaluate the choice of hyperparameters θ by D_2 and test the generalization performance by D_3. The combinational cost is usually impractically high because the cardinality $|triplets| = 4^{|D|}$. Nevertheless, CCV is used here in an explanatory purpose.

The performance can be evaluated by some comparison of predicted values $\hat{y} \in \hat{Y}$ and the known values $y \in Y$. The **loss measure** $loss(\hat{y}, y)$ depends on the data domain and the problem type. The loss over sets is defined e.g. as the mean of individual loss measurements: $loss(\hat{Y}_2, Y_2) = \text{mean}_{y \in\ Y_2}\, loss(\hat{y}, y)$. To perform an **evaluation** $eval()$ of an algorithm $\mathcal{A}$, one uses e.g. an average loss over some data subsets $D_1, D_2 \subset D_0$:

$$eval(\mathcal{A}, \theta, D_0) = \underset{(D_1, D_2) \in pairs(D_0)}{\text{mean}} loss\big(\hat{f}(X_2), Y_2\big), \tag{2.3}$$

where $\hat{f} = \mathcal{A}(\theta, D_1)$ and (usually) $D_0 = D \backslash D_3$.

Testing $test()$ consists of experimentations with hitherto unseen data D_3 using the best predictor $\hat{f}^*$ found in $D \backslash D_3$:

$$test(\mathcal{A}, \Theta, D) = \underset{D_3 \in D}{\text{mean}}\, loss(\hat{f}^*(X_3), Y_3), \tag{2.4}$$

where $\hat{f}^* = \mathcal{A}(\theta^*, D \backslash D_3)$ and $\theta^* = \ \text{argmin}_{\theta \in \Theta}\, eval(\mathcal{A}, \theta, D \backslash D_3)$.

As a summary, evaluation gives an estimate of the expected loss within the known data, whereas testing estimates the expected loss E_+ with new data D_+, which is drawn from a same assumedly stationary random source $\mathcal{D}$ as D, so that $D, D_+ \sim \mathcal{D}$:

$$\begin{aligned} E_{y \in D_0}[loss(\hat{y}, y)] &\approx eval(\mathcal{A}, \theta, D_0) \\ E_{+=} E_{y \in D_+}[loss(\hat{y}, y)] &\approx test(\mathcal{A}, \Theta, D). \end{aligned}$$

Evaluation and testing in CV can be seen as an attempt to have a compromise solution for the so called bias-variance trade-off problem (Hastie et al., 2001). Bias measured by $test()$ in this respect is a degree of unability of a relatively simple mathematical model to represent a possibly complex real life phenomenon. High variance of $eval()$ over different data sets D_0 indicates high flexibility (and low reliability) of the model. Therefore, one attempts to have both relatively low variance and relatively low bias, although, as a general rule of thumb, one could much more easily achieve low bias with high variance, or vice versa.

The **deliverable predictor** can be computed after assessing the Eq. 2.4. One is now free to use whole the data:

$$
\begin{aligned}
\theta^\star &= \operatorname*{argmin}_{\substack{\theta\in\Theta \\ (D_1,D_2)\in split(D)}} loss\big(\mathcal{A}(\theta, D_1)(X_2), Y_2\big) \\
f^\star &= \mathcal{A}(\theta^\star, D). \qquad (2.5)
\end{aligned}
$$

The practical implementation of CV hides the internal parameters and uses sequential procedural steps for testing, training and prediction. Training sets the internal parameters and possibly does some other practical preliminary computations for speeding up the prediction e.g. constructing a space partitioning structure etc. Typically, the prediction is as optimized as practically possible, and this may affect the overall design of an algorithm $\mathcal{A}$.

2.1.2 Examples of practical CV algorithms

As mentioned, CCV is impractical for larger data sets. There are several common strategies to restrict the subset enumeration:

1. leave-one-out (LOOCV): The evaluation subset D_2 is restricted to be singleton: $|D_2| = 1$.

2. leave-two-out (L2OCV) with $|D_2| = 2$.

3. k-fold CV (k-CV): Data is divided to k approximately equal parts and the subset sizes are limited to a ratio $|D_1| : |D_2| : |D_3| \approx (k-2) : 1 : 1$. Also other partitioning strategies exist, though.

4. sub-numerative testing: Some data sets may be too large even for the methods 1-3 and their variants. Then only a minor subsets of the data are being accessed in training, evaluation, testing and production of the deliverable predictor $\hat{f}^\star$. E.g. stream data analysis (Aggarwal, 2006) uses only diminutive part of the data.

5. spatial cross-validation (SCV), which is similar to LOOCV or k-CV except it is specific to the spatial data. This is why the enumeration set *triplets* is conditioned to have a geographical distance between the sets D_1, D_2, D_3 larger than a certain threshold value r. This arrangement reduces the effect of the possible spatial correlation of producing too optimistic estimates for E_+, see (Pohjankukka et al., 2017). Spatial data is different from the cases 1-4, but a similar need for limitation of subsets may occur, if there is a relatively strong correlation over a norm and between a small subset of features. The geographical location is an aspect of spatial samples introduced in the next Section.

There are various alterations to the above scheme e.g. in what comes to the optimization by arranging and embedding the rest of the nested CV, see e.g. (Mullin and Sukthankar, 2000). Another specialty is imbalanced labeling Y or the loss function, which requires some adaptations. (Chawla, 2010)

2.2 Spatial prediction

The left detail of a Fig. 2.3 depicts a spatial generalization problem, where there is a reference area with some field measurements (red squares) on a variable of interest, and a generalization area (blue circles), where the field variable is to be predicted. The spatial data samples z have a location field $p \in \mathbb{R}^2$ added: $z = (p\, x\, y)$.

The public open data is usually available in a regular rasters, e.g. the open data at the reference area has $D_A = (A,\ X_A,\ \_)$ and the generalization area has $D_C = (C,\ X_C,\ \_)$, whereas the field measurements data D_Q is typically with random locations: $D_Q = (Q,\ \_, Y_Q)$. The underscore '_' signifies a missing value here. Typically, the raster data D_A does not have the field measurements Y_A known at the raster points, unless a satellite or aerial measurement is in question.

Usually the field measurements are done in a rather small set of sample points $Q \subset \mathbb{R}^2$ with $|Q| \ll |A|$. The measurements do not necessarily coincide with the raster A. The reference area A can be subjected to several field campaigns D_Q with complex statistical and ML methods used to combine a best possible estimation $\hat{y} \in \hat{Y}_A$ over the area A. For simplicity, it is assumed in this presentation that only one set of field measurements D_Q has been performed.

The spatial generalization problem can be stated formally now: Given the reference data D_A, measurements D_Q and the generalization data D_C, predict $\hat{Y}_C$. The following is a short presentation of the spatial learning phase, spatial merge and the spatial application phase. The text is correspondent to the computational flowchart of the right detail of the Fig. 2.3.

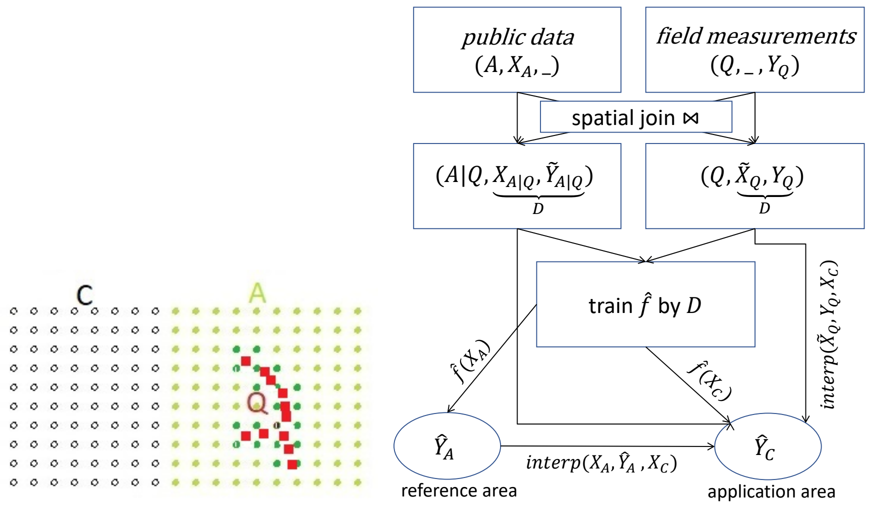

C
A
Q
public data $(A, X_A, \_)$
field measurements $(Q, \_, Y_Q)$
spatial join ⋈
$(A|Q, \underbrace{X_{A|Q}, \tilde{Y}_{A|Q}}_{D})$
$(Q, \underbrace{\tilde{X}_Q, Y_Q}_{D})$
train $\hat{f}$ by D
$\hat{f}(X_A)$
$\hat{f}(X_C)$
$interp(\tilde{X}_Q, Y_Q, X_C)$
$\hat{Y}_A$
$interp(X_A, \hat{Y}_A, X_C)$
$\hat{Y}_C$
reference area
application area

Figure 2.3: *Left*: The raster A of the reference area raster, the raster C of the intended application area, field measurement points Q (red) and the nearest raster points $A|Q$ (green) depicted. *Right*: The work flow from the reference area A to the target area C. Field measurements Y_Q are used directly or indirectly to predict $\hat{Y}_C$.

The spatial learning phase predicts the entity y at the area A, even it was measured only at the sample points Q. First, one has to use a spatial interpolation strategy to form a temporary non-spatial representation $D = (X, Y)$, which is a composition of the data D_A and D_Q. Next, an ordinary ML application proceeds, with possible minor alterations to the generation of teaching, evaluation and testing subsets, which can have added spatial constraints. Possible spatial constraints are based on raster points A and sample points Q.

There are two possible ways to use an interpolation scheme to produce the non-spatial representation: rasterization of field measurements or approximation of the raster data at the field measurements:

1. **Rasterization of measurements**: interpolate from values Y_Q to (typically 1 or 4) nearest raster locations in A to get an approximation $\tilde{Y}_{A|Q}$ to be used as a non-spatial ML data $D = (X_{A|Q}, \tilde{Y}_{A|Q})$. The raster points $A|Q \subset A$ depicted in green in Fig. 2.3 denote the neighboring raster points of Q at the raster A. The construction of $A|Q$ depends on the details of the used interpolation scheme. Interpolation methods such as the nearest neighbor (NN), the Shepard (**?**) and several others can be used in this case. There are excluded details e.g. weighting summands correctly to maintain so called unity property of interpolation when cumulating several values to one position. A pro-

cedural expression for this step is $\tilde{Y}_{P|Q} = interp(Q, Y_Q, P|Q)$, where $interp(., ., .)$ stands for an interpolation scheme for known values Y_Q at Q approximated at new locations $P|Q \subset P$.

2. **Interpolation of the raster values**: interpolate from (typically 1 or 4) nearest feature values amongst $X_{P|Q}$ to sample locations Q to form $\tilde{X}_Q$ of $D = (\tilde{X}_Q, Y_Q)$. Methods such as NN and the bilinear interpolation can be used in this case. A procedural expression for this step is $\tilde{X}_Q = interp(A, X_A, Q)$.

Both alternatives 1 and 2 can be seen as examples of a spatial version of the conditional join operator $\bowtie$ of the relational algebra (Elmasri and Navathe, 2010), where the join condition is dictated by the interpolation scheme used.

The spatial application phase to predict $\hat{Y}_C$ of a distinct new site C, $C \cap A = \{\}$ is considerably simpler. It either utilizes the predictor $\hat{f}^{\star}$ produced in the spatial learning phase or uses data sets $D_A = (A\, X_A\, \hat{Y}_A)$ or $D_Q = (Q\, X_Q\, Y_Q)$ as a vector prototype library to interpolate directly at the feature space the target area C values, see Fig. 2.3. Also, the locations p can be used e.g. for data imputation, and adaptations can be made based on statistical analysis at the base area A. The spatial prediction was used e.g. in P3 to produce the stoniness prediction over a rather large area ($30 \times 35\,\mathrm{km}^2$).

2.3 Feature extraction

The feature extraction tends to seek features X' derived from the original features X so that the original data $D = (X\, Y)$ can be substituted either by $D' = (X'\, Y)$ or $D'' = (X\, X'\, Y)$. If X' is somehow essential and hopefully having much less columns (features) than the set X, then D' is a new problem forth to experiment with. If X' somehow complements the weaknesses of the used method, D'' is a worthy new data. The feature extraction and selection phase has complications e.g. from a typical field campaign setting, where several labels Y are being measured. See an authoritative presentation on (Naula et al., 2014), where the cost aspect of observations is coupled to the prediction performance aspect.

This thesis uses two main signal types: geographic height (derived from various sources) and the video signal. The height signal is used to produce e.g. curvature features, which focus to geomorphological and microtopographical aspects. The generation of them include pattern recognition and possibly also pixel and voxel methods (if the rasterization is being done). The video signal is used to produce e.g. the athlete silhouette for gait analysis and various biomechanical metrics. That is why both of the cases D' and D'' can occur with both signal types.

Chapter 3

TIN geometry, concepts, theory and methods

Topographic height is a very concrete and a very important feature among the open nature resource data. This Chapter presents some features derived from the topographic height represented as triangulated irregular networks (TIN). The curvature has a central role in topographic and micro-topographic analysis, and all the essential concepts and most of the theoretical background of the established research will be presented. Only some essentials of the differential geometry of smooth surfaces is covered fro introducing the principal curvatures and principal directions.

Topographic height can be modeled either by a regular raster format (digital elevation models (DEM)) or by TIN (Meng et al., 2010). TIN is represented as (irregular) triangles. There are several principles to form a TIN from a point cloud e.g. windowless morphological operations of Li et al. (2017). There are also many sources of the initial point cloud, e.g. aerial laser scan, photogrammetry and terrain-based laser scan. The following text focuses on TIN as a given 3D point set.

There are several ways to represent a TIN depending on the purpose of the model and the level of space partitioning optimizations used. The basic building blocks are the cloud points $p \in P \subset \mathbb{R}^3$, triangles $t \in T \subset P^3$ and edges $e \in E \subset P^2$, where P, T and E are the sets of points, triangles and edges, respectively. An obvious way is to present a TIN as a pair (P, T), where $T = \ Delaunay(\underline{P})$ is a Delaunay triangulation (Fortune, 1997), a set of triangles created from the vertically projected points $\underline{P} \in \mathbb{R}^2$ so that a so called Delaunay property holds in T. The exact or approximate Delaunay property holds when there are no other points within a circle circumscribing a triangle. This is an adequate requirement for the numerical analysis of TINs, since it eliminates deformed triangles.

A concept of natural neighborhood follows from the Delaunay property.

Naturally neighboring points $P(p) \subset P$ of a point $p \in P$ are each connected to p by a triangle edge. The notion of natural neighborhood can be generalized to edges and triangles. A dual of Delaunay triangulation is the Voronoi tessellation (Fortune, 1997), which consists of space cells $Voronoi_P(p) \subset \mathbb{R}^3$ closest to each point $p \in P$. A TIN analysis often limits the Voronoi cells to the surfaces of TIN triangles.

Sec. 3.2 shortly presents common topographic height features. Sec. 3.4 presents general concepts and notation. Sec. 3.5 introduces several angular measures including the solid angle, which is needed for point cloud filtering and for the Gaussian curvature. Sec. 3.7 has the general TIN curvature theory and Sec. 3.8 the current implementations and research documented.

Sec. 3.8 concerns generally available, traditional GIS concepts (both grid and TIN based) concerning the curvature calculation and visualization. Sec. 3.9 has a list of published TIN methods for the curvature analysis.

3.1 Topographic height data sources

The height data originates from three sources:

1. nation-wide and publicly available digital elevation map (2m DEM) provided by National Survey of Finland (NSF). The quality of the data is generally good, but the verification of its descriptivity and fidelity when subjected to ground objects below 3 m diameter is not well established. (P2)

2. nation-wide and publicly available aerial laser scan (ALS) point cloud data by NSF. This data has good canopy penetration but rather sparse point density (approx. 0.8 m^{-2}) (P4). Points have also the intensity value and the return number, which could be used in the analysis.

3. photogrammetric point clouds provided by unmanned aerial vehicles (UAV). These are becoming increasingly important, but remain to be site-specific and limited in accessibility. The photogrammetric signal has good point density (approx. 800-3000 m^{-2}) but weak penetration. (P6) Points have the color value, which can be very useful in object registration.

A fourth height data type (not included in the thesis) is the UAV LiDAR. The energy and data communication requirements, the need for a very sophisticated and light-wight movement tracking technology and problems in producing the final point cloud make this technology still difficult for a routine production in large scale. (Karpowicz, 2016) This technology is better than photogrammetry when working under the canopy or in the mining industry.

There is not much research on the micro-topography registration and classification based on aerial point clouds, yet. Most of them are ground-based or using the lowest point of a grid slot approach (Kim et al., 2017).

3.2 Some features derived from TINs

The geographical research and the user community of geographic information systems (GIS) have spun a web of terminology and concepts relating to various height related features. It is the current opinion of the author that the four most important features are the local height, the slope angle, the aspect of the slope and curvature. These features have each various versions and each feature can be based either on a regular raster data or TIN data. In both cases the features can be considered representing aspects of an ideal ground surface. Due to noisy and sparse sampling and non-analytical surfaces of with the natural resource data, there is no chance to reproduce the exact ideal surface nor a probabilistic model of it. Still, many choices, e.g. the assumptions made to produce vertex normal points of TINs, are made with an aim to approximate the ideal surface locally.

A brief outline of each feature is given before dwelling upon the ground curvature from the Sec. 3.4 onwards. The site used is in Vihti, Finland. The LiDAR data is publicly available from NSF. The ground model was developed by SAF (P3) and sparsified to a 2m average distance between points by a method presented in (P6).

3.2.1 Local height

The local height delivers a generic overview of any geomorphological details. Basically, two hills of similar local shape should register in a similar way using this feature independent of the possible difference in the dominant height of the surrounding area. There are many possible ways to define the local height, but most of these definitions involve a horizontal scale factor, which dictates the scope of the height comparisons. A simple way to address the scale is to define a reference height $\bar{z}_r(\underline{p}), \underline{p} \in \mathbb{R}^2$ of the environment of a location $\underline{p}$ as the average on the circular perimeter with a radius r. The local height $z'(\underline{p}) = z(\underline{p}) - \bar{z}_r(\underline{p})$ is having nearly Laplacian distribution (see Fig. 3.1) with small values of r until slowly settling to the typical height distibution of the area with $r \to \infty$. Again, there are several possibilities to define the reference height $\bar{z}_r(\underline{p})$. Introducing first a reference ring with the width δ:

$$ring_r(\underline{p}, \delta) = \{\underline{q} \in P \mid r - \delta/2 \leq \|\underline{q} - \underline{p}\| \leq r + \delta/2\} \tag{3.1}$$

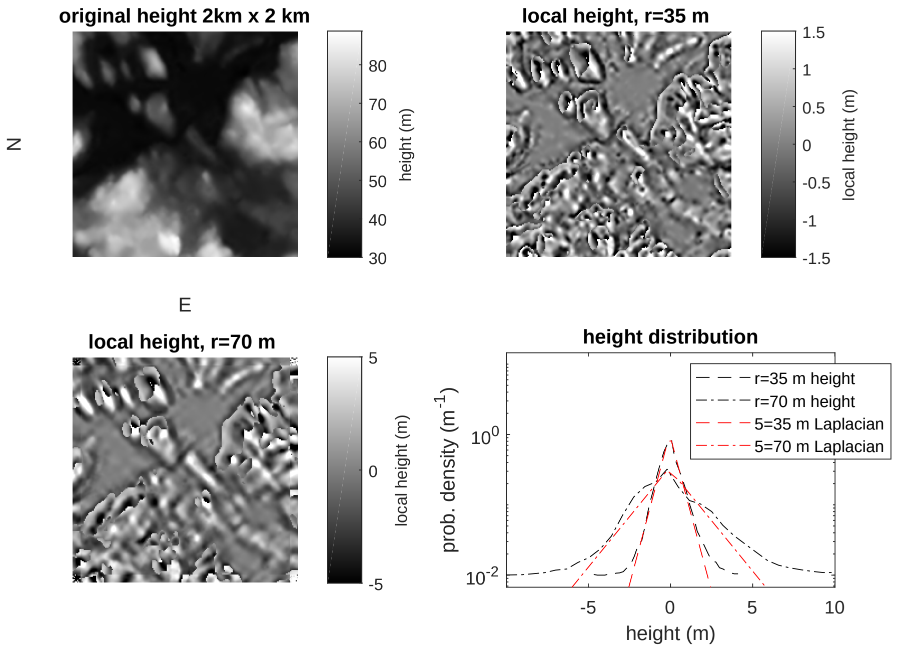


Figure 3.1: *Upper left*: A topographic height at Vihti, Finland. *Upper right*: Local height with the reference perimeter radius $r = 35$ m. *Lower left*: Local height with $r = 70$ m. *Lower right*: Local height distributions and the corresponding ideal Laplacian distributions. The root mean square (RMS) error of the ideal distributions is 27 and 6 cm, respectively.

one can have e.g. the following simple definition:

$$\bar{z}_r(\underline{p}) = \underset{\underline{q}\in ring_r(\underline{p},\delta)}{\text{mean}} z(\underline{q}). \tag{3.2}$$

Alternatively, one can fit a plane $\mathcal{P}(z, n)$ with a plane unit normal n and a height z at a point $\underline{p}$ to the data at the ring zone: $\mathcal{P}(z, n) = \{q | (q - (\begin{array}{cc} \underline{p}^T & z \end{array})^T) \cdot n = 0\}$ to the ring data:

$$\begin{aligned} \mathcal{P}(z, n) &= \{q \in \mathbb{R}^3 \,|\, (q - (\begin{array}{cc} \underline{p}^T & z \end{array})^T) \cdot n = 0\} \\ (z^*, n^*) &= \underset{(z,n)}{\text{argmin}} \sum_{\underline{q}\in disk_r(p,\delta)} w_2(\|\underline{q} - \underline{p}\|) w_1(\|q - \mathcal{P}(z, n)\|), \end{aligned} \tag{3.3}$$

where the distance measurement $\|q - \mathcal{P}\|$ can be either a) the vertical difference or b) perpendicular one (so called total linear fit) to the plane $\mathcal{P}$ and the weight function $w_1(u)$ over the projection distance u can either be of c) usual square fitting $w_1(u) = u^2$, or of d) robust fitting (several choices of $w_1(u)$ are possible), and the radial weight function $w_2(R)$ can be either

e) the uniform $w_2(R) = 1$ or any f) of the so called 'bump' functions, e.g. $w_2(R) = e^{1-(1-(\frac{R-r}{\delta/2})^2)^{-1}}$, $r - \delta/2 \leq R \leq r + \delta/2$ with vanishing derivative at the disk borders. The example in Fig. 3.1 was produced by choices b), d) and f) (local total robust linear fit with continuity property at the disk). The initial solution was computed by choices a), c) and e), though. In this approach, the local reference height is the height of the plane at $\underline{p}$:

$$\bar{z}_r(\underline{p}) = z^* \tag{3.4}$$

A useful optimization trick is to use the state (z^*, n^*) of the plane fit of a neighboring point to initialize the computation at a new point. This reduces the amount of iterations needed for the Eq. 3.3. The regular grid data allows the rapid composition of the disk point sets by relative index set operations, details of this have been omitted here.

This scheme can mimic a wide variety of commonly employed local height formulations by a suitable choice of parameters r and δ.

3.2.2 Slope angle

If the local ground normal $n(\underline{p})$ can be established meaningfully at point $p \in \mathbb{R}^3$ so that it is either smooth enough or representative enough at the discretization level used, one can derive the local slope angle $\beta(\underline{p})$ by a simple definition:

$$\beta(\underline{p}) = \text{acos}(n(\underline{p}), e_3), \tag{3.5}$$

where $e_3 = (0, 0, 1)$ is the vertical unit vector. This quantity is also known as the angle of inclination. The slope angle has been used e.g. in (P6) as one of the features. The triangle slope angle β_t follows from the Eq. 3.5 by substituting $n(p)$ by n_t.

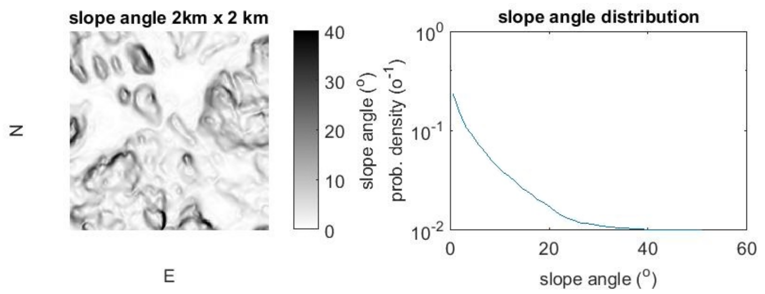


Figure 3.2: *Left*: A slope angle at Vihti, Finland. *Right*: The slope angle distribution. The area has very small elevation differences.

There is a variety of other related concepts used by GIS tradition e.g. the slope $k(\underline{p}) = \tan(\beta(\underline{p}))$ and the differential of the slope angle

$DSA = |\underline{n}(\underline{p}) \cdot \frac{d}{d\underline{p}}\beta(\underline{p})|$, where $\underline{n}$ is the orientation of the maximum inclination. The latter is being used by the GIS community especially for the visual selection of the scale of the DEM models. It is rather close to the analytical curvature H (Kreyszig, 1959) of a smooth 2D curve, when the slope is negligible: $\beta \approx 0 \rightarrow H \approx$ DSA.

DSA is also being used as a directional version for e.g. studying the local polarization of the ground for enhanced visual effects. Rudimentary tests indicate the slope angle is one of the best slope derived multi-purpose features.

3.2.3 Aspect of the slope

The *aspect* of the slope is the direction of the maximum slope, see Fig. 3.3. Using the horizontal projection $\underline{n}(p)$ of the surface normal $n(p)$ of Eq. 3.5, one can write:

$$(\cos(aspect), \sin(aspect)) = \underline{n}^0(p), \tag{3.6}$$

where the vector power zero $n^0 = n/\|n\|$ is a unit vector operator. Just like

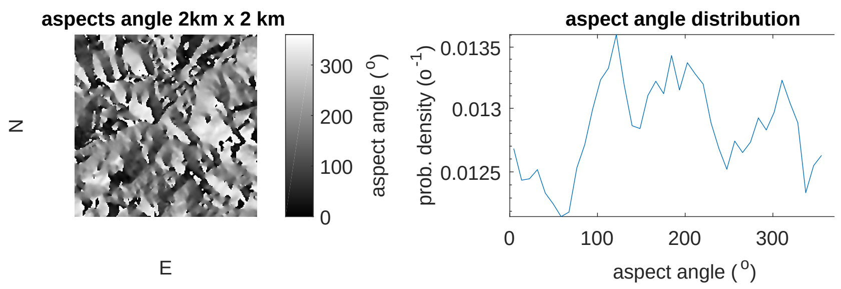


Figure 3.3: *Left*: The aspect of the slope at Vihti, Finland. *Right*: The aspect distribution. The zero value of the aspect is set to south. The ground slopes mostly to south-west. The image reveals watersheds very well.

slope, this feature is surprisingly important with phenomena dependent of vegetation (e.g. soil water dynamics) since the vegetation depends on the amount of sunlight. The aspect is best represented as a feature using the vector components of the Eq. 3.6 to avoid problems of the cyclicity of the aspect angle.

3.2.4 Ground features by other data formats than TIN

Techniques using voxels (Plaza-Leiva et al., 2017), normal vector voting, e.g. the crease detection (Page et al., 2002), establishing local shape deformation maps (Digne et al., 2017) between the sample and ideal surface, have been left out of the scope of this thesis, although they are very promising with respect to natural resource data. Likewise, many methods can be used for vegetation, canopy and tree classification e.g. the above-mentioned voxel approach (Plaza-Leiva et al., 2017) and the local shape probing (Digne et al., 2017). These methods and domains will not be covered in this work, although e.g. the SAF method (P5) can be used for canopy detection (P6) and to the classification, too.

There are also excellent techniques using vector, neighborhood or tensor voting to feed a second stage of analysis, which is either principal surfaces or principal curves (Mao et al., 2016) to detect micro-topographic features like forest machine trails (P6). These have also been excluded from this work.

3.3 Motivation for the ground curvature analysis

Stone profiles are quite similar independent of the ground slope, as long as the slope remains of small scale. The rotation needed to make a ground profile approximately horizontal is called a tilt. It was soon clear, that the functional spaces based on transforms like the Fourier family of methods (taking the definition widely and including also the wavelet approaches) fail to cope with even a moderate tilt of the sample area.

Fig. 3.4 depicts a synthetical 2D experiment constructed from circle arcs with curvatures κ (60 % of the length) and $-\kappa$ (40 % of the length), which alternate in turns. The curvature analysis fails to recover the original structure from the noisy version, whereas Fourier spectrum has an excellent match on the low frequency scale. But, the curvature analysis proves to be tilt-invariant, whereas the Fourier analysis is very sensitive to any rotation larger than 10^o. Note that this limit exceeds commonly with the naturally occurring terrain contours. This indicates that traditional image processing methods, although they are often very efficient, need to be augmented with tilt-invariant[1] methods e.g. curvature analysis.

The above example shows how the Fourier transform is extremely tolerant to noise as long as the tilt of all the samples is similar. There are several possibilities of combining the benefits of Fourier and curvature analysis. One is using a window of a local linear fit to produce an un-tilted signal for the Fourier transform. Second is geometric mesh smoothing (Yutaka and Ohtake, 2003) to eliminate noise on the required scale before the curvature histogram analysis. Third is a multi-scale curvature analysis, which produces a curvature spectrum over a range of local length scale. The result is similar to the Fourier family power spectrum but the range is physical length of each locality.

[1]Rotation invariance in image processing usually means horizontal rotations. Tilt-invariance could be called a gradient-invariance in image processing context.

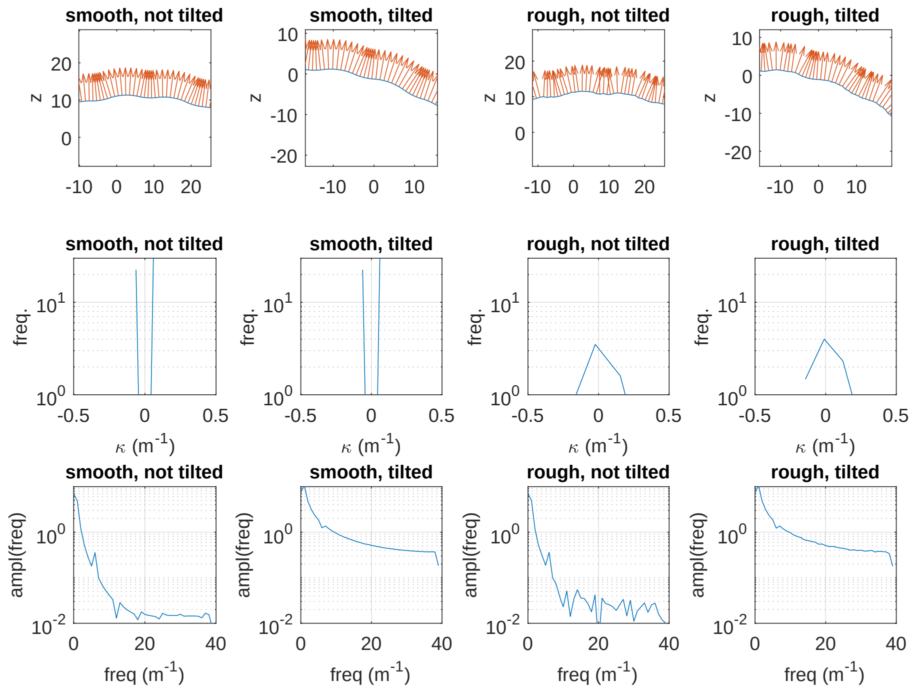


Figure 3.4: *Top row*: A synthetic 2D ground surface height profile with curvature alternating between two values $\kappa = \pm 0.8\,\mathrm{m}^{-1}$. Two curvatures appear in ratio 2:3. The tilted position (second column) has been rotated by 10^o from the original position (first column). Uniform noise with amplitude 0.15 m added (third one). *Middle row*: The observed curvature spectrum is sensitive to noise and insensitive to tilt. *Bottom row*: The obtained Fourier spectrum is sensitive to the tilt and indifferent to noise.

3.4 Concepts and notation for the TIN analysis

A rather expressive notation has to be used to cover the current field with wide range of different presentation styles and topics. Several results have to be presented in a unified way, that is why e.g. the indexing of geometric primitives has a heavy structure. It is likely that the only meaningful generally valid and generally accepted alternative to the notation chosen involves discrete differential geometry (DDG) and geometric algebra (GA) and would be theoretically too heavy for the scope of this presentation.

A triangle t is a counter-clockwise cyclically ordered set of its vertex points. When the context dictates that p is a vertex point, $p \in t = (a, b, c)$ means that $p \in \{a, b, c\}$ as by a structural membership. In some cases $p \in t$ refers to the geometric insidence relation defined by a barycentric inequality:

$$q = w_a a + w_b b + w_c c,\ 0 \le w_a, w_b, w_c \le 1,\ w_a + w_b + w_c = 1.$$

The above parameters w_a, w_b, w_c are called barymetric coordinates of t. The notational practice of using $.\epsilon.$ in a context-specific fashion extends also to other geometric entities with discrete definition and continuous geometric domain. The practice eases the presentation, but requires attention from the reader.

A point, an edge and a triangle have TIN neighborhoods of various kinds, see the list below. These are called natural neighborhoods, when the Delaunay property holds in the triangles T:

$$\begin{aligned}
T(p) &= \{t \in T | p \in t\} \text{ (NN triangles of a vertex)} \\
P(p) &= \{q \in P | \, \exists t \in T : p, q \in t\} \text{ (NN points of a vertex)} \\
E(p) &= \{(p,q) \, | \, q \in P(p)\} \text{ (the set of adjoining edges)} \\
T(t) &= \{t' \in T | \, t' \cap t \neq \{\}\} \text{ (NN triangles of a triangle)}
\end{aligned}$$

Just like triangles, an edge e as an ordered point pair $e = (p,q)$ is seen both as a linear segment in 3D and an ordered set $\{p,q\}$ when addressed by the membership relation $. \in .$. Also, edges are identified by two adjacent triangles: $e = t \cap t'$. This definition is ordered, too. The edge can be referred to by the opposite point, see Fig. 3.5. E.g. when $t = (a,b,c)$ one can refer to an $e = (b,c)$ as $e = e_{ta}$, and to point a as $a = p_{te}$. This undoubtedly heavy index notation comes handy later when presenting a collection of the currently known methods about the TIN curvature. A summary of notations concerning edges is given below:

$t \cap t'$	(an edge between two triangles)
(p,q)	(an edge between two points)
e_{tp}	(an edge opposite to a vertex p within a triangle t)
p_{te}	(a point opposite to an edge e within a triangle t)
$l_e, l_{t \cap t'}, l_{pq}, l_{tp}$	(length of an edge)
$\vec{e} = q - p$	(an edge vector of an edge $e = (p,q)$)

Some general concepts needed for the TIN analysis are presented next in the alphabetical order.

Border points $\partial T \subset P$ are points next to no data (e.g. missing or removed triangles). This is an important category, since many point clouds have often complex border point set ∂T. Note: All convex hull points belong to this set, too.

Duals Duals are usually defined using discrete differential forms (Desbrun et al., 2008) on simplicial manifolds embedded to $\mathbb{R}^3$ with intrinsic dimensionalities 0,1 and 2. This presentation is intentionally limited,

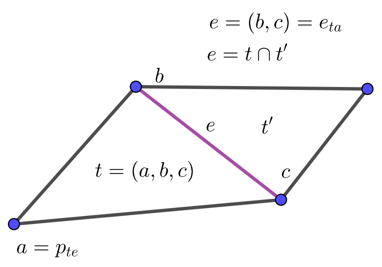


Figure 3.5: The indexing of edges. A vertex point $a \in t$ and an edge $e \subset t$ are opposite to each other.

and a dual is said to be a bijection between primal and dual TIN elements having different dimensionality. The dual counterparts cover their domain completely by their measure. When one enumerates e.g. all points $p \in P$ as primals, their duals cover whole the dual set. Fig. 3.6 gives four examples from left to right, three first ones having points as primals and duals as crossline segments, duals as a set of parts of triangle faces, and duals as edge segments. The fourth example has an edge primal and triangle faces as the dual.

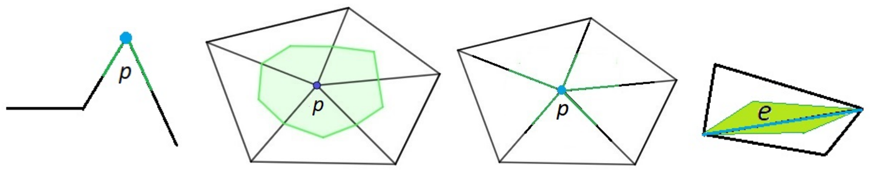


Figure 3.6: A dual of a point p on a crossline (*left*) and on a TIN triangles and edges (*middle*). An edge dual on TIN triangles (*right*).

Euclidean unit co-ordinate vectors e_1, e_2, e_3 (pointing East, North and up) are sometimes needed in the definitions and equations.

Expected size of a natural neighborhood set of a uniformly random point cloud equals 6. The ramifications of this occur occasionally in the text without further mention. Formally:

$$\mathbb{E}(|T(p)|) = \mathbb{E}(|E(p)|) = \mathbb{E}(|P(p)|) = 6. \tag{3.7}$$

Since the above statement is to be used in characterizing computational costs, a short treatise has been included here. The Euler's formula for planar graphs (West, 2000) adapted to TIN's is $|P| - |E| + |T| = 1$.

A partial (upper limit) proof of the expected size $\mathbb{E}(|T(p)|) = 6$ of the natural neighboring triangle set can be found in (Okabe et al., 2000, p.

65). An intuitive proof given here is based on the Euler characteristic of a planar connected graph made of triangles, taking into account the obvious topological and measure dependencies of a triangularization (e.g. 3 halves of edges and 1 half point per triangle), and assuming the number of the boundary edges among the convex hull of P is of lower order than the number of points $|P|$, and taking the limit at the infinity $|P| = \infty$. See Fig. 3.7, which depicts a triangle and some visual cues and arithmetical sketches at the asymptotic limit depicted by the equivalence $. \approx .$. There are about twice as many triangles as points, and the expected sizes of the natural neighborhoods $P(p), E(p), T(p)$ of a point p are about as large.

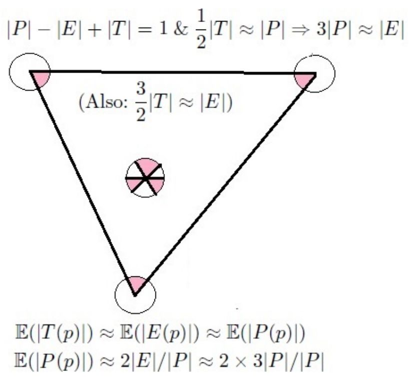

$|P| - |E| + |T| = 1$ & $\frac{1}{2}|T| \approx |P| \Rightarrow 3|P| \approx |E|$
(Also: $\frac{3}{2}|T| \approx |E|$)
$\mathbb{E}(|T(p)|) \approx \mathbb{E}(|E(p)|) \approx \mathbb{E}(|P(p)|)$
$\mathbb{E}(|P(p)|) \approx 2|E|/|P| \approx 2 \times 3|P|/|P|$

Figure 3.7: An informal explanation of the expected size of the natural neighbors assuming uniform horizontal point distribution. $. \approx .$ denotes equivalence at the infinity.

Projection matrices $P\_(v_1, v_2)v$ and $P_\perp(v)v$ projecting a vector v to a subspace spanned by vectors v_1 and v_2, and to a subspace perpendicular of v, respectively:

$$\begin{aligned} P\_(v_1, v_2) &= v_1^0 v_1^{0T} + v_2^0 v_2^{0T} && (3.8) \\ P_\perp(v) &= I - v^0 v^{0T}. && (3.9) \end{aligned}$$

Projection to the horizontal plane has been depicted by the underscore and will be used occasionally:

$$\underline{p} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} p\,. \quad (3.10)$$

Space discretization At some cases there are several possibilities to divide an area or a volume to duals of vertices and edges. A space division is called space discretization.

Triangle area A triangle $t = (a, b, c)$ has an area:

$$A_t = \|(b - a) \times (c - a)\|/2. \tag{3.11}$$

Triangle face normal A triangle $t = (a, b, c)$ has:

$$n_t = ((b - a) \times (c - a))^0 \tag{3.12}$$

as the triangle face normal. The normal is directed away from the solid (the ground, a tumor etc.) which means that the triangle vertices have to be enumerated in a counter-clockwise fashion (when seen outside the solid).

Vector arcus cos The angle between two vectors v_1, v_2: $\mathrm{acos}(v_1, v_2) = \cos^{-1}(v_1^0 \cdot v_2^0)$

Vector power zero is a notational convenience for defining unit vectors.

Vertex unit normal n_a, n_b, n_c are vertex unit normals at the indexed vertices of a triangle $t = (a, b, c)$. They are of unit length, but the orientation varies according to different authors.

3.5 Some angle definitions

Triangle tip angles ϕ_{tp}, signed edge angles β_e, projected tip angles ϕ_{pt} and the solid angle ω_p will be introduced in the following three Sections. These angles are needed in various definitions of the upcoming curvature methods.

3.5.1 Tip angles

The tip angle ϕ_{tp} at a vertex a of a triangle $t = (p, a, b)$ is the angle between edge vectors $a - p$ and $b - p$. It has been depicted at Fig. 3.8:

$$\phi_{tp} = \mathrm{acos}(a - p, b - p). \tag{3.13}$$

An equivalent notation for the tip angle is by the opposite edge. If $e = (a, b)$, then $\phi_{tp} = \phi_{te}$.

3.5.2 Signed edge angle

Edge angles β_e are signed so that the configuration depicted in Fig. 3.8 has a positive angle sign. The Fig. 3.8 has an edge $e = (p, q) = t \cap t'$ opposite to vertex points $a \in t$ and $b \in t'$ has been depicted in Fig. 3.8.

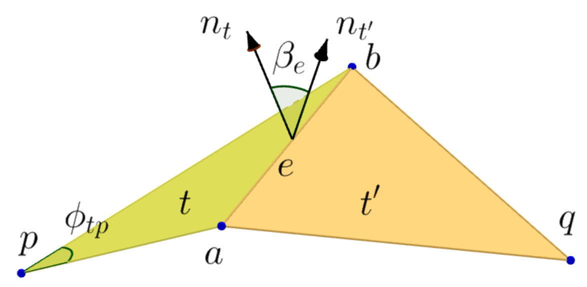


Figure 3.8: A tip angle ϕ_{tp} of a triangle t defined by the vertex point $p \in t$ and the edge $e = (a, b) = t \cap t'$. An edge angle β_e defined by the normals of the adjoining triangles t and t' has been depicted, too.

The edge sign $sgn(e)$ is positive when an edge is 'a ridge' and negative, when edge is 'a valley', see (P2, Eq. 14, p. 687). The edge sign is independent of the triangle order, triangle vertex indexing order and dependent of the triangle normal orientation (triangle normals must be always directed outside from the solid). Mathematically, the edge sign is a crucial component deciding the sign of a handedness determinant of a 3-vector blade, but practically it is a sign of the oriented distance between the point q and the affine plane spanned by t:

$$\begin{aligned} sgn(t \cap t') &= sgn((q-p) \cdot n_t),\, p \in t \cap t', q \in t' \backslash t && (3.14) \\ \beta_{t \cap t'} &= sgn(t \cap t') \operatorname{acos}(n_t, n_{t'}) \text{ (edge tilt angle)} && (3.15) \end{aligned}$$

The edge sign of Eq. 3.14 can be assumed to be widely used in CAD industry already on 1980's as a standard trick when dealing with edges of directed polygon faces.

3.5.3 Projected tip angles

A vertex p and an associated vertex normal vector n_p define a plane $\mathcal{P}(p, n) = \{q \in \mathbb{R}^3 \,|\, (q-p) \cdot n = 0\}$. A tip angle ϕ_{tp} of a triangle $t = \{p, a, b\}$ can be projected on that plane by projecting the individual edge vectors $a - p$ and $b - p$ on the plane $\mathcal{P}$:

$$\phi_{pt} = \operatorname{acos}(P_\perp(a-p), P_\perp(b-p)) \qquad (3.16)$$

The sum of the projected tip angles

$$2\pi \geq \phi_p = \sum_{t \in T(p)} \phi_{pt} \qquad (3.17)$$

equals $\phi_p = 2\pi$, when point p is not one of the border points of T, i.e. $p \notin \partial T$.

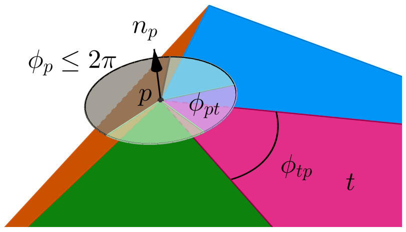


Figure 3.9: A tip angle ϕ_{tp} of a triangle t surface at the vertex p get projected to the plane perpendicular to the vertex normal n_p. The inside points have full neighborhood $\sum_{t \in T(p)} \phi_{pt} = 2\pi$.

3.5.4 Tip angle interpolation

Many equations define e.g. the curvature either at vertices or at triangles. Depending on an application, one may need to transform any feature $f \in X^T$ between vertices and triangles. A standard solution to this need is the tip angle interpolation. The most recommended strategy is to use tip angles ϕ_{tp}, $p \in t \in T$ defined on the face of a triangle t and the projected tip angles ϕ_{pt}, $t \in T(p)$ Crane et al. (2013); Mesmoudi et al. (2012) as explained in this section. The corresponding definitions in this thesis are Eqs. 3.16 and 3.13, respectively. Possible other strategies are discussed in (P5) and Crane et al. (2013).

One can define two transforms of a feature f between vertices and triangles:

$$f_p = \frac{1}{\phi_p} \sum_{t \in T(p)} \phi_{pt} f_t \tag{3.18}$$

$$f_t = \frac{1}{\pi} \sum_{p \in t} \phi_{tp} f_p \,, \tag{3.19}$$

where the total perimeter angle ϕ_p of the Eq. 3.17 gives boundary-adapting weighting for the boundary points $p \in \partial T$, see Fig. 3.9. Note that (Mesmoudi et al., 2012) uses a simplification $\phi_p \approx \pi$ for the boundary points $p \in \partial T$, which is not adequate for applications, where the boundary is dominant as with P6 (due the canopy front) and P2 (due to missing data). Although the original contributions of this thesis are mostly covered in Ch. 4, the Eq. 3.18 is presented here to improve the flow of the presentation.

The transforms of Eqs. 3.18- 3.19 have the property called the partition of unity: if $f \equiv 1$, both transformations are exact, since $\sum_{p \in t} \phi_{tp} = \pi$ and $\sum_{t \in T(p)} \phi_{pt} = \phi_p$. Transformations are also approximate inverses of each other.

Summaries of various ways to define the tip normals n_p (needed to define tip angles) is given in (Crane et al., 2013, pp. 61-67) and in (Klasing et al., 2009, Tables I and II). The methods can be divided to those using a general local fit and those using the TIN neighborhood. The recommended method of Crane et al. (2013) adopted to P2 and P4 is using the tip angles as weights for triangle normals:

$$n_p = \left(\sum_{t \in T(p)} \phi_{tp} n_t \right)^0 \tag{3.20}$$

3.5.5 Solid angle

The term 'solid angle' derives from astronomy, where the observed optical radius $0 \leq \gamma \leq 2\pi$ (in radians) of a visible roundy object is associated by the solid angle $0 \leq \omega \leq 4\pi$ (in steradians):

$$\omega = 2\pi(1 - \cos\gamma). \tag{3.21}$$

In this work, actual physical dimensions of angular measures (rad. and ster.) are left out, whenever possible. The left part of the Fig. 3.10 depicts a roundy object, the optical radius γ and the spatial angle ω.

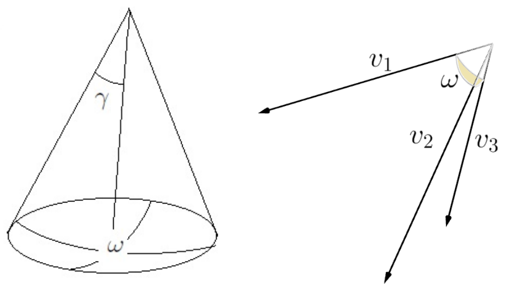


Figure 3.10: *Left*: The relationship of the solid angle ω (steradians) and the corresponding optical radius γ (radians). *Right*: A solid angle $\omega(v_1, v_2, v_3)$ defined by three vectors $v_1, v_2, v_3 \in \mathbb{R}^3$.

3.5.6 Solid angle of a vector triplet

The right part of the Fif. 3.10 depicts a solid angle $\omega(v_1, v_2, v_3) \in \mathbb{R}^3$ defined by three counter-clockwise enumerated unit vectors v_1, v_2, v_3. The solid angle equals the area of the unit sphere captured between three great circles defined by three vectors. An equation of the solid angle is given by **?** and before that by Euler, Lagrange etc.:

$$\omega(v_1, v_2, v_3) = 2 \tan^{-1} \frac{v_1^0 \cdot v_2^0 \times v_3^0}{1 + v_1^0 \cdot v_2^0 + v_2^0 \cdot v_3^0 + v_3^0 \cdot v_1^0} . \tag{3.22}$$

There is another formulation by l'Huillier (P3) with about equal computational complexity (one cross product less, 4 more tan and cos operations):

$$\begin{aligned}
\psi_i &= \operatorname{acos}(v_j, v_k),\ i, j, k \in \{1, 2, 3\} && (3.23)\\
\psi_4 &= 0 &&\\
\psi_0 &= \sum_{i=1}^{3} \psi_i &&\\
\omega(v_1, v_2, v_3) &= 4 \tan^{-1} \sqrt{\Pi_{i=1}^{4} \tan \frac{\psi_0 - \psi_i}{4}}\,, && (3.24)
\end{aligned}$$

where indices of Eq. 3.23 go through one ordered cycle of $\{1, 2, 3\}$, see Fig. 3.10.

The actual choice between Eqs. 3.22 and 3.24 depends on further details of the computation, since each one uses or produces intermediary results to and from other curvature computations presented later.

3.5.7 Solid angle of a vertex point

There are several formulations for the solid angle ω_p, given a TIN vertex point p and its surrounding triangularization $T(p)$. First of all, one can choose summand terms from Eqs. 3.22 and 3.24 to sum up vector triplet angles ω_{tp} specific to each adjoining triangle $t = (a, p, b) \in T(p)$. One triangle is depicted in the Fig. 3.11. Each contribution ω_{tp} will be then summed up to the final sum ω_p:

$$\omega_{tp} = \sum_{(a,p,b)=t \in T(p)} \omega(b - p, a - p, -e_3), \qquad (3.25)$$

$$\omega_p = \sum_{t \in T(p)} \omega_{tp}\,. \qquad (3.26)$$

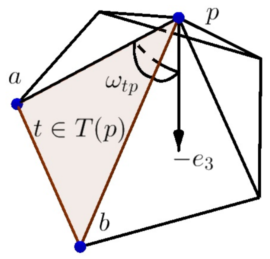


Figure 3.11: ω_{tp} as a summand of the spatial angle ω_p at a vertex t.

Secondly, there is another version of Eq. 3.26 originating from Mesmoudi et al. (2012), which uses edge angles β_e defined in Eq. 3.15. This formulation is economical if the edge angles can be re-used elsewhere (e.g. in the angle excess method of Eq. 3.48. The alternative Mesmoudi definition of the solid angle ω_p at a vertex p follows;

$$\omega_p = 2\pi - \sum_{e \in E(p)} \beta_e \,. \tag{3.27}$$

3.6 Curvature on a continuous surface

This is a minimal treatise on the curvature on continuous manifolds (lines and surfaces) in order to introduce the principal curvatures and principal directions. Fig. 3.12 depicts a black line with a changing curvature (black), and another curve (green) with the same legnth and a constant curvature.

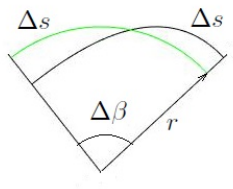


Figure 3.12: A continuous planar curve of length Δs and curve orientation change $\Delta\beta$ with varying (black) and constant (green) curvature.

The definition of the curvature of the planar curve (Kreyszig, 1959) is based on the rate of change of the curve orientation and it requires the continuity of the derivative of the curve:

$$\kappa(s) = \frac{d\beta}{ds}, \tag{3.28}$$

where $\beta(s)$ is the normal angle of the tangential of the curve and s is the geometric length along the curve. One can use the fundamental property of integration to define a local average κ_Γ of the curvature over an interval $\Gamma = [s_1, s_2] \in \mathbb{R}$:

$$\kappa_\Gamma = \frac{1}{s_2 - s_1} \int_{s_1}^{s_2} \frac{d\beta}{ds} ds = \frac{\beta(s_2) - \beta(s_1)}{s_2 - s_1} = \frac{\Delta\beta}{\Delta s} \tag{3.29}$$

Note that the definition allows edges, where there is a discrete concentration of the curvature as a constituent of the total sum $\Delta\beta$. This comes handy, when we move on analyzing discrete TIN surfaces.

3.6.1 Principal curvatures

One can use Eq. 3.28 to chart the curvature state of a point p at a smooth surface $\mathcal{M}$. A plane perpendicular to the surface, when rotated by an angle α, defines an intersection curve with the surface, see Fig. 3.13. The intersection curve has a directional curvature $H(\alpha)$ at the point p (Pressley, 2010) with the following behaviour:

$$H(\alpha) = \kappa_1 \cos^2(\alpha) + \kappa_2 \sin^2(\alpha), \tag{3.30}$$

where κ_1 and κ_2 are the principal curvatures at that point and, α is the angle with unspecified orientation. κ_1 and κ_2 are also the minimum and maximum of the directional curvature $H(\alpha)$. The principal curvatures are related to the mean curvature H and the Gaussian curvature G by:

$$H = (\kappa_1 + \kappa_2)/2 \tag{3.31}$$
$$G = \kappa_1 \kappa_2 \tag{3.32}$$
$$\kappa_i = H \pm \sqrt{H^2 - G},\ i = 1, 2\,. \tag{3.33}$$

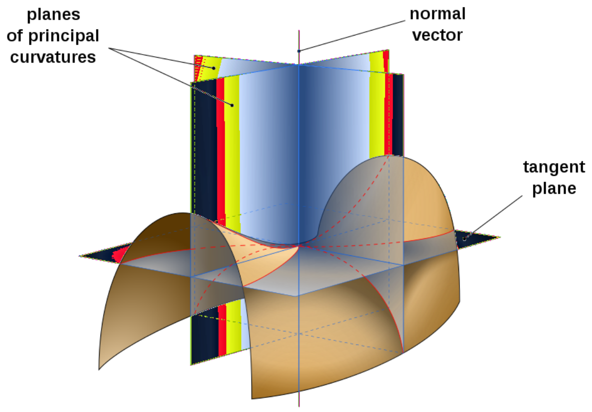


Figure 3.13: Every point has a well-defined normal, along with a plane can be spawn. The intersection of the plane and the manifold define a curve, for which extremes κ_1, κ_2 of the Eq. 3.29 can be defined. Illustration by Eric Gaba, Wikimedia commons, 2018.

For brevity, Lebesque measures meas(S), $S \subset \mathbb{R}^3$ of a manifold S are used occasionally to signify the length, area or volume of the set S depending of the intrinsic dimensionality of the set S. Also, the vector norm $\|.\|$ has been extended over sets in a straightforward fashion: $\|\mathcal{M} - p\| = \min_{q \in \mathcal{M}} \|q - p\|$, where $\mathcal{M} \in \mathbb{R}^3$ is a geometrical object.

3.6.2 Mean and Gaussian curvature

Definitions of some more curvature concepts on a once-differentiable continuous 2D manifold $\mathcal{M} \subset \mathbb{R}^3$ is in order. The expanded manifold $\mathcal{M}(\epsilon)$ can be specified by the original manifold $\mathcal{M}$ and its Gauss map image $\mathcal{M}(\epsilon)$:

$$\mathcal{M}(\epsilon) = \{q + \epsilon n(q) | \, q \in \mathcal{M}\} \tag{3.34}$$

and $n(q)$ is the unit normal of the surface $\mathcal{M}$ at $q \in \mathcal{M}$. The analysis of the whole surface $\mathcal{M}$ is uninteresting, but if one limits to a subset $\Gamma \subset \mathcal{M}$, one can define the mean curvature H_Γ and the Gaussian curvature G_Γ, both averaged over Γ. The details of the derivation are in Pressley (2010). There exists a second degree polynomial of the expansion factor ϵ: $A_\Gamma(\epsilon) = \text{meas}\{n_\epsilon(\Gamma)\}$, where the Lebesque measure meas(.) is in this case the surface area. One can introduce average principal curvatures $\kappa_{\Gamma 1}, \kappa_{\Gamma 2}$, the average main curvature $H_\Gamma = (\kappa_{\Gamma 1} + \kappa_{\Gamma 2})/2$ and the average Gaussian curvature $G_\Gamma = \kappa_{\Gamma 1}\kappa_{\Gamma 2}$, which are coupled by the following equation:

$$A_\Gamma(\epsilon) = (1 + \epsilon\kappa_{\Gamma 1})(1 + \epsilon\kappa_{\Gamma 2})A_\Gamma = (1 + 2H_\Gamma\epsilon + G_\Gamma\epsilon^2)A_\Gamma. \tag{3.35}$$

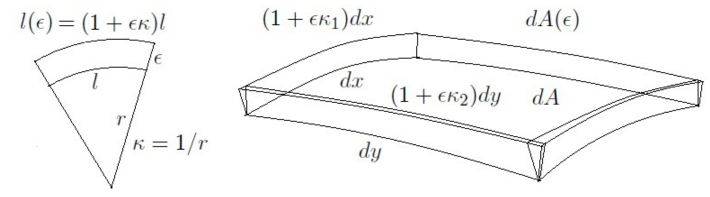

$l(\epsilon) = (1 + \epsilon\kappa)l$
ϵ
l
r
$\kappa = 1/r$
$(1 + \epsilon\kappa_1)dx$
$dA(\epsilon)$
dx
$(1 + \epsilon\kappa_2)dy$
dA
dy

Figure 3.14: *Left*: The expansion of an arc of the length l and radius r. *Right*: A schematics of the Gauss expansion of a small differential part $dA = dxdy$ of Γ. Local curvature components κ_1 and κ_2 contribute to the average curvature components of the Eq. 3.35 as shown in Pressley (2010).

A visualization of Eq. 3.35 is in the Fig. 3.14 as follows: assuming an unspecified locality $dxdy$ (can be infinitesimal or finitesimal), one can use the average line curvatures κ_1 and κ_2 on each direction x, y to estimate the expanded area meas$(\mathcal{M}(\epsilon)) = A_\Gamma(\epsilon)$. In the infinitesimal case the averaging is simply governed by a corresponding differential form. In finitesimal case Pressley (2010) proves that averages κ_1, κ_2 can be found and they are consistent i.e. the Eq. 3.35 has real number constituents.

3.7 Established approach to TIN curvature

This is an account of the currently available theory excluding the formal level of discrete differential forms of e.g. (Crane et al., 2013) and the classical DG approach of continuous surfaces using e.g. the fundamental forms of Kreyszig (1959). These disciplines are treated only informally.

Curvature, unlike the slope angle, is not dependent on any horizontal reference plane, and it can be defined on a free surface, e.g. a tumor. That is why the 'solid' side of the surface is mentioned further in the text, when orientation of normals is being discussed.

To increase the accessibility of the presentation, the formal level of DDG concepts and notation have been avoided. The DDG approach (Crane et al., 2013) can informally be summarized in the following three principles:

1. Convergence at the infinitesimal limit: the discrete-differential entities converge at the infinitesimal limit to the differential geometric (DG) results and behaviour, whenever the discretized surface is smooth. The triangulation of the TIN has to have proper measure behaviour at the infinitesimal limit. This means a sequence of TINs with increasingly smaller triangles approaches a limit value of the smooth surface.

2. Proper summation of dual measures: vertex and edge duals sum up to the total measure of the discretized surface. The summing concerns the line length and surface area measures, and duals are local measured sets (areas or lengths), which can be associated in an intuitive and natural way to the vertices and edges.

3. The formulated quantities are averages over a dual.

This presentation will focus on curvature formulations using physical dimensions such as *angle/curve length* (planar curve curvature, see Fig. 3.12), *angle/surface area* (Gaussian curvature) and *angle* multiplied by *surface length / surface area* (mean curvature).

The definition of the denominator Δs of the Eq. 3.29 is naturally difficult in the discretized setting, where the total change of orientation $\Delta\beta$ has been concentrated e.g. to a vertex point or on an edge. The DDG practitioners have a many choices for this partitioning of duals, this especially with surfaces and polytopes embedded in 3D.

A long line of geometric research started by Euler, Cauchy and Riemann have been summarized in the presentation of the motion invariant measures of convex polytopes in Hadwiger (1957). The measures are based on probabilities of other parallelopipeds of smaller dimensionality intersecting a given polytope. The results apply directly to the TIN curvature analysis as shown in Grinspun and Desbrun (2006). The convexity requirement of Hadwiger (1957) can be relaxed in this presentation since the TIN applications have no such tight concave details, which would change the polynomial coefficients of the volume perturbation introduced next.

It is shown in Hadwiger (1957), that a polynomial of the volume $V(\mathcal{O}(\epsilon))$ of a spatial expansion $\mathcal{O}(\epsilon) = \{p \in \mathbb{R}^3 | \, \|\mathcal{O} - p\| \leq \epsilon\}$ of a polytope $\mathcal{O} \subset \mathbb{R}^3$ can be developed as a sum over a spatial partitioning (dictated e.g. by triangles) of its surface $\partial\mathcal{O} = \mathcal{M} \subset T$ so that: $V(\mathcal{O}(\epsilon)) = \sum_{t\in\mathcal{M}} V_t(\epsilon)$, where $V_t(\epsilon)$ is a similarly perturbed volume of a triangle $t \in \mathcal{M}$. The perturbed spatial volume $V(\mathcal{O}(\epsilon))$ has a polynomial form:

$$V(\mathcal{M}(\epsilon)) = V + \epsilon A + \epsilon^2\ L + \epsilon^3\,\chi, \tag{3.36}$$

where V, A, L and $\chi \in \mathbb{Z}$ stand for the volume, surface area, characteristic length and the Euler characteristic number of the original polytope $\mathcal{O}$. The Euler number of a regular polytope is $\chi = 2$, but the number can vary. Especially a fragment of a polytope (e.g. an individual triangle) can have an Euler number with a real value related to its spatial angular partitioning.

Each coefficient of the polygon can be expressed as a boundary integral, which can be summed up over a dual partitioning, e.g. a TIN triangularization. The dual partitioning sets up a boundary condition for the expansion of an individual triangle, but the nature of expansion polygon coefficients being exact averages of the partition still holds. Instead of relying any canonical ways to partition space, one has to rely on common sense and relate the dual partition strategy to the goal of finding meaningful local averages.

To apply the volume expansion to TIN with concave localities, one needs a space discretization e.g. based on the edges of triangles. The edge tilt angles β_e of Eq. 3.15 have to be divided by the space discretization, and a new triangular solid angle ω_t has to be introduced:

$$\omega_t = \omega(n_a, n_b, n_c), \tag{3.37}$$

where $\omega(v_1, v_2, v_3)$ of the Eq. 3.22 (or the Eq. 3.24) is a spatial angle defined by a vector blade v_1, v_2, v_3 and ω_t is a solid angle defined by vertex normals and associated to the triangle t. The definition of the solid angle associated to a triangle t in Eq. 3.37 is dependent of the triangle index ordering: it must be counter-clockwise when seen from outside of the solid.

The contribution $V_t(\epsilon)$ of a triangle t to the expanded volume $V(\mathcal{M}(\epsilon))$ is visualized in the Fig. 3.15: the yellow face of a triangle t expands by $A_t\epsilon$, edges $e \subset t$ by $l_e\beta_e\epsilon^2/2$ (green) and the vertices (as a sum) by $\omega_t\epsilon^3/3$. All the terms can be expressed as a sum with the original polynomial structure of Eq. 3.36:

$$V_t(\epsilon) = V_t + A_t\epsilon + \frac{1}{2}\sum_{t'\in T(t)} l_{t\cap t'}\beta_{tt'}\epsilon^2 + \frac{1}{3}\omega_t\epsilon^3, \tag{3.38}$$

where the initial volume V_t, edge tilt angle fraction $\beta_{tt'}$ and the solid angle ω_t are each space partitioning related. Tilt angle fractions $\beta_{tt'}$ are depicted in Fig. 3.15. See an introductory presentation of the derivation of Eqs. 3.36 and 3.38 from Grinspun and Desbrun (2006).

There are some general constraints imposed by the space discretization, though. The edge tilt angle fractions sum up to the edge tilt angle: $\beta_{tt'} + \beta_{t't} = \beta_{t\cap t'}$. The spatial angle ω_t associated to a triangle t sums up from the fractions of opening spatial angles ω_p of vertices $p \in t$ in such a way (dictated by the dual partitioning used as depicted in Fig. 3.15), that the sum of spatial angle measures fulfill:

$$\sum_{t\in T} \omega_t \quad = \quad \sum_{p\in\ P} \omega_p \, . \tag{3.39}$$

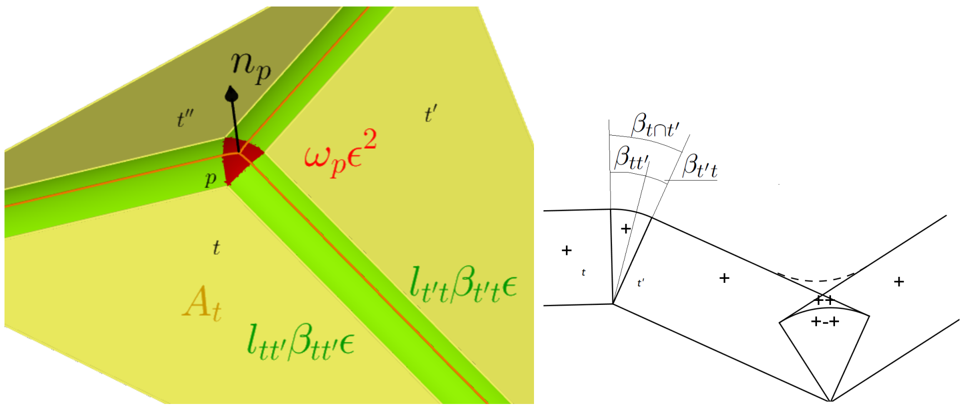


Figure 3.15: *Left*: Volume expansion $V_t(\epsilon)$ once differentiated (Eq. 3.40) has an area term A_t (yellow), a mean length term $\sum_{t'\in T(t)} l_{t\cap t'}\beta_{tt'}\epsilon$ (green) and an Euler term $\omega_p\epsilon^2$ (red). *Right*: Coefficients of the expansion polynomial are measure values, and they follow the measure summation law: $\text{meas}(C_1 \cup C_2) = \text{meas}(C_1) + \text{meas}(C_2) - \text{meas}(C_1 \cap C_2)$. One summation of the 2D case depicted is marked with '+'. The concave alignment of the vertex edges produce negative angles marked with '-' resulting in the concave cases correctly summed up, too. This holds to the 3D case too (not depicted).

One differentiation with respect to ϵ helps to match the polynomial terms of the Eq. 3.38 with the averaged curvature terms of Eq. 3.35 in the specific case of a TIN triangle $\Gamma = t$:

$$A_t(\epsilon) \quad = \quad A_t + \overbrace{\sum_{t'\in T(t)} l_{t\cap t'}\beta_{tt'}}^{2H_tA_t}\,\epsilon + \overbrace{\omega_t}^{G_tA_t}\,\epsilon^2 \tag{3.40}$$

$$H_t \quad = \quad \frac{1}{2A_t}\sum_{t'\in T(t)} l_{t\cap t'}\beta_{tt'} \tag{3.41}$$

$$G_t \quad = \quad \frac{\omega_t}{A_t}. \tag{3.42}$$

Eqs. 3.41 and 3.42 are based on volume expansion and thus they will be called **volume expansion curvatures** in this presentation. The literature and established practice uses them as a basis for formulations specific to vertex points (using a dual partitioning centered on vertices, not on triangles). Although the Eq. 3.41 is an applicable definition by itself, it is subject to many possible space discretization schemes but merely a scheme for definitions, since edge angle fractions $\beta_{tt'}$ are subject to the space discretization with many options. The volumetric approach handles the concave locales in a sensible way, see the right part of the Fig. 3.15, even the original theory of Hadwiger (1957) is based on probabilistic analysis of convex polytopes.

3.7.1 Consistence of the curvature state

Although the definitions of triangular curvatures G_t and H_t (see e.g. Eqs. 3.42 and 3.41) are valid, they fail to produce non-complex primal curvature values in some rare cases where there is large divergence in the normal vector field $n(p)$, $p \in t \in T$ created by the barymetric interpolation on a triangle t. This is an inevitable consequence of the discretization by a TIN. Fig. 3.16 shows two triangles and their normal vector field generated by the barycentric interpolation from the normal vectors at the vertices. The left triangle is from a normal situation. If one allows (right triangle) extremely large noise so that TIN at some places has close to vertical edges, the curvatures G_t and H_t computed by most of the methods presented are not compatible. One can bring the triangular Gaussian curvature G_t arbitrarily close to the upper limit $G_t \leq 2\pi/A_t$. The upper limit is a result of the TIN property, that the triangularization is never folded over itself, since the TIN is being created by a Delaunay process on the horizontal projection of the point cloud. The symmetry of the example causes $H_t = 0$ and the primal curvatures of Eq. 3.33 will be imaginary $\pm i\sqrt{2\pi/A_t}$, where i is the imaginary unit. TIN is not approaching a smooth manifold at these extreme cases.

A TIN can be regularized by any measure, which reduces the occurrence of sharp vertices with an extreme curvature state. One method relies in controlling the smoothness of the vector field - in this case the vertex normals n_p, $p \in P$. The fundamental theorem of vector calculus (Crane et al., 2013) states that every relatively smooth vector field e.g. the horizontal component $\underline{n}(p)$ of the normal field $n(p)$ can be decomposed to a curl-free and divergence-free components:

$$\underline{n}(p) = \overbrace{-\nabla\Phi(p)}^{\text{curl-free}} + \overbrace{\nabla \times \underline{n}(p)}^{\text{divergence-free}} + \overbrace{h(p)}^{\text{harmonic}} \tag{3.43}$$

The actual procedure is called the Helmholz-Hodge decomposition and it has been defined for TINs in Crane et al. (2013). In this case the potential

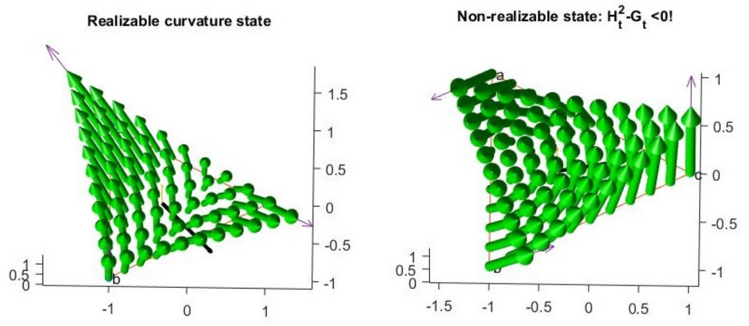


Figure 3.16: The Gauss map $n(p)$ on a triangle t. *Left*: realizable state. *Right*: an effect of extreme noise, where vertex normals are perpendicular to the triangle normal n_t.

field $\Phi(p) \approx z(p)$ where the smooth height $z(p)$ has to be implemented e.g. by Natural Neighbor interpolation (Gold, 1989). The procedure is relatively elaborate, though. Alternatives for solving the incompatibility problem are:

1. Modifying the interpolant within each triangle t in such a way, that the divergence is prevented. Currently it seems difficult to achieve a computationally efficient formulation which has the compatible curvature state with both the triangle and vertex form.

2. Expanding the domain of the interpolation scheme (by taking more triangles in the formulation) and smoothing the interpolation, so that the divergence (which is caused by the extreme height variations around a triangle disappear. This is a complex approach, especially what comes to the behaviour at the point cloud borders.

3. Eliminating the complex values as a noise (if only few such values seem to occur).

4. Formulating the Gaussian curvature G_p at vertices, as in Mesmoudi et al. (2012); Crane et al. (2013) and in P1,P3, and then using the transform from G_p to G_t (or from H_t to H_p).

3.8 GIS curvature implementations

(P1) lists some relevant methods to evaluate DEM derived from the LiDAR data. These methods are usually available in the most common GIS software. These include micro-topography measures such as standard deviation of the curvature and residual topography a.k.a local height analysis (Brubaker et al., 2013). All these approaches require the generation of

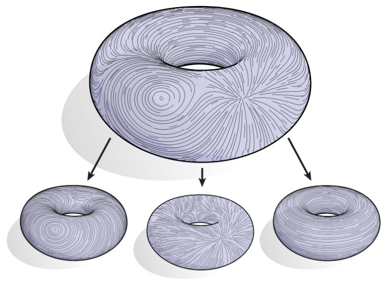


Figure 3.17: Above: The tangential component of the normal vector field. Below left: Divergence-free scalar potential component. Below middle: curl-free vector potential component. Below right: Harmonic component. Triangular averaged Gaussian curvature G_t requires the divergence-free component be eliminated. The image is a courtesy of Crane et al. (2013).

DEM from LiDAR, and using analysis window as large as 5 m reducing the possible micro-topography scale detected. It is mentioned in (Bishop et al., 2012, p. 18) that 1-2 m DEM raster size is possible in applications where the vegetation effect is negligible, e.g. the seashore change detection.

A typical GIS software provides usually several DEM curvature indicators. Some of the indicators are developed because of the efficiency in the local height filter computation or their visualization potential, and they do not necessarily relate properly to the mathematical curvature state. This is why e.g. some of the indicators are slope sensitive (these cases are mentioned in the text). For a mathematically oriented readers there is an excess of terminology, and the following explanations provide only a verbal hindsight.

Profile curvature is measured along the direction of the maximum slope and calculated from 3x3 cell. The profile curvature depends on the slope by its definition.

Planform curvature is perpendicular to the previous one and calculated from 3x3 cell. Differentiation occurs in the horizontal plane. This indicator is slope dependent, too. Profile and planform curvatures visually distinguish micro-topography well, though.

Main curvature is twice the mean curvature H. This term occurs usually in geometric and mathematical literature but is available in many GIS implementations.

Longitudinal curvature is closely related to the profile curvature, except the computations are done in a tilted coordinate system defined by the local tangential plane making it indifferent to the slope.

Cross-sectional curvature is closely related to the planform curvature, except the computations are done in a tilted coordinate system defined by the local tangential plane making it indifferent to the slope.

Tangential curvature is close to the longitudinal curvature except it accommodates a possible rotation around the slope direction on the tangential plane.

An excellent online presentation of the above definitions and their implementation as raster computations is de Smith et al. (2015).

3.9 Curvature computation on noisy TIN surfaces

A direct LiDAR analysis of curvature and slope distributions and surface roughness is very rare (P1,P3). Usually it concerns the infrastructure projects and archaelogy, and scanners are ground-based (P1).

The outdoors PC problems (micro-topography, infrastructure in the nature etc.) is addressed usually by establishing a continuous surface over a large raster window (3x3, 5x5), or by avoiding demanding formulations and using e.g. simple principles like finding the lowest point in the grid etc. A usual way is to compute H and G separately. A good reference of the most common curvature methods (covering both the TIN and the DEM domains) is Gatzke and Grimm (2006).

Meanwhile, the curvature analysis of point clouds have been established well in the technological environments. Most of the literature focuses on cases with the ratio of noise amplitude and the point density typical to built environments, miniature statues etc. The notable exception is Golovinskiy and Funkhouser (2009), but it is the opinion of the author that even this method should be improved when working with the typical recognition tasks and data concerning the wide-scale point clouds of the natural resource open data.

The following is a distilled list of the methods available to capture the curvature state.

Parameterization of a continuous ideal surface. A local fit of a convenient shape function is followed by the classic curvature analysis. This is the oldest approach (Gold, 1989). The shape functions are quadric, cubic or conic ones. The cubic fitting gives good results. These methods are usually applied to technological environment with low noise.

Spherical image is a local sphere fitting method, which approximates Gaussian curvature G_p at a vertex p using normals $n_q, q \in P(p)$. It is designed for smooth surfaces and is sensitive to noise. See (Meek and Walton, 2000).

Angle deficit Computationally very economical method to approximate G_p, which re-uses tip angles ϕ_{tp} (SAF by-products) and the sum of projected tip angles ϕ_p, see depiction in Table 3.1. The quality of estimation is good under wide range of conditions (Mesmoudi et al., 2012). Using the notation of this presentation:

$$G_p = \frac{\phi_p - \sum_{t \in T(p)} \phi_{tp}}{A_p}\,, \tag{3.44}$$

where A_p is the dual surface area of a point p. There are two options to choose A_p. The simple one splits each triangle to equal sizes:

$$A_p = \sum_{t \in T(p)} A_t/3, \tag{3.45}$$

whereas the more complex one by (Mesmoudi et al., 2012) uses Eq. 3.45 only to obtuse triangles[2], and a 3D Voronoi cell $t \cap Voronoi_P(p)$ to split all non-obtuse triangles t to surfaces with a size $A_{pt} = meas(t \cap Voronoi_P(p))$:

$$\begin{aligned}
\phi_{tp} &= \bar{\phi}_{te} + \bar{\phi}_{te'}\ \forall p \in t,\ \{p\} = e \cap e',\ e, e' \in t && (3.46)\\
A_{pt} &= \begin{cases} A_t/3, & \exists q \in t : \phi_{tq} > \pi/2 \\ (\tan \bar{\phi}_{te}) l_e^2/4 + (\tan \bar{\phi}_{te'}) l_{e'}^2/4 & \text{otherwise} \end{cases}\\
A_p &= \sum_{t \in T(p)} A_{pt}\,, && (3.47)
\end{aligned}$$

where edges e, e' flank the vertex p of a triangle t and definition in Eq. 3.46 produces a set of 3 linear equations for solving all 3 tip angle fractions $\bar{\phi}_{te}$, $e \in t$.

A 3D Voronoi cell simply encloses all points $q \in \mathbb{R}^3$ which are closer to the vertex p than any other point $p' \in P$. A typical solution strategy of the vertex area ingredient A_{pt} in a non-obstuse case requires 6 vector operations (to solve 3 tip angle fractions from a set of 3 linear equations) and it occurs roughly for 80 % of the triangles.

It is worth mentioning that Eq. 3.44 is intrinsic to the surface (using only measurements done on the surface!) only when p is an incident point i.e. fully surrounded by its neighborhood T_p (i.e. having $\phi_p = 2\pi$ logically deduced), since the actual computation of $\phi_p < 2\pi$ requires a reference to the vertex normal n_p which is an extrinsic entity. Eq. 3.44 alters the definition of concentrated Gaussian curvature of (Mesmoudi et al., 2012, Def. 1) by having ϕ freely ranging $0 < \phi_p \leq 2\pi$, whereas the original definition has $\phi_p = \pi$ for border points $p \in \partial T$ and $\phi = 2\pi$ for the inner points $p \in P \backslash \partial T$.

[2]having one vertex (any) obtuse

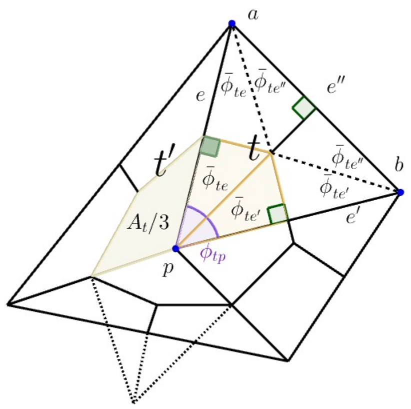


Figure 3.18: A triangle t with only acute angles: A Voronoi region of a vertex p on a triangle t is formed from two rectangular triangles with areas $\frac{1}{2}l_e^2 \tan \bar{\phi}_{te}$ and $\frac{1}{2}l_{e'}^2 \tan \bar{\phi}_{te'}$. A triangle t' with (any) obtuse angle is simply split to 3 parts with equal areas by the center point with barycentric coordinates $(1/3, 1/3, 1/3)$.

Angle excess A method to approximate G_p using edge angles β_e, see Grinspun and Desbrun (2006) and a depiction in Table 3.1. Some formulations of H_t and H_p may uses edge angles too, justifying this approach:

$$G_p = \frac{\sum_{e \in E(p)} \beta_e}{A_p} . \tag{3.48}$$

Similar formulations can be concocted to H_t, G_t and H_p, too.

Integral of mean curvature re-assembles Eq. 3.41 to refer to vertices instead of triangles by using only one half of each edge length $l_e/2$ and using the vertex dual area A_p of Eq. 3.45 instead of the triangle area A_t of Eq. 3.11. Since tip angles β_e have a sign (see Eq. 3.14 and Fig. 3.8), this is a more general method than 'integral of absolute mean curvature' mentioned at Gatzke and Grimm (2006). See the actual definition at (Mesmoudi et al., 2012, Eq. 5) and (P5, Eq. 13). A rudimentary graphical sketch is at Table 3.1.

$$H_p = \frac{1}{2A_p} \sum_{e \in E(p)} l_e \beta_e \tag{3.49}$$

Meusnier's theorem relates the mean curvature H_p and the observed curvatures $\kappa_{\mathcal{P}}$ along the intersection curve $S \cap \mathcal{P}$ formed between a continuous surface S and a plane $\mathcal{P}$ containing $p \in \mathcal{P}$ and with an inclination $\theta_{\mathcal{P}}$ to the normal n_p. The mean curvature is by Meusnier:

$H_p = \text{mean}_{\forall \mathcal{P}}\, \kappa_{\mathcal{P}} / \cos\theta_{\mathcal{P}}$ (Rashevskii, 1956). The approximation of H_p on a TIN uses 3 or more 3D Delaunay circles of $t \in T(p)$, see Chen and Schmitt (1992) and Table 3.1.

Laplace-Beltrami operator (L-B) integrated over an A_p produces an expression using edge lengths l_e and triangle tip angles $\phi_{te} = \phi_{tp}$, $\{p\} = t \backslash e$. See (Mesmoudi et al., 2012, Eq. 4) and the Fig. 3.8:

$$H_p = \frac{1}{4A_p} \sum_{e=t\cap t' \in E(p)} (\cot\phi_{te} + \cot\phi_{t'e}) l_e. \qquad (3.50)$$

The L-B operator is called the Swiss army knife of DDG! (**?**) See an outline of the coefficients used by Eq. 3.50 from Table 3.1.

Triangle tensor by Theisel et al. (2004) solves the complete curvature state within a triangle t from the very assumption that normals are interpolated by the barycentric way, see Table 3.1. Unfortunately, the values at vertices p differ when computed within different triangles $t \in T(p)$, and have to be averaged or voted from the set of all candidates. See more about this approach from (P1 App. II), (Theisel et al., 2004) and **?**.

Taubin's integral formulation by Taubin (1995b) is of $O(|T(p)|)$ (non-iterative) quadratic method, which provides a computationally economical directional curvature $H_p(\alpha)$ uses a truncated Laurent series, which forces a restriction to the orthogonality of the surrounding normals $n_q \cdot (q-p) = 0$, $q \in T(p)$ further away from the vertex p. The assumption is actually not true, but is then remedied by the multiplication factor of 2 (causing the DDG limit behaviour to be correct):

$$\kappa_{pq} \approx 2n_p \cdot (q-p)^0 / \|q-p\|,\ (p,q) = e \subset t, \qquad (3.51)$$

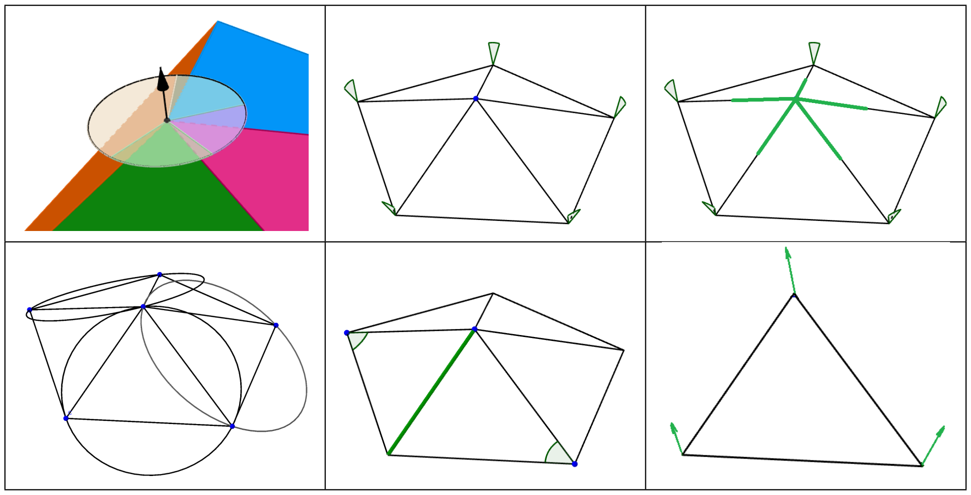
where p and q are vertex points. This approach differs from other methods, where the change rate of the vertex normals are used ($\kappa_{pq} \approx n_t \cdot (n_q - n_p)^0$), but it allows an easy formulation of the quadratic problem of solving an eigenvalue problem by letting q to range continuously on the triangle t. This leads to principal curvatures κ_1, κ_2 after a lengthy definitions. The method provides also the principal directions.

One has to perform approximately 2 vector normalizations, 21 dot products, and 15 'saxpy' operations of Golub and Van Loan (1996), totaling 38 vector operations. The smoothing of the curvature state is easy and has been expanded to cover more points, providing competitive method to (P3,P5,P6). It also has a specific smoothing method (Taubin, 1995a). Unfortunately this formulation of Taubin (1995b) was not known by the author while researching the subject.

Continuous triangle curvature (Rusinkiewicz, 2004) has a very compact presentation of an approach starting from a barycentric normal vector field and, using two fundamental forms (Kreyszig, 1959). The approach is very close to Theisel et al. (2004) e.g. what comes to computational costs except this method provides continuity and compatibility at the triangle edges and vertices. This property would be useful e.g. in eliminating noise from TIN in a controlled way.

Table 3.1 depicts six methods from the above list. Triangle tip angles ϕ_{tp}, $t \in T(p)$ and the plane of the vertex normal are the computational elements of the angle deficit method focused on vertices p. The angle excess needs edge angles β_e, $e \in E(p)$ around a vertex p to sum up the concentrated curvature to estimate the mean curvature H_p. Meusneur approximates H_p by three best fitting spheres. The Laplace-Beltrami defines a nominal length using triangle tip angles and edge lengths. The final approximation used for H_p has a form: *nominal length / dual area* The triangle tensor accesses the vertex normals only.

Table 3.1: Visualization of the geometric stage of the competing methods. From left to right: Angle deficit, angle excess, integral of mean curvature, Meusnier, Laplace-Beltrami, triangle tensor. The input of each method has been indicated by visual clues.



A more detailed summary of the above methods has been omitted. An interested reader is directed to (P3,P5), (Mesmoudi et al., 2012; Crane et al., 2013; Taubin, 1995b) and Gatzke and Grimm (2006). An approach using local smooth fit is presented in (Yang and Qian, 2007). The Laplace-Beltrami construction of the curvature is presented in (Meyer et al., 2003). The last two references are for rather smooth techno-environment cases, whereas other methods tolerate noise little better.

3.9.1 Histogram cumulation by triangles, edges and vertices

A typical ML approach for e.g. micro-topography analysis is to choose a small target area and produce a vector description from it. Then, a property (e.g. stoniness) or quality may be addressable in the vector space by some ML method, e.g. regression. A histogram of a local quantity, e.g. curvature, can be very useful in this sense since aggregates like histogram tend to reduce the noise while they allow a possible increase of the dimensionality of the signal.

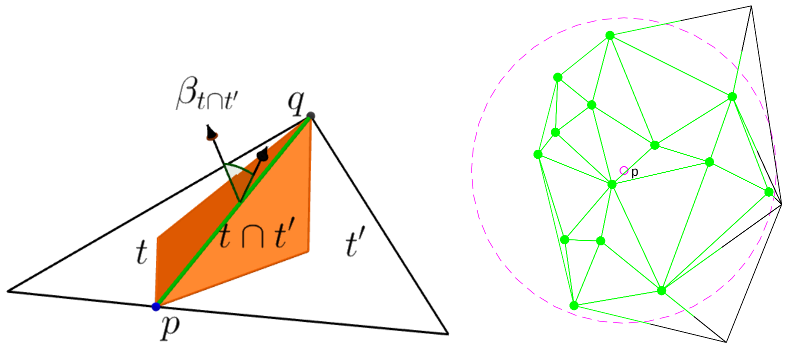


Figure 3.19: *Left*: The setting used to define the mean curvature H_e specific to an edge $e = t \cap t'$. *Right*: The average of the mean curvature $H_r(p)$ cumulated over the edges (green) within a ball centered around p and with a radius r. The Gaussian curvature average $G_r(p)$ cumulated over the points.

Given an area of interest, usually a subset $T_S \subset T$ of all triangles T, the histogram can be cumulated 1) over the triangles or 2) over edges or 3) vertices. The Gaussian curvatures can be cumulated from the discontinuities at vertices and edges (Grinspun and Desbrun, 2006). The mean curvature H_e associated to an edge $e = t \cap t'$ between two triangles t and t':

$$H_e = \frac{3\beta_e \, l_e}{A_t + A_{t'}} \tag{3.52}$$

is basically a rearrangement of the summation terms over all triangles $t \in T$ of the Eq. 3.41. Instead of triangular terms one uses the edge terms. A summation limited to a locality leads to a local average of the mean curvature, see Fig. 3.19. One can limit the edge lengths l_e (approximately) to stay within the sample. Eq. 3.52 is also directly comparable to the vertex curvature H_p of the Eq. 3.49, the latter just has summation focused on vertices. When an edge e is at the border of the triangularization, $e \in \partial T$, the denominator of Eq. 3.52 consists only one triangle area A_t, naturally. Some of the Gaussian curvatures G_p can be used in a weighted manner similar to Eq. 3.52, too.

3.10 Open problems

Possible inconsistency of the curvature state mentioned in Sec. 3.7.1 may have been a reason why such approaches like ones presented in (P1,P3,P5,P6) have not been done earlier, what comes to natural resource point clouds. The problem has been avoided usually by establishing a continuous surface over a large raster window (3x3, 5x5), or by avoiding demanding analysis of the natural resource data and staying with technological environment (indoors, man-made environment, infrastructure). A usual way is to compute H and G separately. A good reference of the most common curvature methods (covering both the TIN and the DEM domains) is Gatzke and Grimm (2006).

Meanwhile, the curvature analysis of point clouds have been established well in the technological environments. Most of the literature focuses on cases with the ratio of noise amplitude and the point density typical to built environments, miniature statues etc. The notable exception is Golovinskiy and Funkhouser (2009), but it is the opinion of the author that even this method should be improved when working with the typical recognition tasks and data concerning the wide-scale point clouds of the natural resource open data.

(P1) and (P3) introduce some current approaches to mean and Gaussian curvature analysis applicable to TINs. (P5, p. 687) aligns the mean curvature concentrated to a vertex introduced by Mesmoudi et al. (2012) to the notation used in (P5). The mean curvature at a vertex is formulated as the ratio formed by edge tilt angles around the vertex, and the area covered. Problems of this approach are the sensitivity to noise and complicated computation of the dual area. The noise sensitivity seems to arise because the formulation refers to only 6 surrounding triangles (on the average), when compared to the proposed formulation in (P5) (13 triangles). Also, the dual area requires extra calculations in the presence of obtuse tip angles of the triangle neighborhood $T(p)$. All alternative formulations are established to vertex points, whereas (P5,P6) use triangles itself.

The triangle tensor method is computationally efficient, but is one of the most sensitive methods to noise, and requires a heavy smoothing phase before applied. Typically, there is no clear connection between smoothing and the method results, and this holds to most of the methods.

An open problem is that smoothing is hard to be implemented as a monotonic procedure. Monotonicity is an essential property required by an efficient CV search. The proposed SAF method (P4) is monotonic.

Chapter 4

Curvature histograms by vertex expansion

This Chapter solves two problems in a novel way. First one is about creating a TIN, which then serves as a basis for further curvature analysis. The second problem is of how to obtain vectorizable curvature summary (e.g. a histogram) from target area consisting a subset of a TIN.

The material in this Chapter is a result of the thesis project, unless there is a comparison to contemporary methods, or a numerical illustration. A method to define the ground model (P3) will be presented in Sec. 4.1. Sec. 4.2 presents several results concerning the curvature computation based on the planar extension from (P5). Sec. 4.3 outlines an efficient directional curvature procedure (P6) for generating directional histograms over sample areas. Sec. 4.4 summarizes possibilities in curvature vectorization. Sec. 4.5 presents three synthetic and application examples. The complexity analysis in Sec. 4.6 is intended to produce a very general and unifying over a variety of methods.

4.1 Solid angle filtering

Point cloud filtering aims to a controlled sparsity in order to denoise or compress the point cloud (Xu et al., 2015). The noisy, censored and sparse nature of the natural resource data requires adaptations to existing filtering methods or completely new approaches. Fig. 4.1 depicts a cross-section projected from a real point cloud with a typical height distribution (national LiDAR, spruce forest, Kuusamo, projected stripe has 12 m width). Each point p_i in the middle image has a vertex angle $0 \leq \omega_{2D}(p_i) \leq 2\pi$ opening downwards. It is defined by the point triplet (p_{i-1}, p_i, p_{i+1}), where the triplet is ordered by the x co-ordinate. The filtering process eliminates the smallest values ω_{2D} (depicted as blue dots in the middle detail of Fig. 4.1), then largest values

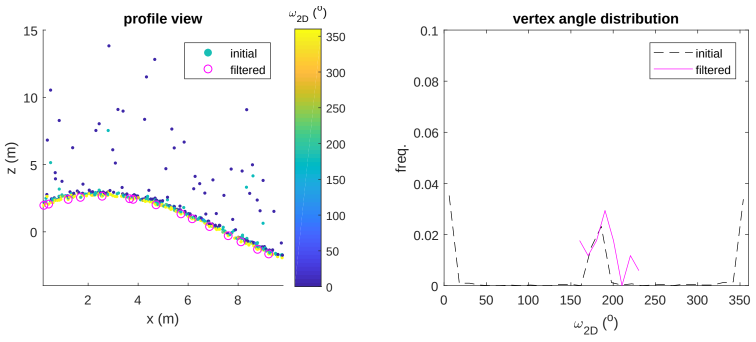


Figure 4.1: An example of angular filtering in 2D. *Left*: A vertical slice of a point cloud, x is horizontal distance and z is the height. Dot colors indicate the vertex angle ω_{2D} (exceptionally in radians for this preliminary example) of each point. The final ground points are marked with circles. The lowest points are favored in the ground model. *Right*: Vertex angle (ω_{2D}) distribution maintains some variation even after the filtering depending on the angular limits.

(depicted in yellow), until the following condition holds for all points:

$$(180 - 25)^o \leq \omega_{2D} \leq (180 + 25)^o \,.$$

The limit values of the vertex angle ω_{2D} used in the example depicted in the Fig. 4.1 are symmetrical, but this does not need to be so. The limit values have a clear regularizing and data compression effect which is outside the scope of this thesis. Solid angle filtering (SAF) is a simple generalization of this idea to 3D.

Observing the Fig. 4.1, one notices that the ground can be defined rather well by a cross-line, density of which can be controlled by the limit values of the cross-line angle ω_{2D}. The canopy top line instead is of less tangible nature and the forest biomass and the canopy study has multitude of approaches (**?**) as well as many targets. The canopy top model may has been introduced in the following, since it can be generated rather simply by the SAF process, and may suit for some practical purposes.

The SAF proceeds on a Delaunay triangulated TIN $T = Delaunay(\underline{P})$, $P \subset \mathbb{R}^3$. A simple assumption is being made that a point $p \in P$ on top of a vertex of sharp cone $T(p)$ is either not a ground point or at least not very representative as such, since it is in some sense a 'local outlier' separated by its natural neighbors $q \in N(p)$ by a steep descent along an edge. The spatial (or solid) angle ω_p (in steradians) limited by $T(p)$ and seen from p downwards inside the limiting cone of triangle surfaces forms the criterion to eliminate points from the cloud. The elimination proceeds by a simple principle: ω_p has to fall within the given limits, otherwise p gets eliminated from remaining points P. There would be various other possibilities for elimination e.g. the smallest edge or triangle slope amongst the neighboring points $N(p)$ etc. Many such criteria have been tested already by the author, and this treatise focuses on the SAF only. This is because only the SAF method has been published (P3). A brief description of the algorithm follows:

Ground TIN:
1) Eliminate pikes (drop points until $\omega_{min} \leq \omega_p$ for all remaining $p \in P$). Keep the triangularization updated accordingly during the elimination.
2) Eliminate holes (drop points until $\omega \leq \omega_{max}$ for all remaining $p \in P$). Keep the triangularization updated accordingly during the elimination.

Canopy TIN:
1) Eliminate holes and 2) pikes just like with the ground model, but using the different limits $\omega_{min}, \omega_{max}$ specific to the canopy model. Note that canopy TIN process needs the holes to be processed first!

There are two versions of SAF, an iterative batch version and a one-go version with the incremental delete. The batch version removes a point p with the spatial angle ω_p too large or small only if the value ω_p is still valid i.e. if the point neighborhood $P(p)$ is still intact. When exhausting such vertices p a new triangularization $T' = Delaunay(\underline{P'})$ of the remaining points P' and new values $\omega_{p'}, p' \in P'$ is done.

The one-go version is possible only if the Delaunay triangularization routine provides the incremental delete $T' = Delaunay(T, P, p)$. This version allows e.g. a CV process to control SAF and scan various possible spatial angle limit settings $\omega_{min}, \omega_{max}$ efficiently within the hyper-parameter optimization loop.

The additional material of (P3) (available at `http://users.utu.fi/ptneva/ALS/algorithms.pdf`) has outline of the iterative batch version of SAF. There are different (potential and realized) platforms for the algorithm, namely Matlab, python, Point Cloud Library (PCL) (Rusu and Cousins, 2011) and julia (Bezanson et al., 2012): At the writing of this thesis, the iterative (python, Matlab) and one-go (Matlab) versions have been implemented and documented in the additional material of P3.

Typical (CV validated) values for limit values of solid angle and the corresponding radial measure are given in Table 4.1. Values are not authorative but observed averages after CV tuning phase in recent few ML projects:

Solid angle approximation

Is there any possibility to use a computationally advantageous but more inaccurate approximation of the spatial angle while the elimination process is

Table 4.1: Typical spatial angle limit values $\omega_{min}, \omega_{max}$. The corresponding optical radii $\gamma_{min}, \gamma_{max}$ (in degrees) are defined in the Eq. 3.21 and depicted in the Fig. 3.10 and are used to help users of SAF to choose the values manually.

Operation	ω_{min}	ω_{max}	γ_{min} (^{o})	γ_{max} (^{o})
Ground model	4.1	11	70	140
Canopy model	π	3π	28	120

going on? The answer is yes, early stages could use the following approximant of ω_p, which is based on tip angles ϕ_{tp}, $t \in T(p)$, $p \in t$, which are also needed in the processing of curvature features after the SAF process halts. The following definitions of the tip angle sum ϕ_p and the approximate spatial angle $\tilde{\omega}_p$ are sketched in the upper part of the Fig. 4.2:

$$\begin{aligned}
\phi &= \pi/|T_p| \text{ (central half-edge angle of the equivalent regular polygon)} \\
\phi_p &= \frac{\sin(\phi)}{\phi} \sum_{t \in T(p)} \phi_{tp} \text{ (the sum of surrounding tip angles)} \qquad (4.1) \\
\tilde{\omega}_p &= 2\pi \left[1 \mp \sqrt{1 - (\frac{\phi_p}{2\pi})^2} \right], \qquad (4.2)
\end{aligned}$$

where the sign is decided upon whether p seems to be a pike or a hole (the necessary heuristics is omitted here for brevity). The approximation uses the spatial angle of a circular cone having an equivalent cone surface angle α_p. The correction term $\sin(\phi)/\phi$ is the ratio of perimeters of a regular polygon and its enclosing circle. Using the observed point cloud distributions of natural neighborhoods $N(p), T(p)$, $p \in P$ of case (P6), one gets the scatter plot of $\omega(p)$ and $\tilde{\omega}(p)$ depicted in the lower detail of the Fig. 4.2. The SAF method can use the approximant till the elimination process starts working on $\omega_p \approx \pi/2$ range (with ground TIN production, or $\omega_p \approx (4 - 1/2)\pi$ with the canopy TIN), then one has to start the exact computation by Eq. 3.26 or Eq. 3.27. The limit values for switching from the approximate spatial angle computation to the accurate one, are author's current recommendation to the future users of SAF based on the visual inspection of Fig 4.2. Note that only the lower end of the plot has been depicted in the Fig 4.2, the missing part mirrors the shown part. This optional early approximation has been implemented in (P5), although no mention of it occurs in the paper due the space constraints.

The heuristics presented in Eq. 4.2 is only one of many strategies possible at the early stages of the point cloud elimination. The early vertex elimination could be based e.g. to the minimum slope of edges $e \in E(p)$ of the edge star $E(p)$. A further study have to be subjected to these possibilities later.

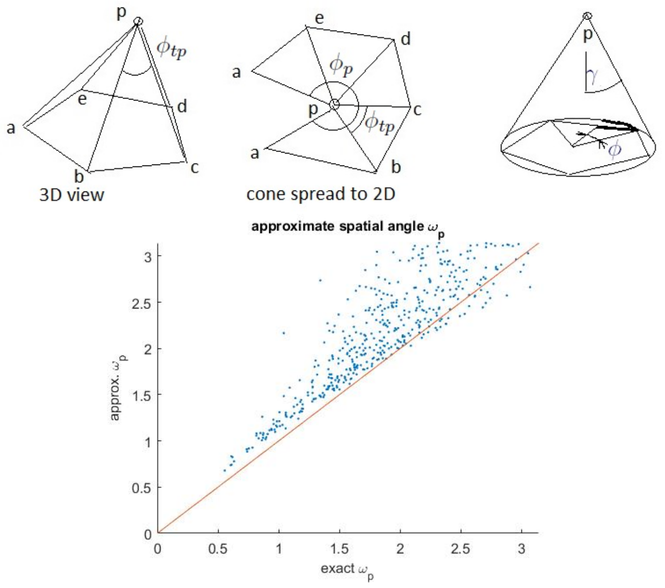


Figure 4.2: *Above*: The construction of the approximate conic, which has the same skirt angle ϕ_p as the original vertex p. *Below*: A comparison of $\tilde{\omega}_p$ and ω_p with a typical LiDAR point cloud (Kuusamo) used for TIN registration. The safe usability limit $\omega_p = \pi/2$ of the approximation corresponds to the half-cone angle $\gamma = 41^o$.

SAF end result

SAF leaves the point cloud in a filtered state. The top detail of the Fig. 4.3 shows the truncated solid angle $\omega(p)$ histogram after SAF with $\omega_{min} = 3.4$, $\omega_{max} = 9.1$. SAF is an iterative algorithm, since removed points expose some points p, which were within the spatial angle limits originally, but with a new reduced vertex neighborhood set $P(p)$ they have to be removed, too. The bottom detail of the Fig. 4.3 depicts the Gaussian curvature $G = 1/r^2$, where r is the associated curvature radius. $G = 50\ \mathrm{m}^{-2}$ yields $r = 0.15$ m, which is approximately the smallest curvature radius geometrically representable by a triangulation with an average triangle side length 0.2 m (P6). Curvatures larger than that are noise, and the amount of noise in the curvature histogram reduces towards the origo. Averages seem to be surprisingly accurate in synthetical and real world examples, e.g. the Pomokaira site ($30 \times 35\,\mathrm{km}^2$) gives the geometric mean curvature radius $r = 6365\,\mathrm{km}$ with various methods, which is close to the average radius of Earth on latitudes 68^o of Pomokaira (author's unpublished experiment).

Typically, 10-15 % of the point cloud are eliminated as non-ground hits in Northern Finland, 20-25 % in Kuusamo area (halfway on the national latitude scale), and 30-70 % in Southern Finland. The largest numbers occur at

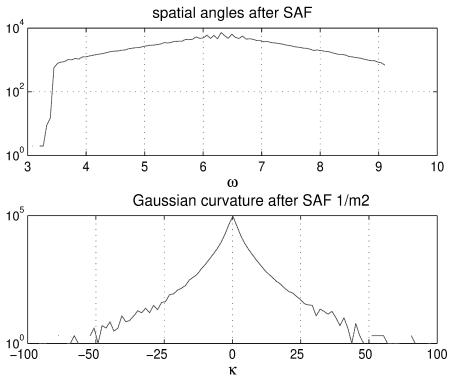


Figure 4.3: *Top*: SAF reduces the range of spatial angles ω_p in an iterative fashion. *Bottom*: Gaussian curvature computed from the resulting triangulation. Site: Vihti of (P6).

the broadleaf forests of the seashore areas. There is not yet studies of the potential of the micro-topography analysis over the whole national scale. Also, interesting connections and possibilities between various information measures of point clouds, results of SAF process with different parameterization and multi-scale ground models have not been studied yet.

4.2 Vertex expansion vs. volume expansion

There is an alternative approach (P5 Eq. 5) to average triangular curvatures by the volume expansion presented in Ch. 3.7. The surface expansion is kept piecewise planar (rounded expanded edges and vertices are omitted), see the left detail of the Fig. 4.4. This alternative is called **vertex expansion curvatures**, and it fulfills DDG requirements of convergence (see p. 36), too. As shown in Pressley (2010), Eq. 3.35 can be differentiated once and twice to get:

$$H_t = \frac{1}{2A_t}\frac{d}{d\epsilon}A_t(\epsilon)_{|\epsilon=0} \quad (4.3)$$

$$G_t = \frac{1}{2A_t}\frac{d^2}{d\epsilon^2}A_t(\epsilon)_{|\epsilon=0}\,. \quad (4.4)$$

The parameter ϵ concerns the perpendicular expansion of the surface of the triangle t. A small error will be introduced when ϵ is changed to address the expansion of vertices, with expanded triangle area $\tilde{A}_t(\epsilon)$ addressing the expanded triangle $t(\epsilon)$ after the discrete Gauss map of Eq. 3.34 $t(\epsilon) = n_\epsilon(\{a, b, c\})$:

$$\tilde{A}_t(\epsilon) \approx \|(b + n_b\epsilon - (a + n_a\epsilon)) \times (c + n_c\epsilon - (a + n_a\epsilon))\|/2. \quad (4.5)$$

By completing the differentiations, one gets (P5):

$$\begin{aligned}\tilde{H}_t &= \frac{1}{4A_t} n_t \cdot [(n_b - n_a) \times (c - a) + \\ &\quad +(b - a) \times (n_c - n_a)] \qquad (4.6)\\ \tilde{G}_t &= \frac{1}{2A_t} n_t \cdot (n_b - n_a) \times (n_c - n_a). \qquad (4.7)\end{aligned}$$

The Eqs. 4.6 and 4.7 can be taken as functions of the triangle t and its vertex normals, e.g.: $\tilde{H}_t = \tilde{H}_t(n_a, n_b, n_c)$. This is notational convenience which will be used later in the text. The surface $\partial t(\epsilon)$ of the expansion $t(\epsilon)$ defined by Eq. 4.5 is not necessarily parallel to the original triangle t, unless a triplet of altered vertex normals n'_a, n'_b, n'_c (not unit vectors!) is specifically constructed to guarantee that. Using such normals proves to be not very useful, but the construction of modified vertex normals n'_p, $p \in P$ guaranteeing the parallel expansion $t(\epsilon) \parallel t$, $t \in \ T$ is presented here for the sake of completeness. The construction uses the standard one vector $\mathbf{1} = (1, 1, 1)^T$. Any three triangles $t, t', t'' \in T(p)$, which define a nonsingular matrix $\mathbf{N}_p$, can be used. If this is not possible, the location $T(p)$ is planar and $n'_p = n_p$. See the left depiction of Fig. 4.4:

$$\begin{aligned} n_t \cdot n'_p &= 1,\, t \in T(p) \ \text{(parallel expansion condition)} \qquad (4.8)\\ \mathbf{N}_p &= (n_t\, n_{t'}\, n_{t''}) \qquad (4.9)\\ n'_p &= \mathbf{N}_p^{-T}\mathbf{1},\, p \in P. \qquad (4.10)\end{aligned}$$

These new elongated vertex normals n'_p, $p \in P$ have to be applied to Eqs. 4.6 and 4.6 to get the improved planar expansion values $\tilde{H}'_p$ and $\tilde{G}'_p$:

$$\begin{aligned}\tilde{H}'_t &= \tilde{H}_t(n'_a, n'_b, n'_c) \qquad (4.11)\\ \tilde{G}'_t &= \tilde{G}_t(n'_a, n'_b, n'_c) \qquad (4.12)\end{aligned}$$

Eq. 3.41 requires some more memory (6 scalars per triangle) and 4 scalar products, whereas Eq. 4.6 requires 7 vector operations, so the comparison between volume extended and planar extended curvatures is not clear. Eq. 3.42 defining the Gaussian curvature G_t requires one determinant and 3 inner products, while Eq. 4.7 uses one determinant and two vector operations. It seems that the comparison of the numerical quality will decide. Since the discretization is coarse in usual TIN applications, all formulas have deficiencies when operating far from the ideal realm of small angular changes. It is worthwhile to compare definitions of Eqs. 3.41, 4.6 and 3.42, 4.7 with a naturally occurring TIN models.

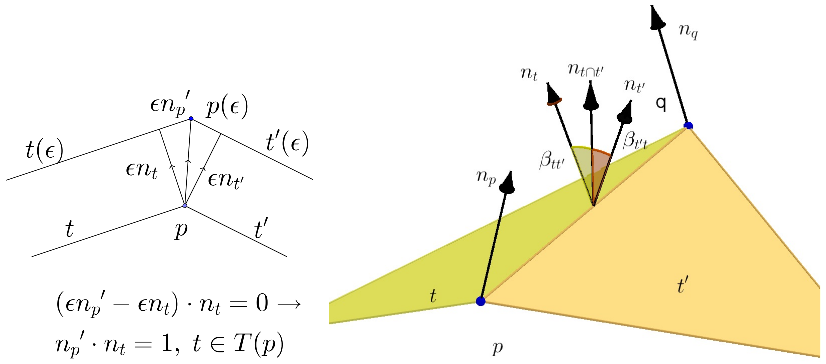


Figure 4.4: *Left*: Triangles $t_1, t_2 \in T(p)$ seen along the edge $t_1 \cap t_2$. Each triangle face is expanded a width of ϵ in a parallel fashion, and $\epsilon n'_p = p(\epsilon) - p$ is defined by the new vertex point $p(\epsilon)$ at the expanded state. *Right*: The modified space discretization divides triangles t and t' in a spiral surface in order to reduce the sensitivity to noise, when computing mean curvatures H_t and $H_{t'}$. New edge tip angle fractions $\bar{\beta}_{tt'}$ and $\bar{\beta}_{t't}$ are computed using the edge specific mean vertex normal $n_{t\cap t'}$.

Fig. 4.5 demonstrates the relationship of volumetric and planar mean curvature H_t by a scatterplot. The non-correlation in the top left detail can be explained by the increased noise level in plane expansion. A less noisy tilt angle formulation exists, but it is computationally expensive (*top right*). The volumetric H_t rounds edges and vertices, whereas the planar surface expansion keeps all edges and vertices sharp. This leads to differences in numerical values of H_t and G_t. The noise also might arise because H_t refers only to three triangles in the neighborhood $T(t)$ of a triangle t, whereas $\tilde{H}_t$ is influenced by all the triangles $T(t)$ via vertex normals. This can be checked by applying two different dual partitioning strategies to the mean curvature H_t:

1. Splitting edge angles to equal halves: $\beta_{tt'} = \beta_{t't} = \beta_{t\cap t'}/2$

2. Splitting edge angles by a spiral cut from one vertex normal to the other (see Eq. 4.15) and making sure that: $\bar{\beta}_{tt'} + \bar{\beta}_{t't} = \beta_{t\cap t'}$

The alternative 2 uses the mean of the angle between the vertex normal interpolated over an edge $t \cap t'$, and the triangle normal n_t. (The straight orange division line at the each edge in Fig. 4.4 would appear as a spiral.) The mean of the vertex normals $n_p, n_{p'}$ is given in Eq. 4.14. Using the projection matrix $P_\perp(v)$ of Eq. 3.9 to project the mean of the vertex normals to the normal plane of the edge $t\cap t'$, one is able to define the edge tilt normal fraction $\bar{\beta}_{tt'}$ of Eq. 4.15, and finally a modified version of the triangle average

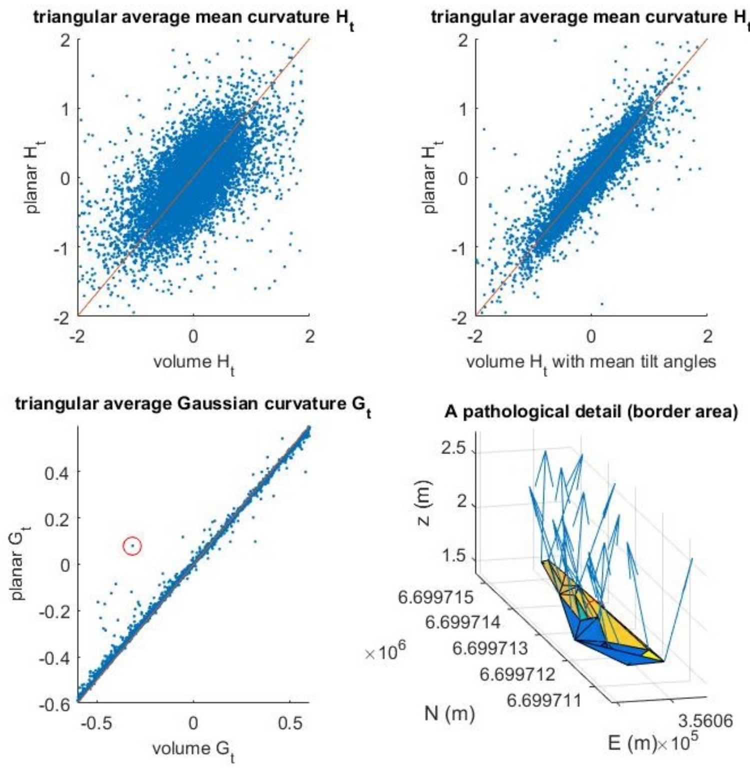


Figure 4.5: Scatter plot comparison of volume and vertex expansion. Volume expansion $V(\epsilon)$ (*top left*) and the parallel plane expansion of Eq. 4.11 *(top right)*. Gaussian curvature has an easier nature (*bottom left*). A special case of the case marked by a red circle (at *bottom left*) is at the border of the cloud where the local sample is too noisy (*bottom right*).

of the mean curvature H_t, see the right depiction of Fig. 4.4:

$$\begin{aligned}
P_{t\cap t'} &= P_{\perp}(p'-p),\ p,p' \in t \cap t' && (4.13)\\
n_{t\cap t'} &= (n_p + n_{p'})^0 && (4.14)\\
\bar{\beta}_{tt'} &= \text{acos}(P_{t\cap t'} n_{t\cap t'}, n_t) sgn(t,t') && (4.15)\\
H_t &= \frac{1}{2A_t} \sum_{t' \in T(t)} l_{t\cap t'} \bar{\beta}_{tt'}. && (4.16)
\end{aligned}$$

Similar experiments of increasing complexity seem to indicate, that the good performance of the planar expansion version $\tilde{H}_t$ of Eq. 4.6 happens indeed because of its access to large number of triangles $1 + \mathbb{E}(|T(t)|) = 13$ whereas the basic volume expansion of H_t refers only to 4 triangles. It is possible to formulate volume expansion in such a fashion, that more complex local contour features can be accommodated, though. (P5) mentions the noise filtering capabilities of the planar expansion version in relation to the so called egg container pattern. It is not mentioned in (P5) that the volume extension formulation has similar capability as long as the egg container pattern is exact. If there are even slight height differences included, the performance of the volume extension formulation collapses. The vertex expansion formulation is permutationally more advantageous as explained in (P5) but also influenced over a wider surrounding than the volume expansion formulation. Fig. 4.6 is a schematical one about the influence areas of two approaches, when triangle values f_t are in question. A similar schematics could be drawn for the vertex values f_p, but both situations are more complex.

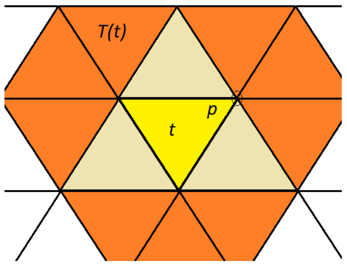


Figure 4.6: The triangle neighborhood $T(t)$ of a triangle t gets referred to via three vertex normals n_p, each referring $\mathbb{E}(|T(p)|) = 6$ triangles. The basic volume expansion curvatures refer only to 4 triangles $1+3$ depicted as yellow. The planar expansion curvatures refer to 13 triangles.

4.3 Directional curvature

The Eq. 4.6 defines the mean curvature $\tilde{H}_t(n_a, n_b, n_c)$ of a triangle $t = (a, b, c)$ with vertex normals n_a, n_b, n_c according to the planar expansion approach. The appendix A of (P6) and additional material of (P6) on the web site have a proof that when Eq. 4.6 is provided by directionally projected vertex normals $n_p(\alpha)$, $p \in t$, the result equals the projected curvature $\hat{H}_t(\alpha)$ typical to GIS DEM analysis. Denoting $\vec{\alpha} = (\cos\alpha \ \sin\alpha \ 0) \in \mathbb{R}^3$ as the directional horizontal vector and using $P\_(\vec{\alpha}, e_3)$ to project vertex normals n_p to a directional vertical plane spanned by the vectors $\vec{\alpha}$ and e_3 to new values $n_p(\alpha)$, one gets:

$$
\begin{aligned}
n_p(\alpha) &= P\_(\vec{\alpha}, e_3)\, n_p && (4.17)\\
\tilde{H}_t(\alpha) &= \tilde{H}_t\left(n_a(\alpha), n_b(\alpha), n_c(\alpha)\right),\ t = (a, b, c), && (4.18)
\end{aligned}
$$

where the nature of the projected curvature $\tilde{H}_t(\alpha)$ is explored in more detail in (P6). On the informal level of discussion, $\tilde{H}_t(\alpha)$ is a constant (everywhere at a triangle t) curvature induced by the dilatation $s(\epsilon) - s(0)$ of the expanded and original lengths $s(\epsilon)$ and $s(0)$ of directed stripes defined by the orientation α, see the Fig. 4.7, leading to:

$$\tilde{H}_t(\alpha) = \frac{1}{s(0)} \frac{ds(\epsilon)}{d\epsilon} \,. \tag{4.19}$$

Shortly, there is linear relationship between vertex normals and the mean curvature, and both the projection in Eq. 4.17 and the differentiation operation in 4.19 are linear, so the operation order can be changed to get Eq. 4.18.

A perhaps unnecessarily constructive proof of the directed curvature $\tilde{H}_t(\alpha)$ being constant on a triangle t, and its numerical value equaling Eq. 4.18 is given in (P6 App. A) and in the additional material. The proof is based on the assumption that the normal vector field $n(q)$, $q{\in}t$ is defined by the barycentric interpolation:

$$n(q) = n_a u + n_b v + n_c w,\; u + v + w = 1,\; 0 \leq u, v, w \leq 1. \tag{4.20}$$

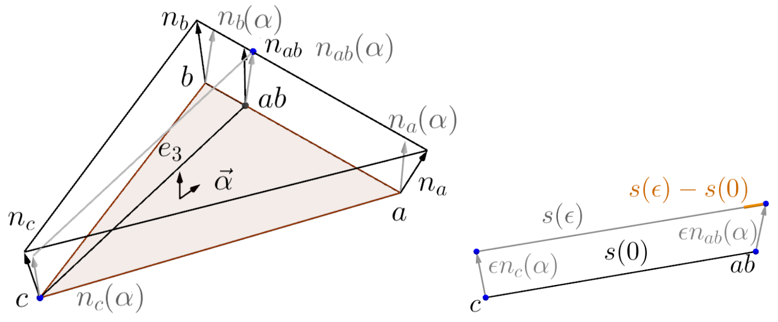


Figure 4.7: *Left*: The projection of the vertex normals to the directed plane $span\{\vec{\alpha}, e_3\}$. *Right*: A line on the expanded plane at a direction $\vec{\alpha}$ dilates defining the directed curvature, which is constant everywhere of the triangle (when using the barycentric interpolation of the Gauss map $n(q)$, $q \in t$.

Eq. 4.18 holds for all triangles $t \in T$, when $q \in t$. Gathering all vertex normals to one matrix $\mathbf{N} = \{n(p)\}_{p \in P}$ and all edge vectors $\vec{e}$, $e \in E$ to a matrix $\mathbf{W}$ using Eq. 4.6 (see P6) as an assembly rule, one can express the computation of the directional curvatures of all triangles $t \in T$ in a chosen direction α by:

$$\mathbf{H}(\alpha) = [P\_(\vec{\alpha}, e_3)\mathbf{N}]^0 \mathbf{W}, \tag{4.21}$$

where $\mathbf{H}(\alpha) = \{H_t(\alpha)\}_{t \in T}$ is the set of directional curvatures as a vector. The implementation of Eq. 4.21 is computationally efficient and enables computationally economical sampling of orientations $0 \leq \alpha \leq \pi$. Possible features and properties derived from Eq. 4.21 are: local and large-scale dominant directions (P6), minimum and maximum curvatures κ_1 and κ_2, various sample area plots like: Gaussian vs. mean curvature distributions, the principal curvatures distributions, etc. Fig. 4.8 depicts examples of some one and two variable histograms (κ_1, κ_2), (H, G), H and G.

The directional curvature $\mathbf{H}(\alpha)$ is bound to a vertical plane $span\{\vec{\alpha}, e_3\}$, which enabled an optimized matrix computation of Eq. 4.21. It is possible that some applications require binding to a plane orthogonal to the triangle t defined the triangle normal n_t. In that case there is no artificial distortion from the vertical plane, but the optimization possibility of Eq. 4.21 is lost. One has to substitute e_3 by a vertex and triangle specific vector n_{pt}, $p \in t$ in defined in the Eq. 4.17 and compute H_t of each triangle $t \in T$ separately:

$$n_{pt}(\alpha) = P\_(\vec{\alpha}, n_t) n_p \, . \tag{4.22}$$

In both cases (global e_3 projection orientation, local normal orientation) the resulting histogram has to be weighted by the triangle areas A_t.

4.4 Histogram of oriented curvatures (HOC)

Micro-topography is analyzed often in a vectorized format of the supervized learning, which uses samples $z = (x\, y)$ with a feature vector x and a label y, see Ch. 2.1 and Sec. 3.9.1. Convolutional neural networks used for finding suitable helicopter landing sites from LiDAR point cloud (?) provides a rare exception to this rule. The main issue with neural networks is expenses in generating the labels y by ground truth measurements or by expert judgement. Nevertheless, the representational power of deep learning is a great advantage, and several sub-sampling methods can alleviate the sample population problem.

Using the combination of the mean curvature by vertex expansion in Eq. 4.6 and the cumulation of the global directional curvature Eq. 4.21 subjected to small localities is called a method of histogram of oriented curvatures (HOC). It is intended in the analysis of the micro-topography of sample areas of various sizes. The name of the method derives from the histogram of oriented gradients (HOG) (?), (?). Localized histograms (and other aggregates) are gaining popularity, since they dampen noise, are applicable to both vectorization and to deep learning (?), (?) and are resilient to missing data.

Fig. 4.8 presents a curvature analysis of one positive and one negative sample of glacial liquefaction spread. A liquefaction of soil happens when soil and water form a two-phase liquid on enough large an area that it is geographically significant feature. The feature is hard to detect from the ground, though.

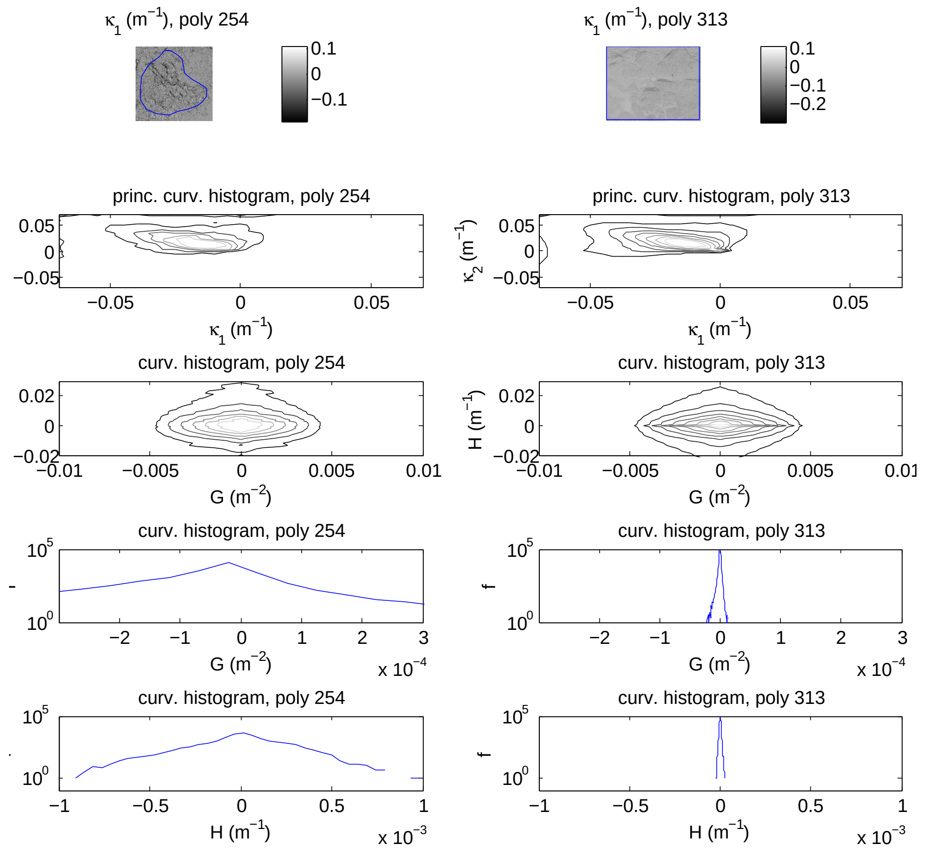


Figure 4.8: A liquefaction area at Kuusamo, Finland. Referring to subimages by the reading order: 1: κ_1 of a liquefaction spread (a positive sample). 2: κ_2 of a a neutral site (a negative sample). 3-4: principal curvature histograms (κ_1, κ_2), 5-6: (G, H) histograms. 7-8: G histograms. 9-10: H histograms. In this case taking only the s.t.d. of G or H suits well as a vectorization strategy.

There are four one-dimensional histogram cumulations possible, see Fig. 4.8. These are two two-dimensional ones with (H_t, G_t) and $(\kappa_{t1}, \kappa_{t2})$, and two angular cumulations of the principal orientations $(\kappa_{t1}, \kappa_{t2})$, all cumulated over a sample $S \subset T$.

4.5 Numerical experiments

The volume expansion method deals with curvatures on an assumed surface, which goes through the vertex points (at the limit of increasing point density). The vertex expansion addresses a surface, which osculates the triangle face centers. This will be demonstrated by the example 1, where a regular polygon is used as an illustration. Aggregative methods like HOC require that the summation of consitutients is close to additive. This means that e.g. the cumulated mean $(\tilde{H}_t + \tilde{H}_{t'})/2$ (see Eq. 4.6) of two neighboring triangles t, t' should be close to the approximate mean curvature $H_{t \cup t'}$ of the two triangles as a whole. This property of HOC is demonstrated in example 2 using bilinear formulation of the normal vectors $n(q)$, $q \in \cup t'$. The example 3 completes the height feature list of the Vihti area started at Sec. 3.2.

Example 1: Qualitative comparison in 2D

What is the most natural definition of the reference curvature when comparing the volume extension and the planar extension approaches on a regular polytope? The curvature state is after all a property of an ideal surface passing through or nearby the sample points and that ideal surface is basically unknown, and sometimes even a non-computable entity.

There is a definition of the 2D curve curvature based on Gauss map, which is analogous to the planar expansion derivative[1] of Eq. 4.3. A curve segment S with the total length $s(0) = \text{meas}(S)$ is Gauss mapped by a distance ϵ. Then, the average curvature κ over the total length equals:

$$\kappa = \frac{1}{s(0)} \frac{ds(\epsilon)}{d\epsilon}, \tag{4.23}$$

where the expanded length $s(\epsilon) = \text{meas}(S(\epsilon))$ and $S(\epsilon) = \{q + \epsilon n(q) \mid q \in S\}$ is the curve shifted ('expanded' or Gauss mapped) by $\epsilon n(.)$, and $n(.)$ is the local unit normal of the curve S.

There are three possibilities to subject 2D polylines to Eq 4.23, see Fig. 4.9. Each of them is analogous to some of the already presented 3D strategies of computing mean curvature H_t, the relevant Eqs. are mentioned in each case. The length extend $s(\epsilon) - s(0)$ has been depicted in different colors in Fig. 4.9:

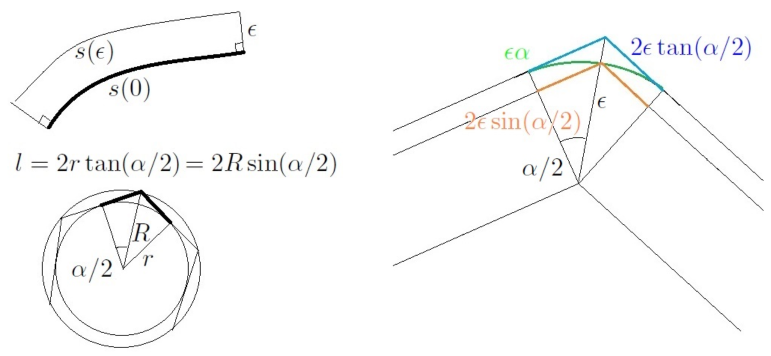
1. 'volume' expansion i.e. rounding depicted in green, see Eqs. 3.28, 3.35 and 3.41.

2. 'planar' expansion depicted in orange, see Eqs. 4.3 and 4.6.

3. improved 'planar' expansion with elongated vertex normals depicted in blue, see Eq. 4.11.

Eq. 4.23 can be applied in each of the three cases and related to either the inner fitting circle with radius r or the outer fitting circle with radius

[1]Eq. 4.3 actually follows from the volume expansion principle, but is named differently due to its application to the planar expansion.



$s(\epsilon)$
ϵ
$s(0)$
$l = 2r\tan(\alpha/2) = 2R\sin(\alpha/2)$
R
r
$\alpha/2$
$\epsilon\alpha$
$2\epsilon\tan(\alpha/2)$
$2\epsilon\sin(\alpha/2)$
ϵ
$\alpha/2$

Figure 4.9: *Left above*: Gauss mapped 2D curve S with an original length $s(0)$. *Left below*: Two possible circles fit to a cross-line with constant vertex angles α and constant segment lengths l. A vertex point has the length of its dual (equaling l) marked with a thick line zone. *Right*: A detail of a vertex when using three different strategies to approximate $S(\epsilon)$.

R. Results given as truncated Taylor series developed at $\alpha = 0$ are in the Table 4.2. Differences with the alternative ideal curvatures $\kappa = 1/r$ or $\kappa = 1/R$ are small ($< 10\,\%$) when assuming practical polyline vertex angles $\alpha \leq 50^o$.

Table 4.2: Relation of 2D curve expansion strategies to the curvature κ defined by the inner and outer fitting circle. The first missing terms are of magnitude $O(\alpha^4)$.

	κ	
Case	inner circle	outer circle
1	$(1-\alpha^2/12+...)/r$	$(1+\alpha^2/24+...)/R$
2	$(1-\alpha^2/8+...)/r$	$1/R$
3	$1/r$	$(1+\alpha^2/8+...)/R$

One could argue that the outer circle fit (case 2) is more natural in the analogous 3D problem of natural resource data with relatively sparse points, which assumedly really represent samples of the ground. This argument favors the proposed planar expansion strategy implemented by Eq. 4.6 (P3,P5). In case of really dense data and relatively smooth surfaces (traditional technological environments and targets) one can assume a Gaussian (or at least symmetric) noise, which would favor the volume expansion (case 1), since its curvature estimates are between inner and outer circles. The improved planar expansion (case 3) could be useful in cases, where the noise level is high, but also the density is rather high, and the geometric objects to be detected are close to the signal density range.

The aesthetical choice or a choice of an expert in a specific 2D application could be either of the three presented, yet e.g. the existing national LiDAR clouds (with a density $\rho \approx 8\,m^{-2}$) seem to be best to handle with the alternative 2. The cases with symmetrical noise give a chance to try to define an ideal surface staying between two extremities $1/R < \kappa < 1/r$. Chasing the properties of the ideal surface lead to approaches such as information geometry (Nielsen, 2010) and principal surfaces (Chang and Ghosh, 2001), both of which are deliberately left out from this work.

Example 2: Curvature histogram cumulation on small triangle sets

In an infinitesimal setting on a continuous surface $H = \text{mean}_{0\leq\alpha\leq\pi} H(\alpha)$ i.e. an average of all curvatures of 2D curves passing through a point, when curves are created by cutting the surface by vertical planes. This is a consequence of Eq. 3.30. The directional curvature $\tilde{H}_t(\alpha)$ of Eq. 4.6 has the same property, proven in a constructive way in the supplementary material of (P6). Also, it behaves as the mean of the constituent triangles of a bilinear interpolation scheme on a regular grid. A visual demonstration of this has been provided in Fig. 4.10. The average of the mean curvatures of adjacent triangles (left detail) equals the numerical average of the mean curvature of the bilinear interpolation (right detail).

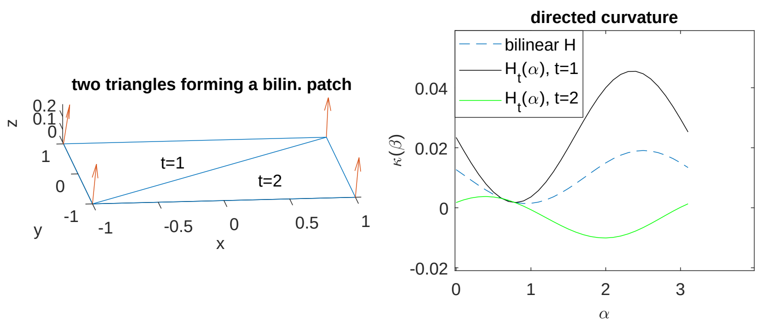


Figure 4.10: *Left*: Triangles 1 and 2 with their normal vectors forming a bilinear interpolation patch. *Right*: Bilinear interpolation over two adjacent triangles gives the average of the directed curvature $H_t(\alpha)$ of triangles $t = 1$ and $t = 2$.

This example serves as a hint how the expected mean of the mean curvature of a HOC very accurately equals the expected mean curvature produced by a bilinear interpolation of the Gaussian map $n(S)$ over a regular grid square S. A more complex analysis using the perturbation theory (not included in the Thesis scope) seems to prove that also the expected variance would equal in two cases: TIN based on a regular grid and a bilinear interpolation based on a regular grid (DEM). This means that the HOC can perform better than a DEM immediately when the sampling density exceeds the DEM sampling density. Further analysis requires case-by-case comparison of two data sources.

Example 3: Principal curvatures at Vihti

Eqs. 3.41, 3.42 and 3.33 were implemented in order to demonstrate the principal curvatures. Edge angle fractions $\beta_{tt'} = \beta_{t \cap t'}/2$ were split to exact halves. Fig. 4.11 depicts the same Vihti area with the principal curvatures κ_1 and κ_2.

An important observation at this case is that κ_1 obviously detects an aspect of micro-topography not revealed by other three height derived features (local height of Eq. 3.1, slope angle of Eq. 3.5, aspect of the slope of Eq. 3.6 and κ_2), see Figures 3.1, 3.2 and 3.3 at pages 20, 21 and 22, respectively.

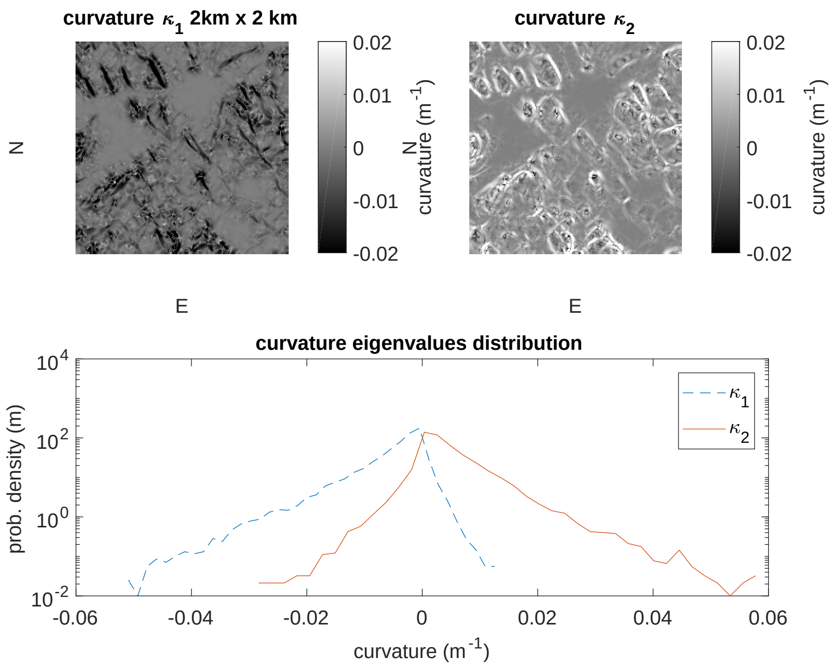


Figure 4.11: *Above*: Principal curvatures κ_1 and κ_2 at Vihti, Finland. *Below*: The distributions of the principal curvatures over the same area.

Results in Examples 1 and 2 are novel (not included in the publications P1,P3,P5,P6) and seem to verify that the vertex expansion is an applicable new basis for the curvature analysis. The example 3 would need a similar DEM analysis to compare the two data sources (aerial point clouds and DEM).

4.6 The computational costs of the TIN curvature

A summary of most of the possible methods and formulations, and the computational complexity of likely combinations is given here. Some comparisons to the established methods are given, too. The presentation evolves around the graph depicted in Fig. 4.12, which provides a strategic overview on the computation order and costs. The graph is annotated with computational costs and referred to by the Table 4.3 where the actual summary of the costs of the curvature properties has been given. Analysis is rather coarse focusing on counting vector (v) operations (affine transform $a + bv$ with scalars a and b, inner product, norm). This is rather close to 'saxpy' used in Golub and Van Loan (1996). Some details e.g. solving linear equations, or several scalar summations etc.) are counted in as roughly equivalent vector operations. The summary excludes computation of the principal orientations except with Taubin's integral formulation, where the eigenvalue problem is built in.

Fig. 4.12 has SAF related computation inside the gray sack. The initial Delaunay process has $O(n \log n)$ cost, and later iterations reduce the point cloud by $O(kn)$ costs per iteration. Grey circles (the solid angle ω_p and the slope angle β_t of the Eq. 3.5) can serve as outputs Y by itself. There are two possible ways to compute the vertex dual area A_p, and the choice depends on whether triangular values A_t, n_t are actually needed.

E.g. the cost of the arrow $t \rightarrow A_t, n_t$ in the Fig. 4.12 equals $3n$: one cross-product (approx 2 inner products) and one vector norm. Each arrow is given the cumulated costs of its input data and the calculation step itself.

Each curvature method has a set of inputs which can be found from the Fig. 4.12. E.g. the mean curvature $\tilde{H}_t$ of the Eq. 4.6 has the structure:

$$\tilde{H}_t = \frac{1}{4A_t} \cdot [() \times () + () \times ()]$$

and it has 2 cross products (approx. cost 3 dot products each), one dot product and one scalar-vector multiplication i.e. approx. $2 \times 3 + 1 = 7$ cost factor units, see first row of the Table 4.3. Although the cost measure is coarse and does not take into account many possible processing environment optimizations, it is similar to all methods, and has some indicative power.

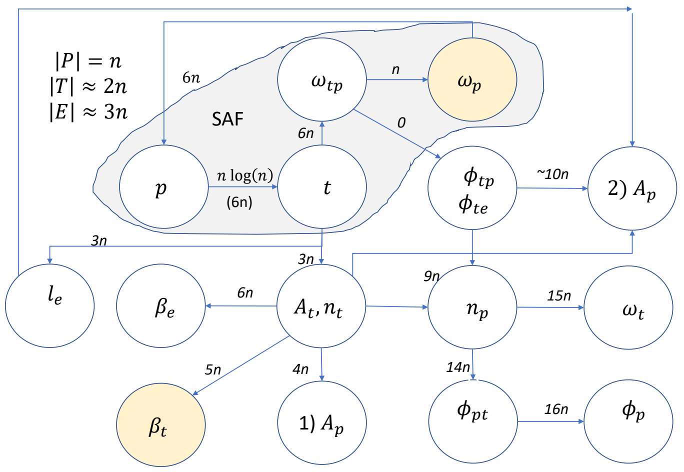


Figure 4.12: The order of computations of the preliminary parameters. Two end results highlighted. The cumulative computational demand has been characterized in approximate vector operations (+, inner product, norm).

Table 4.3 has a summary of the approximative costs of each method mentioned here. The cost factors of the last row can be conctructed by following the example given in the previous paragraph.

The cost is factored to each end result as it were the only output. This is unfair to some methods, which produce several useful inputs to other methods with only a small additional cost. A more thorough analysis coupled with the noise sensitivity aspect would be in order. There are problem types in micro-topography analysis, where e.g. the local distribution of the principal directions is important.

One needs at least a pair (H, G) of methods to capture the curvature state properly. The vertex expansion and L-B has temptingly good efficiency, but they are not a consistent pair: a large number of vertices would have imaginary principal curvatures without heavy smoothing. A transform from triangle values to vertex values costs approx. 2 vector operations, which has to be taken into account when finding the best pair of methods.

Most experience from the computational speed comes from data of (P3), which consists of aerial point clouds over an area of 30×35 km. The triangle tensor method and the proposed $\tilde{H}_t$ computed approx. 1 km^2 (0.8×10^6 points) in 10 minutes, whereas the L-B method was approx. 30 % slower. This was due to the computational speedups vector formulation can provide vs. the trigonometric and arithmetic of the L-B method. This example sheds light to the approximate nature of the vector operation column of the Table 4.3.

The suitability of the methods for a HOC like oriented mean curvature generation varies. The method of (Taubin, 1995b) would be the most efficient for a very high grained HOC analysis, but has a large initial computational demand and sensititivy to noise. (Theisel et al., 2004) is close to impractical what comes to noise sensitivity. The L-B method requires a compiled environment, since its formally good efficiency requires fast arithmetic, not vector computations. By many practical considerations, the proposed pair $(\tilde{H}_t, \tilde{G}_t)$ is a very good one, especially with the HOC algorithm. All possible combinations and specific needs have not been inspected yet, though. Especially the efficiency of computing the principal directions have not been tested yet with a large set of the methods.

Table 4.3: The computational demand of Gaussian and mean curvature by various methods. Methods originated from this study are in boldface. Each result has been assumed to be computed independently. Results with (*) have been transformed to vertices via Eq. 3.18. Excluded methods have costs higher than 35 vector operations per vertex.

	description	page and Eq.	parameters	v. oper./n
$\tilde{G}_t$	**vertex exp.**	p. 57 4.7	n_p, n_t	7
H_p	L-B	p. 45 3.50	A_p, ϕ_{te}, l_e	7..13
G_p	angle excess	p. 44 3.48	A_p, β_e	10..16
$\tilde{G}'_p$	**vertex exp.**	p. 57 4.12	$\tilde{G}_t, \phi_{pt}, \phi_p$	13
H_p	integral	p. 44 3.49	A_p, β_e, l_e	13..19
H_t	triangle tensor	p. 40	p, n_p	14
$\tilde{H}_t$	**vertex exp.**	p. 57 4.6 and P5	p, n_p, n_t	15
G_t	triangle tensor	p. 40	p, n_p	17
G_t	vol. expans.	p. 38 3.42	ω_t, A_t	18
H_t	**experimental**	p. 59 4.16	A_t, l_e, n_t, n_p	19
G_p	angle deficit	p. 43 3.44	A_p, ϕ_p, ϕ_{tp}	20..26
$\tilde{G}_p$	**vertex exp. (*)**	p. 30 3.18	$\tilde{G}_t, \phi_{pt}, \phi_p$	23
G_p	vol. exp. (*)	p. 30 3.18	G_t, ϕ_{pt}, ϕ_p	25
H_p	integral form.	(Taubin, 1995b)	p, n_p	38
G_p	integral form.	(Taubin, 1995b)	p, n_p	38
$\tilde{H}'_p$	**vertex exp.**	p. 57 4.11	$\tilde{H}_t, \mathbf{N}_p$	31
$\tilde{H}_p$	**vertex exp. (*)**	p. 30 3.18	$\tilde{H}_t, \phi_{pt}, \phi_p$	31

Chapter 5

Dynamic analysis by video metrology

Video analysis is an essential part of the swimming coaching. The main benefit is the quick and accurate feedback, which helps the athlete to reflect the feedback in the light of his still recent kinesthetic memory. It is also possible to have metrics not only from the performance (speed) but about the swimming technique as well. The silhouette vectorization and capturing of the biomechanics are naturally easier, too. Silhouette vector and biomechanics signal e.g. a subset of the body skeleton joint angles (Nguyen et al., 2016) are a very useful input for ML methods. A concise utilization of video signal to measure several geometric aspects at the same time is called video metrology (Criminisi, 1999), and the following is an attempt to expand the work of Criminisi with sparse underwater camera array.

(R2) explores possibilities of the direct projection method in swimming analysis and (R4) compares the results with the traditional camera calibration. Both approaches (camera calibration and direct projection) are computationally simple, yet rather involved with the geometrical concepts and relations. The calibration requires a theoretically valid optical theory including the air-plexiglass-water interaction supported by the camera model. The direct projection requires more carefully orchestrated calibration measurements.

The camera calibration documented in (R2) and (R4) seems a necessary step to get good biomechanical signals for further attempts to cluster athletes. Also, there is a natural need to measure and observe athletic performance (motivation 1 for a geometric modeling presented in Sec. 1).

Camera position and orientation errors translate to the pixel errors rather directly and are intimately related to the general arrangement of the site. A motivation for the direct projection method presented in papers (R2) and (R4) is given in Sec. 5.2. The general layout of the measurement site and the swimming coaching sessions is presented schematically in 2D in Sec. 5.4.1 and in Fig. 5.4. Sec. 5.4.3 evaluates the camera installation accuracy in the stereo-camera approach. The approach used in the publications (R2) and (R4) is reported in Sections 5.5 and 5.6. Nomenclature of this Chapter has been provided separatedly in the Sec. 5.1 for the reader's convenience.

5.1 Nomenclature for the swimming analytics

D,DM,DS	direct projection method with a free interpolant, or with mono or stereo camera model as the interpolant
M,S,I	mono and stereo camera calibration model, ideal camera model with no projection error
RBF	radial basis functions
S, S_{12}	stereo-calibration model, a model created by cameras 1 and 2
α_c	camera model parameters of a camera c
$\Delta.$	standard deviation s.t.d. of an entity
γ	rotation matrix exponent
a_c	camera orientation pixel
c	camera index
e_1, e_2, e_3	unit vector basis of the global coordinates
$e_{ci},\ i = 1, 2, 3$	unit vector basis of the camera c coordinates
F_t	a smooth mapping from pixels p to projection plane locations g
$g \in\ G$	the projection plane point
g_c	camera location
$G \subset \mathbb{R}^3$	the projection plane
$\mathcal{I}_c = (P_c, I_c)$	an image of the camera c
H	Householder projection matrix
I_c	intensity values of an image, can be understood as a function: $P \rightarrow$ pixel intensities
k	refraction index
$n_c, n_c(p)$	ray direction specific to a pixel p
$p \in P$	a pixel
$(p, g) \in\ U$	a ground truth measurement
P_c	a pixel set of a camera c (not necessarily rectangular)
$\mathbf{R}$	a rotation matrix
$\mathcal{R}(p)$	a pixel ray
U	ground truth measurement set
$w(p)$	image blending weight
x, y, z	the global coordinates

5.2 Motivation

The original motivation was to capture an approximative 3D dynamics of the swimmer using a sparse camera array. This faces several challenges, only some of which have been addressed in this Thesis (marked with '+') while the rest remain to be a topic of future research (marked with '-'). The first challenge is the accurate capture of the swimmer silhouette (+), speed (+) and the planar skeletal biometrics (-). The second is reducing the deterioration of the signal from bubbles (-), and proper correspondence of two views to from a 3D biometrics model (-). Only some of the goals were reached, and thus the presentation is to chart a starting point for the future research.

The proposed methodology extends the work of (Criminisi, 1999) to underwater conditions using the line segment-based calibration of (Zhang et al., 2016) to get accurate calibration input data. The paper (R4) also presents a coherent methodology to estimate the accuracy of the existing systems.

The traditional approach with rigidly set multi-camera systems can be presented in two steps. One first creates a pinhole camera model, which maps individual pixels to rays in the local camera coordinates. The next step is to apply a coordinate transformation from camera specific local coordinates to the global ones. The installation tolerance of the cameras naturally has to be of the same or smaller magnitude as the characteristic length of the object covered by a pixel. The installation tolerance requirement is especially high for the orientation of the camera (Dang et al., 2009), as discussed in Sec. 5.4.3.

Publications R2 and R4 demonstrate, that the the direct camera calibration can have superior accuracy with a rather robust implementation. The only economical methods for swimming analysis based on sparse camera array are the direct projection method presented here, and a multitude of auto-calibration methods e.g. Vupparaboina et al. (2015), since these require no accurate camera placement. The auto-calibration faces difficulties in an underwater scenery, since the corresponding visual details cannot be paired well from the blurred view. Also, the only central object is the human body with a non-rigid shape.

5.3 Preliminaries

5.3.1 Underwater optics

Underwater optical observations (Buiteveld et al., 1994) deal with many difficulties, first three (refraction, dispersion and absorption) are systematical and can be countered rather efficiently. The last two (dispersion and vignetting) are noise signals proportional to the distance and hard to eliminate. The vignetting occurs only at the borders of the image.

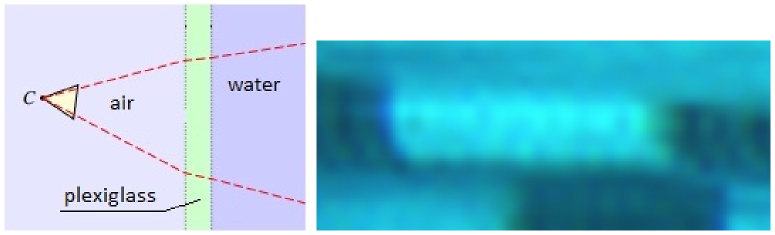


Figure 5.1: *Left*: Refraction at the water-plexiglass-air system. The camera is not exactly perpendicular to the glass, and the plexiglass properties, especially its refraction factor, are not well defined. *Right*: A detail of a swimming lane buoy (approx 45 cm long at a distance of 9 m). Dispersion colors large intensity gradients. The effect is strongest at the border of the camera view.

To start from **refraction**: there is a change of the direction of the light rays at surfaces where two mediums with different speed of light meet. E.g. the ratio of speed of light in air and water is approx. $k = 1.4$ (the refraction index). This phenomenon described by the Snell's law can be combatted by including the refraction to the camera models, or using spherical plexiglass casings in the front of the camera view. **Dispersion** is a related phenomenon: the speed of light in water (and in lesser extend in air!) depends on the wavelength. This means sharp contrasts in images get blurred with a rainbow coloring, see Fig. 5.1. This phenomenon can be countered e.g. by calibrating the camera three times using different wavelengths, and then using the camera models to produce a compound image. Water is optically rather dense material and its **absorption** of different wavelengths varies: 400 nm wavelengths are absorpted the least, and 980 nm (actually already at the invisible range) wavelengths the most. There are several methods to counter this loss of color for the purpose of more aesthetical images. The last phenomenon, **scattering**, is the most difficult to deal with. Scattering deviates a number of light beams and brings noise to the image, and there is an ultimate distance limit for the video analysis with and without a dense camera array. There is still one phenomenon, **vignetting** (**?**), which is the loss of brightness at the border of the view. Vignetting is caused by the plexiglass properties at large ray angles, since the camera view is close to 140^o.

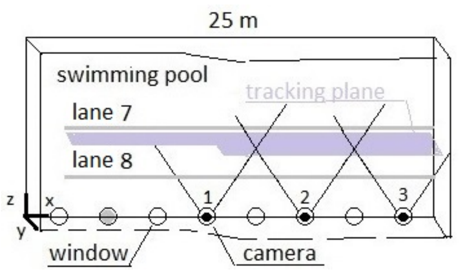
5.3.2 Current research

An excellent presentation of the optics of the water-plexi-air system can be found in (Bäuer-Burchardt et al., 2016) and a more directly applicative approach (the Pinax model) in (Luczynski et al., 2017). It is becoming more common to see the camera metrology problem as a continuum starting from optical models to defining a smooth enough direct mapping between world points and the pixels, as in (R2,R4) and (Criminisi, 1999). The latter is for the dry setting and consider many minimally measured settings, which differ from this work, where only one swimming plane is used. This work (R2,R4) is dealing also with the correspondence problem (Scharstein and Szeliski, 2002) at two different instances. First one is in detecting the chessboard pattern of the calibration board, which easily induce errors. These practical problems were solved by various means, e.g. matching lines (Zhang et al., 2016) instead of crossings. The second is about to match the correspondent anchor points (Lu et al., 2017) in the swimmer body, and this is still an open research problem.

5.3.3 The swimming site

Characteristic to Impivaara swimming site in Turku is that athletics analysis happens in a pool which is used by the general audience at the same time. This sets special requirements: e.g. the video cameras have to be only on one side of the pool and safely behind plexiglass windows. A short pool (25 m) was chosen for technique practise of start, glide and turn phases. The number of cameras (3) was limited by the budget and the camera positions were dictated by the architecture of the pool, see Fig. 5.2.

The swimming lane 7 was dedicated to the athletes and the lanes 8-10 were kept empty during the swimming analysis sessions. Distance between the cameras and the center of the lane 7 was 8.75 m. A camera view covers approx. 9.3 m.



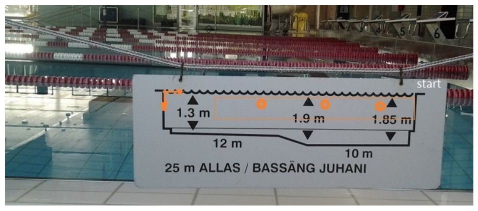


Figure 5.2: *Left*: The swimming pool and the projection plane at the middle of the swimming lane 7. *Right*: The depth profile of the pool. Camera windows marked with bullets and the final virtual view with a rectangle.

5.4 Camera Calibration

Camera calibration means creating a map from pixels to pixel rays emanating from the camera location. Geometric rectification means that a camera view is transformed onto a common coordinate system, in this study to a swimming plane G. Four models are used for the latter. These are mono-camera and stereo-camera calibration models (M and S), an ideal camera model (I) with only the placement and orientation errors, and the direct projection method (D). The model I is used for assessing the impact of the camera installation accuracy to the overall geometric mapping from pixels to global coordinates.

Traditionally, the video analysis of swimming has been done by an array of calibrated cameras with a very accurate camera casing. If the camera array is dense, an S or multi-camera calibration (Gai et al., 2016) is the choice. The multi-camera approach has been left out from this study.

Each camera in a dense array have overlapping views with more than neighboring two cameras whereas a sparse camera array has at most two neighboring views overlapping (Vupparaboina et al., 2015). If the camera array is sparse as in our case depicted in Fig. 5.2), the M approach is still possible. These methods require a calibration checkerboard or similar source of the ground truth information (see Fig. 5.7) to be recorded in the calibration volumes of cameras c and c' depicted by grey circles M_c and $S_{cc'}$ in the left part of the Fig. 5.3.

There is a practical limit in the depth dimension z of the underwater camera view restricting the stereo calibration, making it both an inaccurate and difficult procedure in case of the sparse camera array. The M method has more calibration volume to be used, but it sets heavier demands on the camera placement. Methods M and S are limited to the calibration sampling being limited vertically between the surface and the bottom, too. M and S calibration are introduced more thoroughly in the next Section. The direct projection D is being presented in Sec. 5.5.

5.4.1 Mono and stereo calibration

The left part of the Fig. 5.3 depicts the stereo view volume for the stereo-camera model S_{12} of cameras 1 and 2, and the calibration volume of the mono-camera model M_1 of the camera 1. The mono-camera calibration happens for each camera separatedly by recording the calibration board with a chessboard pattern underwater in the front of the camera. The calibration process (?) fixes the camera model parameters $\alpha_c \pm \Delta\alpha_c$, where the first argument is the vector of actual parameters (5 to 9 of them depending on the chosen camera model complexity) and the second is an estimated s.t.d. The remaining part is to specify the camera location $g_c \pm \Delta g_c \in \mathbb{R}^3$ and

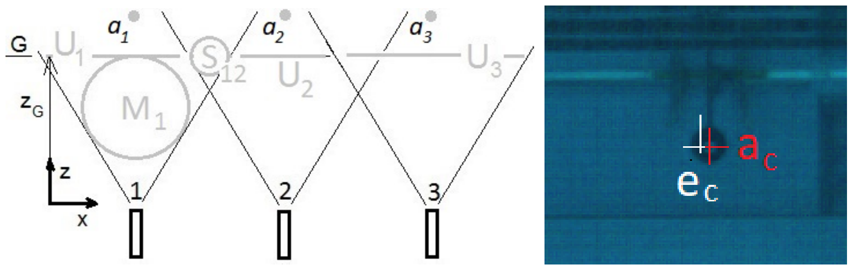


Figure 5.3: *Left*: The calibration volume of mono calibration of the camera 1 (M_1) and stereo volume of cameras 1 and 2 (S_{12}). The ground truth measurement sets U_1,U_2 and U_3 of the geometric projection method are also depicted. *Right:* The (red) orientation pixel a_c marks the global e_3 axis (along z). The pixel e_c marks the optical axis of the camera. A view from the camera $c = 3$.

its accuracy (std.) in the global co-ordinates before the pixel ray $\mathcal{R}(p) = \{tn_c + g_c \,|\, t \in \mathbb{R}_+\} \subset \mathbb{R}^3$ with a normal vector $n_c(p) \pm \Delta n_c$ specific to a pixel p can be computed.

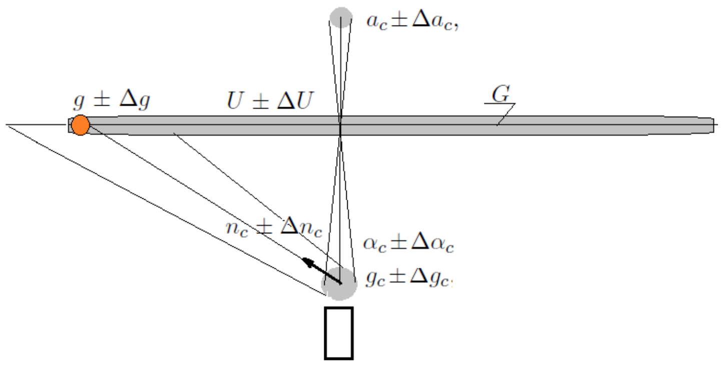


Figure 5.4: A schematics in 2D showing the camera orientation marker error Δa_c, the camera location error Δg_c, the camera model error $\Delta \alpha_c$ (a vector of 5 to 9 terms depending on the complexity of the calibration model), the pixel ray error Δn_c, the marker errors ΔU and the resulting location error Δg on the projection plane G.

The stereo-camera calibration is very much like the mono-camera case except that the measurement observations in the common view volume S_{12} depicted in the Fig. 5.3 are addressed differently; they produce common error from the both camera models. The parameters subject to the optimization of the total error are twice as many as in the mono-camera calibration, while the common view introduces a compatibility constraint. Therefore, the stereo calibration requires some more samples than two mono-camera calibrations.

One can now define a mapping $F_t : p \rightarrow g$ from pixels to a suitably chosen projection plane G by calculating the projection point g:

$$\{g\} = \mathcal{R}(p) \cap G\,. \tag{5.1}$$

See Fig. 5.4, which has an overall depiction of the camera position and the mapping of pixels p to projection points g.

5.4.2 Camera position error evaluation, S method

Mono-camera calibration requires a special additional observations to orient the camera. In Impivaara case, the additional observations were the global axis pixels $a_c, c = 1..3$ geometrically measured to be exact (with a sub-pixel accuracy), and the corresponding optical axis pixels e_c, see the right part of the Fig. 5.3. For stereo-camera calibration, it is also possible to chain the models S_{12} and S_{23} by forming a best square sum fit between them without actually using the orientation pixels. This reveals the poor quality of the measurements. The cameras should roll out in a relatively straight line, but the calibration data is simply not accurate enough due to too small overlapping camera views and lack of a more sophisticated camera model. The quality test by chaining the models produces a camera configuration which is approx. 4 cm erroneous, the rest of the misalignment shown in the right part of Fig. 5.5 is explained by the actual physical realization of the camera setup. The M and S calibrations were done by Matlab toolbox (**?**), which is based on the works of **?** and **?**. Probably the best currently available water-plexiglass-air camera model of Luczynski et al. (2017) was not available at the moment of the research reported in the papers (R2) and (R4).

5.4.3 Assembly costs of the camera array

Usually, the camera model is attained while not referring to the position and orientation at all. This leaves the mechanical fixing of the cameras likely with the sub-pixel accuracy i.e. a negligible level. In the setting of the swimming lane 7 of the pool depicted in Fig. 5.2, it means $4\,\mathrm{mm}/5\,\mathrm{m} = 0.05^o$ orientation accuracy for a pixel size 4 mm (the refraction index $k = 1.4$ taken into account in the distance 5.0 m). If the camera coupling surface is 50 mm long, this means the assembly accuracy of $4\,\mathrm{mm} \times 50\,\mathrm{mm}/5\,\mathrm{m} = 0.04\,\mathrm{mm}$.

It is generally known and very universal engineering law, that machining costs of high-precision mechanical parts grow faster than exponentially, when the geometric tolerance is improved (Colding, 1961), e.g. when the positioning and orientation accuracy of a camera rig are increased, see Fig. 5.6. Modern production methods have not been able to challenge this law. A similar law holds for positioning and orientation of the assembly of machinery components. Achieving tight tolerances is rather costly, especially if the

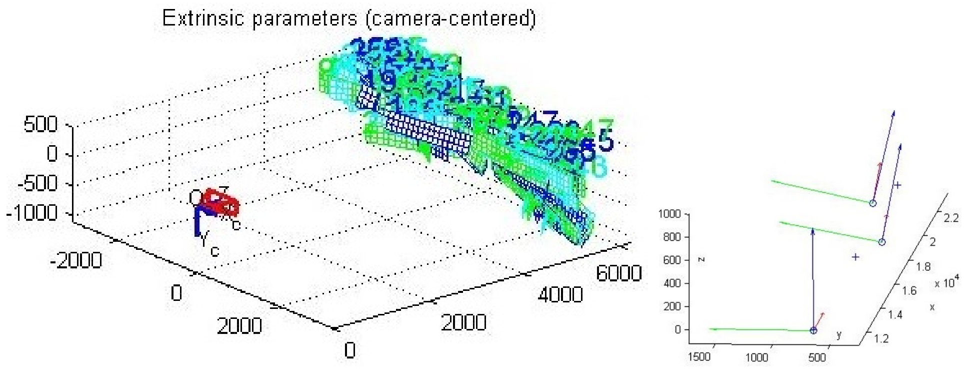


Figure 5.5: *Left*: The calibration chessboard pattern in the camera 2 specific co-ordinates. Notice the vertically limited positions. *Right*: Local coordinate frames of the cameras $c \in \{1,2,3\}$ (from left to right) after minimizing the square error between the measurements of models S_{12} and S_{23} by keeping camera positions g_c as the minimization argument.

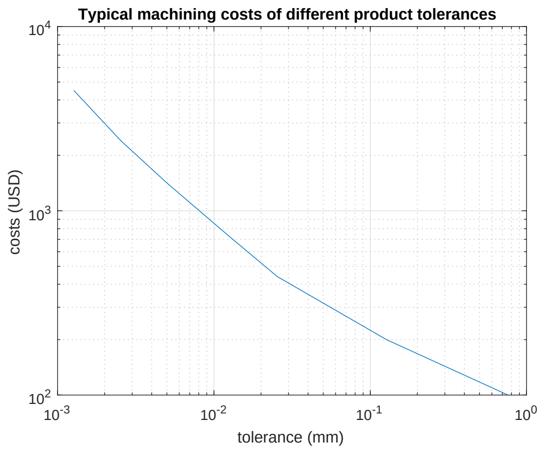


Figure 5.6: Typical costs of a small machining object with a limited production series. A reproduction of an image of Colding (1961).

question is about a row of several cameras, and if the cameras should be easily removable (a common requirement).

5.5 Direct projection method (D)

The direct geometric projection is a historical alternative from 1920s (Mündermann et al., 2006), when camera models were not computationally possible yet. A projection plane dictated by the center of the swimming lane is calibrated directly, i.e. the pixel mapping sampled by a sparse measurement set U (see grey line at the Fig. 5.2 and the point set at the Fig. 5.7) is generalized to whole of the projection plane G. The idea is to avoid the demand for accurate camera positioning. Cameras still need positioning in the practical sense in order to cover the view adequately and to keep them steady, and only a visual and manual assessment of the positioning is enough.

5.5.1 Practical arrangements

Fig. 5.7 depicts the practical details and the workflow of the direct projection method (D). Two pontoons support a chessboard, position of which is measured by a laser meter along the swimming lane. The chessboard is guided by a rope, which gives the estimated positioning accuracy of 1.5 cm std. Cameras are held by a simple frame, allowing manual aiming of the optical axis is aimed towards a marker at the opposite wall. The cumulated measurements $(p, g) \in U$ form two point clouds, one of observed pixels $p \in P$ (right top corner) of the pixel frame P and the other of the observed positions $g \in G$ (the detail below the previous one). The interpolation generalizes to a mapping $F_t : \; p \to g$ for all pixels $p \in P$ in a given camera view while minimizing the global error (third detail in the right). The resulting rectified image (below right) is produced by a fast version of F_t assigning 4×4 mm space for each pixel. The spatial size of the final virtual view is a deliberate choice: it is round figure conveniently close to the average spatial size of the pixels of each camera view.

Fig. 5.7 shows the measured locations $U \subset G$ on the projection plane G. Each measurement (a pair $(p, g) \in U$ of a pixel p and its location $g = (x, y, z) \in G$ on the swimming plane allowing the interpolation $\tilde{g} = f(p)$, $p \in P$. The pixels are corners of the chess board pattern (see Fig. 5.7). The swimming plane has an approximately constant $z \pm \Delta z$ with a s.t.d. $\Delta z = 0.015$ m in the global coordinates. The measurements were rather dense, approx. 9000 measurements over the area of $1.5 \times 17\,\mathrm{m}^2$ resulting in 5 cm average distance between the measured points. Even a local linear interpolation would have worked, but the method of **?** was used. It consists of local thin plate spline fitting, see details in (R2).

Several other interpolation methods are possible, e.g. using a small collection (3 per each camera view) of the local mappings governed by mono-camera models each with 6+3+2+2=13 parameters for the actual camera model, position and orientation, respectively. This approach actually has less parameters ($3 \times 13 = 39$) than the approach used in (R2) (approx. 90 to 100). Using arguably the best available air-plexiglass-water camera model of Luczynski et al. (2017), one ends to $3 \times 12 = 36$ parameters while gaining almost global physically correct interpolant.

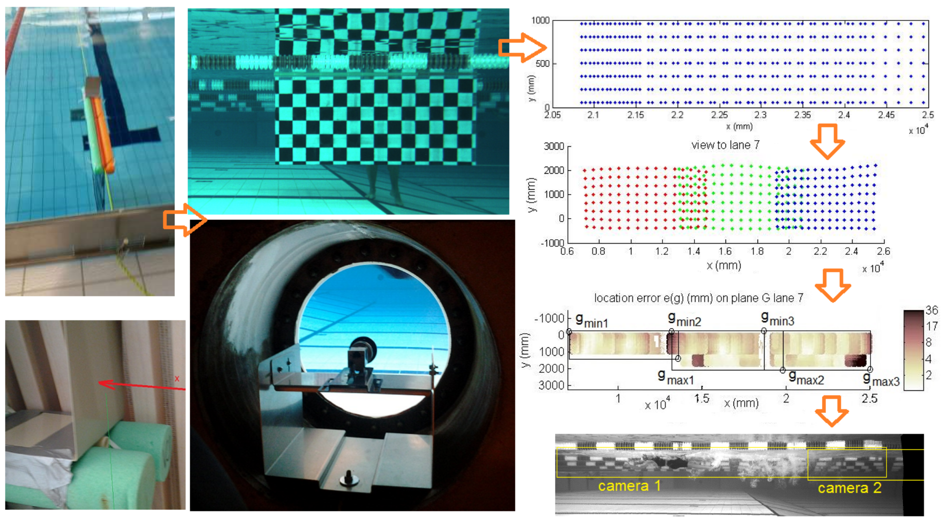


Figure 5.7: A set of images depicting the direct geometric calibration, which also serves as a quality check for traditional camera calibration methods. A floating calibration board floats guided by a wire through the swimming lane. Laser distance measurement (*bottom left*) makes it possible to generate the calibration data U (top right). *Bottom right*: The validation and the resulting virtual view.

5.6 Accuracy summary

The accuracy of the methods M and S can be compared to D by assigning the best possible camera positioning to each camera and minimizing the global error that way. This is actually favoring those methods, since the basic M and S do not use the information derived by measurements U.

Recall the Fig. 5.4. At this stage, the projection plane G is useful to only make the accuracy comparisons sensible, i.e. the error of the methods M and S is directly about the swimmer measurements (and speeds, after a simple additional calculation). The s.t.d. Δg of g is caused by s.t.d.'s Δg_c, Δn_c and measurement uncertainty ΔU of the corresponding quantities g, n_c and U. The orientation error Δn_c inherits its inaccuracy from the following: $\Delta \alpha_c$ (camera model s.t.d. from the camera calibration software), Δg_c (placement s.t.d.) and Δa (orientation pixel s.t.d). The actual definitions and calculations involved are relatively basic and documented in (R4), although e.g. Δn_c has a von Mises distribution (**?**) on a unit sphere instead of a Gaussian distribution. The handling of the rotation matrix is somewhat special, and due to some typographical errors in (R4), is covered briefly in Sec. 5.6.1.

5.6.1 Rotation

To properly compare the accuracy of M and S methods with direct methods, one needs to let the orientation and location errors of cameras contribute properly to the final projection error Δg. A symbolic algebra package Sympy (Rocklin and Terrel, 2012) supporting random variables was used to compute proper error distributions.

After a pixel p has been registered with an intensity $I_c(p)$ by a camera c, it is time to find a corresponding orientation within the camera specific co-ordinates, which are more or less unaligned with the global co-ordinates $E = \{e_1, e_2, e_3\}$. The camera specific co-ordinates have an optical axis directed (almost) towards a special marker $a_c \in P_c$. The camera rotation around the optical axis has been aligned to the horizontal level with a sub-pixel accuracy already. The camera specific co-ordinates $N_c = \{n_{c1}, n_{c2}, n_{c3}\}$ are dictated completely by the observation of the alignment marker at the pixel a_c. The rotation matrix $\mathbf{R}$ is a linear map which helps to transform the camera model pixel beam $n_c(p)$ to the global beam $n(p)$: $n(p) = \mathbf{R}^{-1}n_c(p)$. The global beam is then used to find out the final projection point.

This treatise acts as a clarification of (R4), since (P4 p. 10) has unfortunate typographic errors swapping some matrices $\mathbf{H}_{..}$ (Householder) and $\mathbf{R}_{..}$ (rotation).

A 3D rotation can be constructed from two line reflections, (Dorst et al., 2007) which is the best formulation for the analysis of inaccuracies, since it allows an easy derivation of the Taylor series and is a viable way to apply the distribution algebra. The geometric algebraic approach and various formulations of the projective geometry can surely give attractive alternatives, but they have been excluded from this thesis. The necessary details can be found from (Dorst et al., 2007). Fig. 5.6 depicts a point p rotated around a point O. An individual line reflection dictated by a line $span\{u\}$ is denoted as $p' = H_u p$, where H_u stands for the Householder projection and equals:

$$H_u = -I + 2u^0 u^{0T}, \tag{5.2}$$

and I is a 3D identity matrix. The construction in Fig. 5.6 has $p'' = H_{u+v} H_u \, p$, where $H_.$ stands for a Householder projection matrix. The above arrangement can be directly utilized as the actual definition of the rotation matrix R_{uv}:

$$R_{uv} = H_{u+v} H_u \, . \tag{5.3}$$

It can be seen by the direct substitution that the rotation fulfills the following

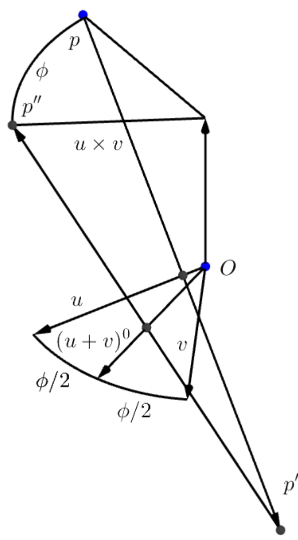


Figure 5.8: A point p rotated to p'' around the line $span\{u \times v\}$ by an amount of ϕ using two sequential line projections through a line $span\{u\}$ and through a line $span\{u + v\}$.

four requirements for the unit vectors u and v:

$$R_{uu} = I \text{ (identity)} \tag{5.4}$$

$$R_{uv}u = v \text{ (rotation from } u \text{ to } v) \tag{5.5}$$

$$R_{uv}w = w \text{ (eigenvector property)} \tag{5.6}$$

$$R^2_{u(u+v)} = R_{uv}\,, \text{(exponentiation)} \tag{5.7}$$

where a vector w has $w \cdot u = w \cdot v = 0$ and is thus perpendicular to the rotation plane $span\{u, v\}$. The last requirement extends (by a construction of a sequence of a binary choices converging to a chosen rational number) to a more familiar exponentiation property of rotation angles: Let R_{uv} have a rotation angle ϕ. Then R^γ_{uv}, $\gamma \in \mathbb{R}$ has a rotation angle $\gamma\phi$.

The actual projection of perceived pixels p to the projection plane can be later sped up by recording the map $p \to g$ directly, as explained in (R2).

5.6.2 Direct methods with camera models as interpolants

The best possible camera model with detailed air-plexiglass-water refraction (Luczynski et al., 2017) should indeed provide the best interpolation function especially what comes to the extrapolation outside the measured U zone, although ordinary regularized interpolants perform rather well at the practical zone nearby U. The (Luczynski et al., 2017) model was not available at the time of the study, though.

Multi-view arrangement is a modern approach (Furukawa and Ponce, 2009), which fashions a very neat theory and adapts to a multitude of camera models. It was not usable in this case due to the architectural constraints of the site and the budget limitations.

It is possible to use any smooth interpolant with the D method, e.g. (Luczynski et al., 2017) or the camera models of M and S. The two latter choices lead to DM and DS methods. DM and DS are very close to interpolation with radial basis functions (RBF) with very large radii.

All these approaches M,S,D,DM and DS produce a mapping $F_{model} : P \to G$ from pixels $p \in P$ to points $g \in G \subset \mathbb{R}^3$ on the projection plane G, see a summary in Table 5.1. Included is an ideal (I) camera model to demonstrate the effect of the camera location and position error. The model I is forced to be absolutely correct by defining the pixel ray n_c to be exact:

$$n_c = (g - g_c)^0, \tag{5.8}$$

whereas all the other camera models produce some error Δn_c. The camera calibration tool used (**?**) formulates both the pixel ray $n_c(p)$ and the corresponding error $\Delta n_c(p)$ with a best Gaussian distribution approximation.

5.6.3 Comparison of the methods

The models M and S use the orientation information provided by the alignment pixels a_c, while methods DM and DS use the best possible fit i.e. the camera orientation a_c and the location g_c were interpolation parameters alongside the usual camera model parameters α_c. Table 5.1 summarizes the mean error and characterizes each method by their parameterization. The mean error is the difference $\text{mean}_{(p,g)\in U} \|\hat{g}(p) - g(p)\|$ of the predicted position $\hat{g}(p)$ and the actual position $g(p)$ over the calibration set $(p, g) \in U$. the model DS has a rather weak accuracy and it has been excluded from the treatise. 'I' stands for the ideal (non-realisable) camera model, which has only the camera location and position as parameters. The last five columns are: mean pixel projection error at the projection plane G, number of the cameras involved needing a pair of orientation and placement parameters a_c, g_c, number of the camera model parameters α_c, whether the calibration point set U is used or not, and the total number of parameters d for the final virtual view, respectively. d is counted over all the cameras, e.g. I has $d = 3 \times (2 + 3) = 15$. The number of parameters d of direct method D may seem high, but the interpolation is heavily regularized avoiding overfitting.

Figs. 5.9 and 5.10 provide an overview of the projection error. It is important to have a suitable stitching of the overlapping projections without too large systematical deviations. Methods DM and D both deliver a rather good projection accuracy. E.g. the D method has the speed error within $\Delta v \leq 0.07 m/s$ for an athlete speed $v = 1.6\,\text{m/s}$, which is $\pm 4.5\%$. Both M

Table 5.1: Parameterization summary. The spatial reference is either the marker pixel a or the markers of the projection plane G. Camera model is either used or not. Parameters not needed are marked with '_'.

Model	Spatial ref.	Cam. model	Error (mm)	a_c, g_c	α_c	U	d
M	a	yes	18	1	1	no	33
S	a	yes	37	2	2	no	44
D	G	no	4.0	_	_	yes	-
DM	a	yes	8.8	1	1	yes	33
DS	a	yes	40	2	2	yes	44
I	a	no	11	1	_	no	15

and S methods show large error at the perimeter. This is due the inadequate shape of the calibration volume (too shallow), large camera placement error (approx. 1 cm) and inadequate camera model for large angular view (air zone has the view angle close to 140^o). S method suffers from the sparse array placement, too.

Another method to validate the camera calibration is the commonly used back-projection (Zhang, 1999) in Fig. 5.11, which usually shows the error of the projection in pixels, but is modified here to show the projection error on the projection plane G. The camera $c = 1$ is at the shallow end of the swimming pool (see Figs. 5.2 and 5.7). This squeezes the volume used for calibration chessboard, and causes a visible loss of accuracy in M and S models. The D method is a clear winner. M method works about as well in the vertical direction and on both directions on camera 3. This camera has a reduced width of view due the proximity of the wall at the starting end of the pool. The angular vertical change is small, and even the inadequately narrow calibration samples cover the vertical direction well. An estimate of the author is that a combination of 3 mono-camera models used in DM approach would reach the performance of the interpolation used in D method.

The D method produces final combined mapping, which can be used for fast geometric rectification.

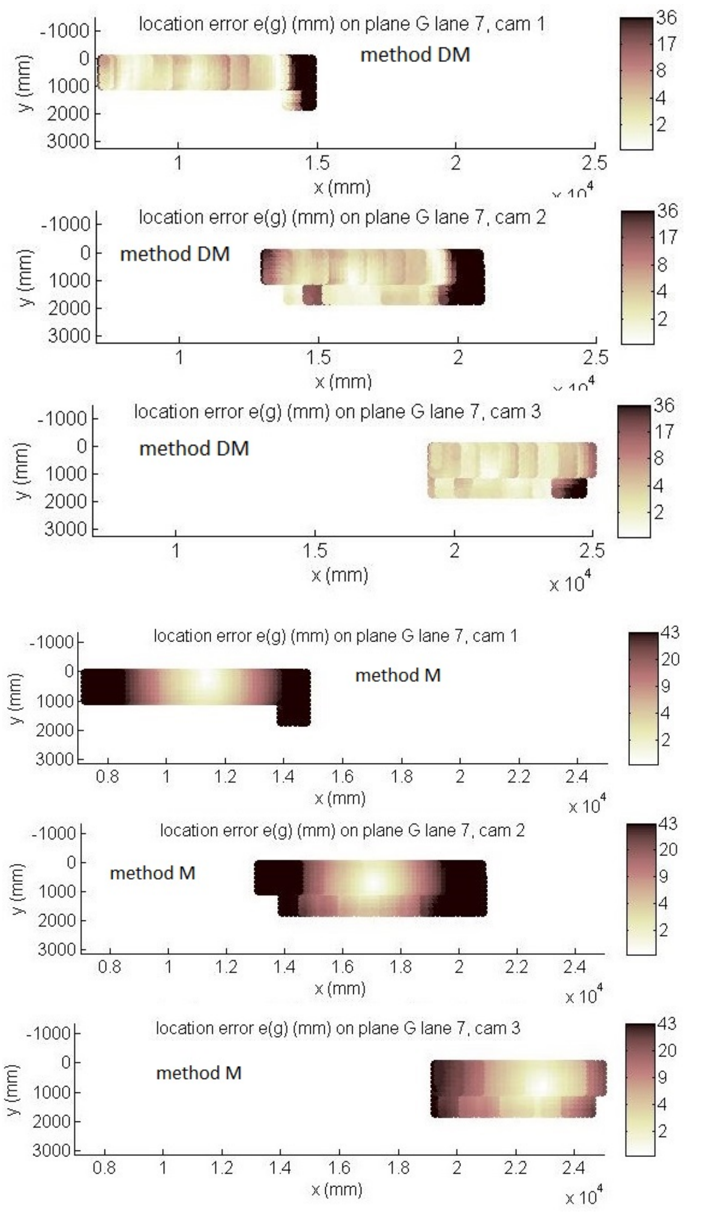


Figure 5.9: The placement accuracy by cameras 1,2 and 3 (top to bottom) by each calibration method. *Top*: direct mono model (DM). *Bottom*: Mono calibration model. (M)

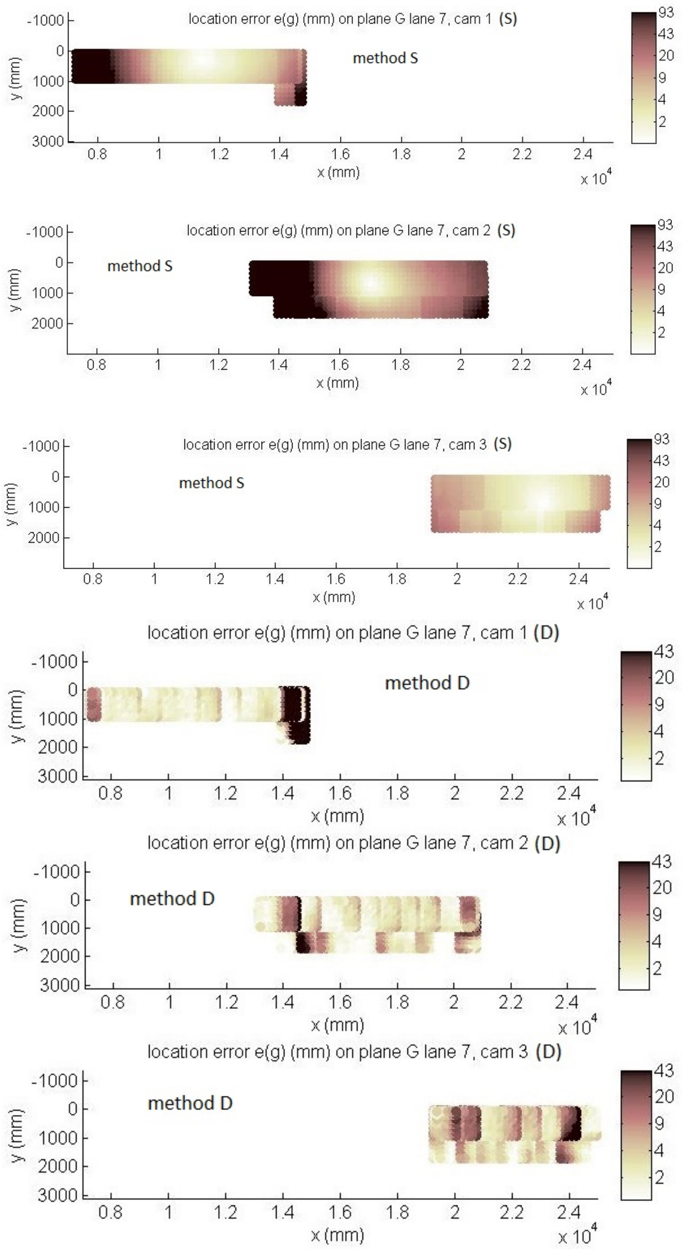


Figure 5.10: From top to bottom: The placement accuracy by cameras 1,2 and 3 of stereo calibration (S) and by cameras 1,2 and 3 of direct calibration (D).

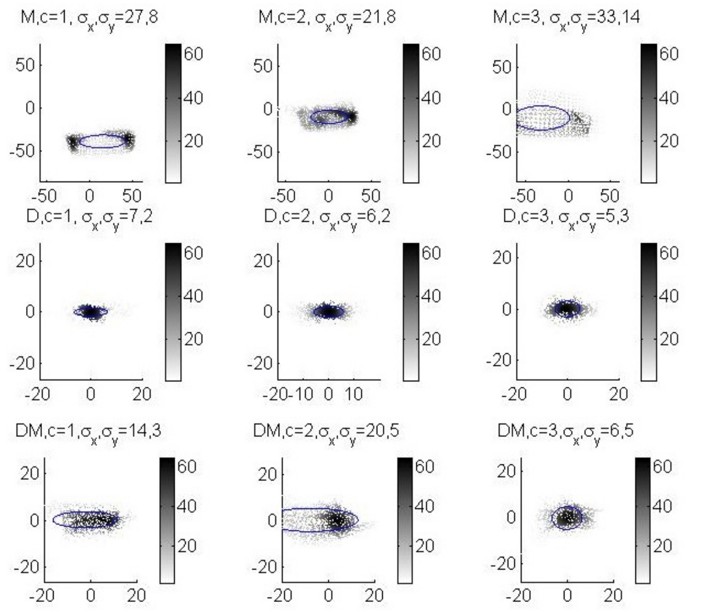


Figure 5.11: Back projection test of various calibration methods. The units are mm's on the projection plane G.

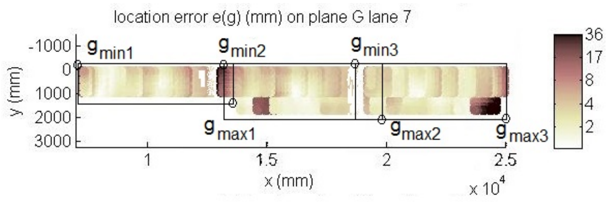


Figure 5.12: The stitching of the three PP mappings.

Chapter 6

Research description

Research questions and a summary of the research are presented in the following two sections.

6.1 Research questions

Research questions are formulated in the following:

Q1 Is the curvature signal usable in natural resource data?

Q2 Do established curvature methods apply to natural resource data?

Q3 Which is the mildest mechanical accuracy requirement still adequate for underwater stereo view analysis in swimming research?

Q1 originated from a more pompous quest over a set of useful features derived from the geographical height signal (of any source, DEM, LiDAR, photogrammetric, satellite radar). This quest quickly focused on Q1, although a brief presentation of various height features has been presented in Sec. 3.2.

To answer Q1, several techniques and approaches were experimented with and three field campaigns were conducted. Adequate predictors were based on curvature on each case, although a lot of improvement would be possible combining several methods together.

The local linear fit, if suitably adapted to converge to the maximum density in the local height distribution, allows both a multi-scale approach (see Section 3.3 at p. 23) and regular grids to be formed from the point cloud. Direct point cloud analysis is very difficult without some sort of filtering to eliminate effects of outgrowth, foliage and canopy, and one promising filtering method was proposed (P5). The directional curvature histograms derived from the triangularization proved to be numerically efficient and applicable to practical problems (P6). This result, when properly integrated with point cloud filtering to achieve a multi-scale capability, seems to be the most promising of the findings.

Q2 was based on the author's suspicion that most of the curvature algorithms are developed for technological environments. Laser scans of small artifacts, architecture and such have good sampling density and relatively low noise in the off-the-plane component. To answer Q2, the author had approximately 6 different attempts to find new noise-tolerant formulations and principles, until one, triangular mean and Gaussian curvature via planar expansion, was found. Recently, two publications were found, which seemingly have not been applied widely to technological environments, and which may be suitable to the natural resource data with minor adaptations, but somewhat less efficiently. These are methods of Theisel et al. (2004) and Taubin (1995b). The Taubin's method facilitates a built-in smoothing, which may be useful with a multi-scale approach. It may be possible to implement the directional curvature histograms as efficiently to the method of Theisel as to ones developed in this work.

Q3 has an intrinsically geometrical nature, too. It led to a demanding analysis of probabilistic representation of rotation tensors. To answer Q3, a projection plane approach was used. This method has been applied in 1910-1930 for biomechanics movement capture (Mündermann et al., 2006), but was then abandoned, when modern mono- and stereo-camera calibration methods were developed. But, if the cameras cannot be positioned nor oriented accurately, these methods render useless. The answer to the question was found, it is possible to analyse swimming with sparse camera array and to have a very small budget to mechanical camera installation. The most important requirement for the stationary camera installation is the steadiness (unmovability over time and environment conditions) of the cameras, but not the orientation accuracy.

6.2 Summary of the articles

(P1): Micro-topography registration from point clouds, ground stoniness as a case

Summary: The title of (P1) (Nevalainen et al., 2015) is *Detecting stony areas based on ground surface curvature distribution*. The sample areas sum up to $3.3\,km^2$ but are distributed over a rectangle of $32 \times 36\,km^2$. LiDAR data of sample density $\rho = 0.8\,m^{-2}$ was used to detect stony areas labeled to positive and negative areas by a geology expert. The motivation of the study was twofold: to try to reach close to the theoretical object detection limit, and to verify if the method is applicable for commercial usage.

Method: The curvature was approximated by choosing a set of regular raster points, each of which had a local planar fit to approximate the tangential of the ground surface. The distance weight was an ad hoc function of Appendix I to ensure, that the occasional vegetation points did not deviate the plane too high above the ground. Each plane defined the grid normal, which was then used to approximate Gaussian curvature by barycentric interpolation given in Meyer et al. (2003). A peculiar arrangement was used to estimate the vertex Gaussian curvature G_p: the barycentric formula of Eq. 8 gives the curvatures G_{tp} at each adjoined triangle $T(p)$, then the mode of the curvatures is addressed: $G_p = mode_{t \in\ T(p)}(\{G_{tp}\})$.

Since the multi-scale approach with grid square sizes δ ranging between $\delta = 1.25 \leq \delta \leq 6\,m$, the missing information had a significant effect at the densest two grid. Even the local fits performed at regular grid points arise a chance to use bilinear curvature formulations, the barycentric triangular version was used, since this reduced the effect of missing information, see (P1) Fig. 2. The Gaussian curvature spectrum was cumulated over each sample area, and this was used as a vector source for logistic regression. **Output and observations**: The results were adequate for practical purposes. The paper entered to a previously uncharted area of accuracy in micro-topography detection.

Method parameters were numerous, although a large subset of parameters seemed to have no effect at all to the cross-validated prediction performance. This called for more research to find curvature extraction methods with less parameters and more predictable behaviour. This paper contributed mostly to Q2 in an indirect way, since most of the tested traditional curvature methods did not perform well.

Personal contribution: the algorithm development, analysis of the data, drafting of the manuscript. Geological expertise was provided by PhD Maarit Middleton from Geological Survey of Finland (GTK). The data was gathered by prof. Raimo Sutinen (GTK) and PhD Maarit Middleton.

(R2): Video calibration without a camera model: A sparse camera array as a case

Summary: The paper (Nevalainen et al., 2016a) titled *Video based swimming analysis for fast feedback* addressed a situation where small budget forced the swimming analysis to be based on only few sparse cameras. No adequate and accurate methods existed, and the project had a very limited funding. The reliability and simplicity were the leading design principles of the system. The camera calibration is well established as long as the camera positioning and orientation is known rather accurately. At the moment of writing, accurate pin-hole camera models for flat-pane housing with front glass correction in underwater conditions existed. (A recent reference is Luczynski et al. (2017), which has an adequate model, but no correction for possible axial orientation error.) Also, the mechanical accuracy required in non-immersive conditions was too expensive to arrange, though.

The aim of the paper was to measure the swimmer speed and to enable further bio-mechanical analysis over the swimmer profile. A specific notation is used for images $I = (P, J)$: they are formulated to consist of a set of pixels $P \subset [1, n] \times [1, m] \times \{1\}$, where n, m is the size of the camera output images in pixels, and the intensity value map (a function) $J : P \to [0, 255] \subset \mathbb{N}$. This allows identifying images like $(P_1 \cap P_2, J_3)$, where the domain is not necessarily rectangular, and several possible intensity maps J_3 can be applied. The direct plane calibration is actually a very old method, which was used in 1920s and 1930s with the physical scaling, e.g. a chessboard pattern behind a galloping horse was scaled in size to correspond to correct measures of the projection plane (also the camera was removed as far as possible to reduce parallax errors).

Method: A direct calibration method was used, where the actual planar positions of the swimming lane are measured in the absolute reference frame, and related to the pixel positions by non-linear interpolation. The computational arrangement enables a quick referencing of only one nearest neighbor, or approximately four time slower referencing of 4 nearest neighbors for Shepard interpolation.

Output and observations: It was proven that the first option is simple enough for real-time video signal adjustment with the 50 Hz frame rate. The overall system arrangement was very economical. Its weak point is the manual file management and lack of proper production tools for the swimming coaching sessions. The paper contributed to the research question Q3 by demonstrating in practice that the camera positions need not to be known.

Personal contribution: Design and practical implementation of the camera setup and calibration. The algorithm development, analysis of the data, drafting of the manuscript. The video material was gathered by MSc Antti Kauhanen, Turku Sport Academy (TSA).

(P3): Micro-topography registration from point clouds, extension of (P1)

Summary: The article Nevalainen et al. (2016b) has a title: *Detecting terrain stoniness from airborne laser scanning data*. Two new methods are introduced, one of which uses the traditional data format (DEM, the regular raster data). There is a new data set, which is considerably larger, and the focus is on characterizing each sample by a curvature distribution vector, just like the previous paper did.

Method: Curvature profiling based on triangularization (LTC) was being compared with the local linear fit of (P1), and a traditional DEM analysis using a Laplace operator to register stoniness. LTC needs a proper ground

model, which was produced by the SAF process, but not all the parameters were subjected to the cross-validation tuning. The logistic regression was used just like in (P1).
Output and observations: The equation (Eq. 8, P1) for Gaussian curvature was changed to (Eq. 5, P3) creating much less noise to the cumulated curvature histograms. LLC curvature results (both the physical scale of curvature, and the prediction performance) were verified by a totally different method using the Laplace operator based curvature estimation on regular DEM grid. The TIN -based method performed worst. It became apparent, that Gaussian curvature is not adequate as the sole basis of sample vectorization. A contribution to the research question Q1 is that one should generate the full curvature state, and rely also on other point cloud and height derived features, if possible.

The (Fig. 8, P3) demonstrates the usage of the resulting classifier to large-scale prospecting. There $32 \times 32\, m^2$ samples were fed to the predictor and the resulting log-likelihoods visualized. The prediction is influenced by the set of positive teaching samples, which were of higher ground and lesser tree volume. A separate predictor could be tuned for woody areas, though.
Personal contribution: The algorithms, analysis, drafting of the manuscript.

(R4): Swimmer tracking without a camera model: an extension of (R2)

Summary: Nevalainen et al. (2017a) is an expansion of (R2) and has a title: *Real-time swimmer tracking on sparse camera array.* The error analysis covers the speed of the swimmer, location at the projection plane (20 cm towards the camera from the center line), and the camera positioning and orientation. A swimmer tracking is somewhat ad hoc method based on comparison of large rectangular compartments of the image. The detection of the swimming movement cycle duration is based on the silhouette tracking. The notation changes from (R2): An image is $\mathcal{I} = (P, I)$ where P are the pixel positions (or identities) and I is the usual map from a pixel to its intensity. The notation differs from (R2), it was intended to provide an easy extension for a future paper, which uses intensities $I(B)$ of rays of light B instead of pixels P.
Method: The location error analysis is a standard treatment of a measurements $(p, \mathbf{g})$, $p \in P$, $\mathbf{g} \in \mathbb{R}^3$, and an interpolation $(p, F(p))$, where p is an image, $\mathbf{g}$ is a calibration measurement in the real world (the projection plane), and F, which is the provided mapping from a pixel to the real world.

An analysis of the swimmer velocity accuracy in this setting has a specific difficulty: the frame frequency (50 Hz corresponding to ≈ 0.03 m movement) dictates an absolute lower limit to the time differentials. Therefore the speed analysis was performed at that definite limit (P4 Eq. 8) with an assumption of a maximum swimming speed of 8 pixels per second.

A considerable effort was put to analyze the errors caused from the camera positioning and orientation. The probabilistic treatment of the camera orientation is very complex, thus a numerical distribution algebra package (Python 3 Sympy) was used.

A simple absolute difference of pixels was used for the swimmer tracking. This introduces a lot of noise but is very fast.

Output and observations: The traditional approach to swimming analysis is to have a high density of cameras, the stereo-camera model and an ad hoc approach to the water-plexiglass-air refraction. Usually the systems are expensive and require a very elaborate camera positioning. It seems that the simple approach presented, supported by a robust and well checked algorithm implemented to an application specific integrated circuit (ASIC), could be a possibility for a real-time tracking system. A careful reconsideration should be done about what to compute in-real-time, and what to leave to a batch processing mode, and which kind of compression to perform before the long-term data storage. What comes to research question Q3, the conventional M and S methods seem to require much higher camera positioning and orientation tolerances than feasible by a limited budget.

Personal contribution: The algorithm development except about a half of the effort of developing the tracking method, which was done by DrTech Muhammad Hashem Haghbayan. Analysis of the data, drafting of the manuscript.

Errata: There is a typographic error in the paragraph between equations (10) and (11): $\mathbf{H}(\mathbf{u},\mathbf{u})$ and $\mathbf{H}(\mathbf{u},\mathbf{v})$ have to be replaced by $\mathbf{R_{uu}}$ and $\mathbf{R_{uv}}$. Also $\mathbf{R}(.,.)$ should be $\mathbf{R}_{..}$ in two occasions. Fig. 6 has a wrong caption text. It should read: *Swimming speed over 5 swimming cycles, non-smoothed version. The first cycle reveals the effect of bubbles. Every second cycle is sharp due to left-right hand discrepancy.*

(P5): Triangular mean curvature

Summary: A mean curvature is usually formulated by the volume expansion (see Eq. 3.36 in Sec. 3.7). In case of TIN, this requires references to three neighboring triangles producing a noisy triangular mean curvature estimate. This has been traditionally countered by formulating the curvature in the vertex points, whereby a reference to (an average of) 6 edges of 6 surrounding triangles is needed. To produce histograms over a sample area, one has to construct an elaborate dual area of the vertex (Mesmoudi et al., 2012).

An alternative approach of computing the triangular mean curvature is introduced in (P5) (Nevalainen et al., 2017b). A vertex expansion is used, which is accurate in the infinitesimal limit, and which leads to a very simple triangular mean curvature formula.

Method: The traditional formulation is based on the surface area expansion polynomial leading to definitions of the triangular mean curvature H_t and triangular Gaussian curvature G_t of Eq. 3.41 and Eq. 3.42. If one relaxes from the requirement of parallel expanded planes, one gets the averaged triangular mean curvature of Eq. (5) of (P5). Properties of this formulation are compared to current approaches. A benefit of Eq. (5) is that some commonly occurring noise patterns get filtered, even singular pikes get dampened only moderately.

The process of mapping a triangular feature X_t to a vertex feature X_p is presented in a concise fashion. A method of computing the principal curvature orientations is presented: the contribution of each orientation vector v_t and its exact opposite $-v_t$, $t \in T(p)$ have been made equal in the neighborhood voting process. The convergence on the infinitesimal limit has been visualized using an analytical torus model with synthetic std. noise.

A rare comparison of the produced curvature histogram and the analytical exact solution have been provided. Also a unique curvature analysis of a prostata lesion have been included as an example.

Output and observations: The proposed definition of the triangular mean curvature compares well in computational efficiency, although the analysis is not detailed. The triangular mean curvature seems to produce useful signal for micro-topography applications especially in the presence of sparse sampling, relatively high noise and curvature histogram vectorization, see (P6).

The mean curvature formulation proposed outperforms others in the presence of large noise. The edge signum (Eq. 14, P5) is a rather efficient formulation, which has been used in the CAD industry widely from 1980s.

There is an additional complication in computing the principal curvatures, see Eq. 20 in (P5). Due the possibility of curl in vertex normals due the noise, it is possible that the discriminant $H_t^2 - G_t$ is not positive, resulting complex primal curvature values. This must be countered by removing the curl component from the vertex normals (when considered as a vector field). This issue has not been addressed in this paper.

The analytical example of a noisy torus surface (Fig. 9, P5) demonstrates how a wide sortiment of possible histograms presented in Sec. 4.4 might be needed in various applications. In this case the Gaussian curvature histogram is very accurate, whereas the mean curvature and principal curvatures are not. The effect of noise is seen especially in estimation of the constant principal curvature $\kappa_2 = 1/r$ dictated by the smaller torus radius r.

This paper has been included to the thesis because it provides a view over the contemporary approaches to the curvature analysis of discrete surfaces, and so contributes to the research question Q2.

Personal contribution: algorithm development, analysis of the data, drafting of the manuscript. The magnetic-resonance imaging (MRI) data of the prostata lesion was provided by PhD Ivan Jambor and prepared to voxels by MSc Jussi Toivonen.

(P6): Micro-topography registration from point clouds, rut detection as a case

Summary: Several TIN analysis methods have been combined in order to visualize the rut formation of foresting machines. A method for producing directional curvature histograms directly from TIN is presented and applied to detect the local dominant direction of curvature in Nevalainen et al. (2017c). Data is produced by a UAV photogrammetry.

Method: The SAF process is applied both for the canopy height model and the ground TIN. A parameterized TIN thinning algorithm has been developed and documented. It increases the average distance of the natural neighbours to a given limit. (Natural neighbours p and q have a common triangle t, $p, q \in t \in T$). A mean curvature flow (MCF) is a TIN smoothing method, which alters the height values of points in order to reduce local extremities of the mean curvature (Crane et al., 2013). Histograms of the sample areas are derived by the globally projected vertex normals of Eq. 4.21, (Eq. A5, P6).

The detected dominant curvature directions, where there is a largest difference between two perpendicular histogram versions, have been used to visualize the TIN model and the machine trails. The visualization is used in manual control point insertion. Control points are an input for the convolution search, which produces an accurate match between numerical trail centerline and the actual trail in the point cloud. The rut profile analysis is based on rasterization, since the convolution scheme is numerically fastest to arrange on a traditional raster format.

Output and observations: The profile depth corresponds to manual observations with 0.65 accuracy, when two depth categories (less and more than 20 cm rut depth) are being used. This is enough for practical purposes. Two obstacles remain:

1. ML methods experimented to register the ruts produced too much noise, especially too many false positives among the young trees. New combinations of methods, e.g. team of experts approach, several classification categories (e.g. young trees, trails, open ground, canopy) should be used. The pair of histograms of directional curvatures seems to give rather good signal and should be included as one vector source to the team of experts approach.

2. The manual field measurements (rut depth) used were geo-referenced, although not very well (3 geomarkers were used when 5-6 is the standard with the equipment and algorithms used). Experiments with more data and with less or no georeferencing at all should be made to determine the geometric quality of the point cloud under this more economical arrangement.

The paper contributes to reasearch question G1. The ground TIN produced by SAF process described seems to produce a high-quality ground model even at the approximity of the canopy front. The HOC method is numerically efficient, and shold be experimented with over a larger variety of microtopogrpahy problems.
Personal contribution: The algorithm development, analysis of the data, drafting of the manuscript. The background and motivation from the forestry point of view was written by PhD Aura Salmivaara Nature Resource Center of Finland (Luke). The data and the initial photogrammetric processing was done by Juuso Hiedanpää (Metsälinkki Ltd.).

Chapter 7

Conclusions

This section summarizes the contributions given in six original publications included into thesis. We also consider extensions of the algorithms and methods for more general settings in future work as well.

7.1 Contributions of the thesis

In this thesis several new algorithms were introduced.

- SAF (P3), which delivers ground and canopy TIN models from a point cloud. It requires two parameters, which have to be tuned by a CV process.
- A fast HOC (P6) method, which produces a directed curvature histogram from the ground TIN model.
- A useful TIN thinning algorithm (P6) for multi-scale analysis. It has one parameter, the average distance between neighboring points. It is tailored to a situation where the ground model is bordered by the possibly fragmented canopy front.
- Direct geometric projection method for swimming analysis was experimented with and the accuracy compared to conventional methods. Although this is not a new method, it is demonstrated that it is a feasible alternative for a low-budget swimming analysis.

It seems that the triangular mean curvature of Eq. 4.6 (P5) and triangular Gaussian curvature (Eq. 4.7) are useful new ingredients to the micro-topography analysis.

When compared to the rather generic title of this thesis, the presentation naturally falls short of covering all the aspects of geometric data understanding. Especially the theoretical considerations have been limited to exclude the probabilistic modeling and finesse of the DDG notation. Multi-scale models are an important tool in micro-topography but were mostly excluded, except in (P1) and (R2). General 3D analysis of swimming performance, and bundle adjustment approach similar to (Hansen, 1992), has been excluded.

7.2 Future work

There are various open problems concerning e.g. the computational costs of the total work flow and of alternative point cloud problem formulations.

(R4) introduces a ground TIN model with 2 parameters. There are possibilities to develop a ground TIN model with only one parameter. One possible model would be an information theoretical one with a triangular curvature penalty e.g. the Willmore energy (Grinspun and Desbrun, 2006) $H^2 - G \propto \kappa_1^2 + \kappa_2^2$ as for a regularization term. What is missing is the efficient categorization of cloud points similar to the RANSAC (Hast et al., 2013) before a minimum description length (MDL) formulation can be casted. The resulting ground model would have either local scale factor (Yang and Meer, 2017) or information per area as the only parameters. Also, Taubin's formulation includes a smoothing method (Taubin, 1995b), which should be compared with SAF related smoothing, TIN regularization and a multi-scale curvature analysis.

There are open questions like implementation of multi-scale modeling of ground TINs in a theoretically sound way, and connections to MDL and smoothness regularization, which have not been covered in this thesis and which are not yet present in standard industry implementations. It may be possible that some old formulations like Taubin (1995a) or Theisel et al. (2004) can be casted to a multi-scale formulation. The method of Theisel et al. (2004) can provide a competitive formulation of HOC. The HOC method of Sec. 4.4 uses 3 vector dot products and one vector normalization and stores 2 vectors per vertex, When (Taubin, 1995a) is extended to output the directional curvature, it uses only 2 vector dot products and stores 2 vectors. The method is superior if principal vectors are required. The HOC formulation requires 16 additional vector dot products, whereas (Taubin, 1995a) needs approx. only 2 dot products.

The natural resource data seems to have a discrepancy between the size of available data (approx. 1.3×10^9 samples) and the size of typical field measurement campaign (typically $10^{2\ldots3}$ samples). This calls for a blend of unsupervised and supervised methods, which hopefully could take into account the (very diminutive) set of field measurements.

Deformation states between the point cloud and a local shape probe (Digne et al., 2017) can be collected to a library, which neatly addresses point cloud subsets of various local dimensionality. This approach addresses both the raster data and point clouds, but it can be adopted to the TIN models. Local parts of a TIN usually have only one or two dimensional subparts. Applicatios and variations of the local shape probe remain as a potential research topic.

The SAF algorithm has to be implemented so that it can be an efficient part of the CV process. There are some algorithmic details and potential for optimization by approximating both the spatial angles and the sort ordering of the spatial angles, which requires some study.

An applicable theoretic framework combining a parameterized geometric model as a one component and geometrically meaningfully defined noise as another other component would integrate the research. Candidates for such a theoretical framework are e.g. MDL, stochastic geometry (Schneider and Weil, 2008), information geometry (Nielsen, 2010) and geometric algebra (Dorst et al., 2007) (GA). The MDL approach includes a difficult sub-problem of defining outliers efficiently. The GA provides an interesting formulation for the local point cloud covariance matrix, which is often used to evaluate various dimensional saliency values. The local covariance can be presented as a geometric object subjected to normal algebraic manipulations, and some combinations of common geometric objects can be covered concisely.

The D method of papers (R2) and (R4) extends easily to the 3D analysis. By using two planes G_1 and G_2 depicted in Fig. 7.1, each pixel can be given a beam with an observed intensity, and the beam field with its natural neighborhood relations can be used to quickly find out the depth information from observed surfaces (colored bars on the swimmer body). The end result is a priori distribution for bio-mechanical posture detection. The actual bio-mechanical underwater 3D surface registration is still a focus of research, the key issues concern the presence of bubbles and blurred featureless surfaces.

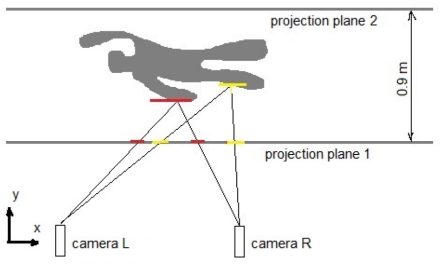


Figure 7.1: The direct projection method, a 3D approach.

The proposed direct calibration approach adapts to the coming synthetic aperture photography (light-field or plenoptic cameras) (Ng et al., 2005), which are well-suited for swimming research, since a dense array of cheap plenoptic cameras is easy to install, do not protrude to the swimming spool space, and is able to reduce the effect of light scattering and bubbles, thus rendering the ordinary imaging methods (using e.g. anchor points) usable.

References

Aggarwal, C. C. (2006). *Data Streams: Models and Algorithms (Advances in Database Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Amazon.com, I. (2018). Machine learning - algorithms. https://s3.amazonaws.com/MLMastery/MachineLearningAlgorithms.png. Accessed: Feb 1st 2018.

Bäuer-Burchardt, C., Heinze, M., Schmidt, I., Kühmstedt, P., and Notni, G. (2016). Underwater 3d surface measurement using fringe projection based scanning devices. *Sensors*, 16(1).

Berthold, M., Borgelt, C., Höppner, F., and Klawonn, F. (2010). *Guide to intelligent data analysis : how to intelligently make sense of real data*. London [u.a.] : Springer.

Bezanson, J., Karpinski, S., Shah, V. B., and Edelman, A. (2012). Julia: A fast dynamic language for technical computing. *CoRR*, abs/1209.5145.

Bishop, M. P., James, L. A., Jr., J. F. S., and Walsh, S. J. (2012). Geospatial technologies and digital geomorphological mapping: Concepts, issues and research. *Geomorphology*, 137:5–26.

Bouguet, J. Y. (2008). Camera calibration toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/. Accessed: Mar 8th 2018.

Brubaker, K. M., Myers, W. L., Drohan, P. J., Miller, D. A., and Boyer, E. W. (2013). The use of LiDAR terrain data in characterizing surface roughness and microtopography. *Applied and Environmental Soil Science*, 2013.

Buiteveld, H., Hakvoort, J. H. M., and Donze, M. (1994). Optical properties of pure water. In *Proc. SPIE 2258, Ocean Optics XII*.

Burnham, K. P. and Anderson, D. R. (2004). Multimodel Inference: Understanding AIC and BIC in Model Selection. *Sociological Methods & Research*, 33(2):261–304.

Chang, K.-Y. and Ghosh, J. (2001). A unified model for probabilistic principal surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):22–41.

Chawla, N. (2010). Data mining for imbalanced datasets: An overview. In *Data Mining and Knowledge Discovery Handbook*, pages 875–886.

Chen, X. and Schmitt, F. (1992). Intrinsic surface properties from surface triangulation. In Sandini, G., editor, *Computer Vision — ECCV'92*, pages 739–743, Berlin, Heidelberg. Springer Berlin Heidelberg.

Colding, B. N. (1961). Machinability of metals and machining costs. *International Journal of Machine Tool Design and Research*, 1:220–248.

Conway, D. (2010). The data science venn diagram. http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram. Accessed: May 7th 2018.

Crane, K., de Goes, F., Desbrun, M., and Schröder, P. (2013). Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, pages 7:1–7:126, New York, NY, USA. ACM.

Criminisi, A. (1999). *Accurate Visual Metrology from Single and Multiple Uncalibrated Images*. PhD thesis, University of Oxford, Dept. Engineering Science. D.Phil. thesis.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA. IEEE Computer Society.

Dang, T., Hoffmann, C., and Stiller, C. (2009). Continuous stereo self-calibration by camera parameter tracking. *IEEE Transactions on Image Processing*, 18(7):1536–1550.

de Smith, M. J., Goodchild, M. F., and Longley, P. A. (2015). *Geospatial Analysis: A Comprehensive Guide to Principles, Techniques and Software Tools*. Troubador Publishing Ltd, 5 edition. See: http://www.spatialanalysisonline.com/HTML/index.html?profiles_and_curvature.htm.

D'Errico, J. (2006). Surface fitting using gridfit. Technical report, MATLAB Central File Exchange.

Desbrun, M., Kanso, E., and Tong, Y. (2008). Discrete differential forms for computational modeling. In A. I. Bobenko and, J. S., Schröder, P., and Ziegler, G., editors, *Discrete Differential Geometry*, volume 38. Oberwolfach Seminars, Birkhäuser Basel.

Desbrun, M., Meyer, M., Schröder, P., and Barr, A. H. (2000). Discrete differential-geometry operators in nd. pages 35–57. Springer-Verlag.

Digne, J., Valette, S., and Chaine, R. (2017). Sparse geometric representation through local shape probing. *IEEE Transactions on Visualization and Computer Graphics*, pages 1 – 1.

Dorst, L., Fontijne, D., and Mann, S. (2007). *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Elmasri, R. and Navathe, S. (2010). *Fundamentals of Database Systems*. Addison-Wesley Publishing Company, USA, 6th edition.

Fischer, P. and Brox, T. (2014). Image descriptors based on curvature histograms. In *German Conference on Pattern Recognition (GCPR)*, LNCS. Springer.

Fortune, S. (1997). Handbook of discrete and computational geometry. chapter Voronoi Diagrams and Delaunay Triangulations, pages 377–388. CRC Press, Inc., Boca Raton, FL, USA.

Furukawa, Y. and Ponce, J. (2009). Accurate camera calibration from multiview stereo and bundle adjustment. *Int. J. Comput. Vision*, 84(3):257–268.

Gai, S., Da, F., and Fang, X. (2016). A novel camera calibration method based on polar coordinate. *PLoS ONE*, 11(10).

Gatzke, T. D. and Grimm, C. M. (2006). Estimating curvature on triangular meshes. *International Journal of Shape Modeling*, 12(01):1–28.

Gold, C. M. (1989). Surface interpolation, spatial adjacency and gis. In Raper, J., editor, *Three Dimensional Applications in Geographic Information Systems*, pages 21–35. Taylor and Francis, London.

Golovinskiy, A. and Funkhouser, T. (2009). Min-cut based segmentation of point clouds. In *IEEE Workshop on Search in 3D and Video (S3DV) at ICCV*.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA.

Grinspun, E. and Desbrun, M., editors (2006). *Discrete differential geometry: an applied introduction*, ACM SIGGRAPH Courses Notes. ACM Press New York, NY, USA.

Hadwiger, H. (1957). *Vorlesungen über Inhalt, Oberfläche und Isoperimetrie*. Springer-Verlag, Berlin.

Haner, S., Svärm, L., Ask, E., and Heyden, A. (2015). Joint under and over water calibration of a swimmer tracking system. In *Proceedings of the International Conference on Pattern Recognition Applications and Methods*, pages 142–149. ScitePress.

Hansen, P. (1992). Numerical solution of laplaceś equation. *Electri. Eng. and Comp. Sci., Technical Report*, 9.

Hast, A., Nysjö, J., and Marchetti, A. (2013). Optimal ransac - towards a repeatable algorithm for finding the optimal set. *Journal of WSCG*, 21(1).

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Heikkilä, J. and Silven, O. (1997). A four-step camera calibration procedure with implicit image correction. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1106–1112.

Karpowicz, J. (2016). Figuring out aerial surveying with a drone instead of arguing about photogrammetry vs LiDAR. https://www.expouav.com/news/latest/figuring-aerial-surveying-drone-instead-arguing-photogrammetry-vs-lidar/. Expo UAV: Accessed: May 12th 2018.

Kim, M.-K., Kim, S., Sohn, H.-G., Kim, N., and Park, J.-S. (2017). A new recursive filtering method of terrestrial laser scanning data to preserve ground surface information in steep-slope areas. *ISPRS International Journal of Geo-Information*, 6(11).

Klasing, K., Althoff, D., Wollherr, D., and Buss, M. (2009). Comparison of surface normal estimation methods for range sensing applications. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3206–3211. IEEE.

Kondor, I. R. (2008). *Group Theoretical Methods in Machine Learning*. PhD thesis, Columbia University, New York, NY, USA. AAI3333377.

Kreyszig, E. (1959). *Differential geometry*. Mathematical expositions. University of Toronto Press.

Li, Y., Yong, B., van Oosterom, P., Lemmens, M., Wu, H., Ren, L., Zheng, M., and Zhou, J. (2017). Airborne LiDAR data filtering based on geodesic transformations of mathematical morphology. *Remote Sensing*, 9(11):1104.

Lu, H., Li, Y., Zhang, Y., Chen, M., Serikawa, S., and Kim, H. (2017). Underwater optical image processing: A comprehensive review. *CoRR*, abs/1702.03600.

Luczynski, T., Pfingsthorn, M., and Birk, A. (2017). The pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings. *Ocean Engineering*, 133(Supplement C):9 – 22.

Magnussen, S., Nord-Larsen, T., and Riis-Nielsen, T. (2018). Lidar supported estimators of wood volume and aboveground biomass from the danish national forest inventory (2012Ű2016). *Remote Sensing of Environment*, 211:146 – 153.

Mao, Q., Wang, L., Tsang, I., and Sun, Y. (2016). Principal graph and structure learning based on reversed graph embedding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1.

Matsakis, P. (2016). Affine properties of the relative position phi-descriptor. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE.

Maturana, D. and Scherer, S. (2015). 3d convolutional neural networks for landing zone detection from LiDAR. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3471–3478.

Meek, D. and Walton, D. (2000). On surface normal and Gaussian curvature approximations given data sampled from a smooth surface. *Computer Aided Geometric Design*, 17(6):521 – 543.

Meng, X., Currit, N., and Zhao, K. (2010). Ground filtering algorithms for airborne LiDAR data: A review of critical issues. *Remote Sensing*, 2(3):833.

Mesmoudi, M. M., De Floriani, L., and Magillo, P. (2012). Discrete curvature estimation methods for triangulated surfaces. In *Applications of Discrete Geometry and Mathematical Morphology*, pages 28–42. Springer.

Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H. (2003). *Visualization and Mathematics III*, chapter Discrete Differential-Geometry Operators for Triangulated 2-Manifolds, pages 35–57. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mullin, M. and Sukthankar, R. (2000). Complete cross-validation for nearest neighbor classifiers. In *17th International Conference on Machine Learning (ICML)*.

Mündermann, L., Corazza, S., and Andriacchi, T. P. (2006). The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications. *Journal of NeuroEngineering and Rehabilitation*, 3(6).

Naula, P., Airola, A., Salakoski, T., and Pahikkala, T. (2014). Multi-label learning under feature extraction budgets. *Pattern Recognition Letters*, 40:56–65.

Nevalainen, P., Haghbayan, M. H., Kauhanen, A., Pohjankukka, J., Laakso, M., and Heikkonen, J. (2017a). Real-time swimmer tracking on sparse camera array. In Fred, A. L. N., Marsico, M. D., and di Baja, G. S., editors, *Pattern Recognition Applications and Methods - 5th International Conference, ICPRAM 2016, Rome, Italy, February 24-26, 2016, Revised Selected Papers*, volume 10163 of *Lecture Notes in Computer Science*, pages 156–174.

Nevalainen, P., Jambor, I., Pohjankukka, J., Heikkonen, J., and Pahikkala, T. (2017b). Triangular curvature approximation of surfaces: Filtering the spurious mode. In Marsico, M. D., di Baja, G. S., and Fred, A. L. N., editors, *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2017, Porto, Portugal, February 24-26, 2017.*, pages 457–466. SciTePress.

Nevalainen, P., Kauhanen, A., Raduly-Baka, C., and Heikkonen, J. (2016a). Video based swimming analysis for fast feedback. In Marsico, M. D., di Baja, G. S., and Fred, A. L. N., editors, *ICPRAM*, Proc. of Intern. Conf. on Pattern Recognition Applications and Methods (ICPRAM), pages 457–466. SciTePress.

Nevalainen, P., Middleton, M., Kaate, I., Pahikkala, T., Sutinen, R., and Heikkonen, J. (2015). Detecting stony areas based on ground surface curvature distribution. In *2015 International Conference on Image Processing Theory, Tools and Applications, IPTA 2015, Orleans, France, November 10-13, 2015*, pages 581–587.

Nevalainen, P., Middleton, M., Sutinen, R., Heikkonen, J., and Pahikkala, T. (2016b). Detecting terrain stoniness from airborne laser scanning data. *Remote Sensing*, 8(9):720.

Nevalainen, P., Salmivaara, A., Ala-Ilomäki, J., Launiainen, S., Hiedanpää, J., Finér, L., Pahikkala, T., and Heikkonen, J. (2017c). Estimating the rut depth by UAV photogrammetry. *Remote Sensing*, 9(1279).

Ng, R., Levoy, M., Brédif, M., Duval, G., Horowitz, M., and Hanrahan, P. (2005). Light field photography with a hand-held plenoptic camera. Technical Report Tech Report CSTR 2005-02, Stanford University Computer Science.

Nguyen, T.-N., Huynh, H.-H., and Meunier, J. (2016). Skeleton-based abnormal gait detection. *Sensors*, 16(11).

Nielsen, F. (2010). Legendre transformation and information geometry. Technical Report CIG-MEMO2. http://www.informationgeometry.org.

Okabe, A., Boots, B., Sugihara, K., and Chiu, S. (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Series in Probability and Statistics. John Wiley and Sons, Inc., 2 edition.

Page, D. L., Sun, Y., Koschan, A. F., Paik, J., and Abidi, M. A. (2002). Normal vector voting: Crease detection and curvature estimation on large, noisy meshes. *Graphical Models*, 64:199–229. Elsevier.

Plaza-Leiva, V., Gomez-Ruiz, J., Mandow, A., and García-Cerezo, A. (2017). Voxel-based neighborhood for spatial shape pattern classification of LiDAR point clouds with supervised learning. *Sensors*, 17(3).

Pohjankukka, J., Pahikkala, T., Nevalainen, P., and Heikkonen, J. (2017). Estimating the prediction performance of spatial models via spatial k-fold cross validation. *International Journal of Geographical Information Science*, 31(10):2001–2019.

Pressley, A. (2010). *Elementary Differential Geometry*. Springer Undergraduate Mathematics Series. Springer London.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2016). Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*.

Rashevskii, P. K. (1956). Kurs differentiarnoi geometrii. Moscow Univ.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5):465–471.

Rocklin, M. and Terrel, A. R. (2012). Symbolic statistics with sympy. *Computing in Science & Engineering*, 14(3):88–93.

Rusinkiewicz, S. (2004). Estimating Curvatures and Their Derivatives on Triangle Meshes. Technical Report TR-693-04, Department of Computer Science, Princeton University.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3859–3869.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3):210–229.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42.

Schneider, R. and Weil, W. (2008). *Stochastic and Integral Geometry*. Probability and its Applications. Springer, Berlin, Germany.

Sever, A. (2015). An inverse problem approach to pattern recognition in industry. *Applied Computing and Informatics*, 11(1):1–12.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, pages 517–524, New York, NY, USA. ACM.

Sober, B. and Levin, D. (2016). Manifolds' projective approximation using the moving least-squares (MMLS). *CoRR*, abs/1606.07104.

Solomon, J., Crane, K., and Vouga, E. (2014). Laplace-beltrami: the swiss army knife of geometric processing. sgp2014 tutorial. http://ddg.cs.columbia.edu/SGP2014/LaplaceBeltrami.pdf. Accessed: May 10th 2018.

Taubin, G. (1995a). Curve and surface smoothing without shrinkage. In *Proceedings of the Fifth International Conference on Computer Vision*, ICCV '95, pages 852–, Washington, DC, USA. IEEE Computer Society.

Taubin, G. (1995b). Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of the Fifth International Conference on Computer Vision*, ICCV '95, pages 902–, Washington, DC, USA. IEEE Computer Society.

Theisel, H., Rössl, C., Zayer, R., and Seidel, H. P. (2004). Normal based estimation of the curvature tensor for triangular meshes. In *In PG04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG2004)*, pages 288–297. IEEE Computer Society.

Trauth, M. H. (2007). Statistics on directional data. In *MATLAB(R) Recipes for Earth Sciences*. Springer, Berlin, Heidelberg.

van Oosterom, A. and Strackee, J. (1983). A solid angle of a plane triangle. *IEEE Trans. Biomed. Eng.*, 30(2):125–126.

Varma, S. and Simon, R. (2006). Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics*, 7(91).

Vupparaboina, K. K., Raghavan, K., and Jana, S. (2015). Euclidean auto calibration of camera networks: Baseline constraint removes scale ambiguity. *CoRR*, abs/1510.01663.

Wang, Z., Li, H., Ouyang, W., and Wang, X. (2016). Learnable histogram: Statistical context features for deep neural networks. In *ECCV (1)*, volume 9905 of *Lecture Notes in Computer Science*, pages 246–262. Springer.

West, D. B. (2000). *Introduction to Graph Theory*. Prentice Hall, 2 edition.

Xu, L., Wang, R., Zhang, J., Yang, Z., Deng, J., Chen, F., and Liu, L. (2015). Survey on sparsity in geometric modeling and processing. *Graph. Models*, 82(C):160–180.

Yang, P. and Qian, X. (2007). Direct computing of surface curvatures for point-set surfaces. In *SPBG'07*, pages 29–36.

Yang, X. and Meer, P. (2017). Scale Adaptive Clustering of Multiple Structures. *ArXiv e-prints*.

Yang, Y. (2007). Prediction/estimation with simple linear models: Is it really that simple? *Econometric Theory*, 23(1):1Ũ36.

Yutaka, A. B. and Ohtake, E. B. Y. (2003). A comparison of mesh smoothing methods. In *In Proceedings of the Israel-Korea BiNational Conference on Geometric Modeling and Computer Graphics*, pages 83–87.

Zhang, Y., Zhou, L., Liu, H., and Shang, Y. (2016). A flexible online camera calibration using line segments. *Journal of Sensors*, 2016:1–16. Article ID 2802343.

Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. In *in ICCV*, pages 666–673.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334.

Publication P1

Detecting stony areas based on ground surface curvature distribution

P. Nevalainen, M. Middleton, I. Kaate, T. Pahikkala, R. Sutinen, J. Heikkonen. IPTA 2015 The 5th International Conference on Image Processing Theory, Tools and Applications, 11/2015.

Detecting stony areas based on ground surface curvature distribution

Paavo Nevalainen[1], Maarit Middleton[3], Ilkka Kaate[2],
Tapio Pahikkala[1], Raimo Sutinen[3], Jukka Heikkonen[1]
[1] Department of Information Technology, University of Turku, FI-20014 TURKU, FINLAND
e-mail: ptneva@utu.fi,Tapio.Pahikkala@utu.fi,Jukka.Heikkonen@utu.fi
[2] Department of Geography and Geology, University of Turku, FI-20014 TURKU, FINLAND
e-mail: Ilkka.Kaate@utu.fi
[3] Geological Survey of Finland, P.O.Box 77, Lähteentie 2, 96101 ROVANIEMI, FINLAND
e-mail: Maarit.Middleton@gtk.fi, Raimo.Sutinen@gtk.fi

Abstract— **Presence of ground surface stones is one indicator of economically important landmass deposits in the Arctic. The other indicator is a geomorphological category of the area. This work shows that ground stoniness can be automatically predicted with practical accuracy. Northern forests have less biomass and foliage, thus direct analysis of stoniness is possible from airborne laser scanning (ALS) data. A test set of 88 polygons covering $3.3\,km^2$ was human-classified and a method was developed to perform the stoniness classification over this set. The local curvature of the surface is approximated directly from the point cloud data without generating the Digital Terrain Model (DTM). The method performs well with area under curve $AUC = 0.85$ from Leave-Pair-Out cross-validation, and is rather insensitive to missing data, moderate forest cover and double-scanned areas.**

Keywords— **Logistic regression, aerial laser scanning, remote sensing, natural resources, curvature, ground relief, point clouds, open data.**

I. Introduction

This work is about the stones and bolders detection at mass-flow deposits in Northern Finland. Mass-flow deposits [12] have become under consideration as a potential sediment supply for infrastructure construction. They are convex landforms composed of poorly sorted coarse-grained sediments occurring in topographic depressions as fields of tens or hundreds of hummocks. They are characterized by irregular distribution of boulders and stones on their surfaces although the size distribution and density varies greatly. The surface stoniness is a key parameter for recognition of the mass flow deposits.

The characteristics described above can be detected by automated geomorphology analysis. The automation lacks classification to stony and non-stony areas, though. Kemijärvi district in Northern Finland provided polygon areas with expert classification to 43 positive (stony) and 45 negative (not stony) sites, which were used to verify the stoniness classification method documented here.

The next two subsections present the data used and the current state of the art on stoniness detection and related research. Ch. 2 outlines the local curvature method and compares it to the digital terrain based method. Ch. 3 presents results, and Ch. 4 brings conclusions.

A. Test data

Airborne Light Detection and Ranging (LiDAR) is an active remote sensing technique that measures travelling time of a sent high energy monochromatic light beam to determine distance to a target. Intensity of the these backscattered laser pulses is also recorded. The airborne LiDAR data by the National Land Survey of Finland (NLS) was obtained in the fall 2012 with a Leica ALS50-II laser scanner (Leica Geosystems, St. Gallen, Switzerland) from a flight altitude of 2000 m. Last-return data allows in this case $0.09 - 0.14\,m$ vertical ground resolution.

This work concerns the point cloud coordinates $q \in \mathbb{R}^3$ only. The additional information available at laser exchange format (LAS) specification [3] is not used.

The point cloud in this specific instance has c. 20-30 % hits to forest canopy, the ground signal is rather clear. Also, the stones and boulders are relatively large, the actual size distribution has not been measured. Stones are bare and the vegetation is thin due to northern latitude. On the other hand, the point cloud is sparse with the sample density $\rho \approx 0.8/m^2$ at the ground level and the stone detection task is close to the theoretical limit. Requiring 1..2 hits per stone one gets an approximation for the minimum detectable stone radius r_{min}: $2\pi r_{min}^2\,\rho \approx 1..2$ (samples) $\rightarrow r_{min} \approx 0.5..0.6\,m$. Stones of this size may have a detectable effect, if occurring at large numbers.

Total area of the positive sample polygons is $1.7\,km^2$ and the number of LAS points is 1.6×10^6. The same numbers hold for the negative samples.

B. Current approaches to detect stones

There seems to be no research about aerial ALS data and stoniness detection applied to forest areas. Most of the research on rock exposure and rock mass characterization is based on on-site scanning with considerably higher point densities. Usually the target area has no tree cover, objects are elongated (walls, ditches, archaelogical road tracks, etc.) and often multi-source data like photogrammetry or wide-spectrum tools are used. E.g. [14] detects curbstones which separate the pavement and road. The sample density $\rho = 5/\,m^2$ produces geometric error of size $0.3\,m$ which is larger than the observed shapes (curbstones) and thus not practical.

Digital Terrain Model (DTM, or public terrain model in Fig. 1) is a standard analysis concept used by Geographic Information Systems (GIS). Many implementations and heuristics exist to form DTM from LAS. Usually, the smallest raster grid size is dictated by the sample density and in this case grid size $\delta = 2\,m$ is possible, and $\delta = 1m$ already suffers from numerical instability and noise.

A rare reference to DTM based detection of a relatively small local ground feature (cave openings) at forest circumstances is in [15]. In that paper the target usually is at least partially exposed from canopy and the cave opening is more than 5 meters of diameter. On the other hand, the forest has more biomass.

One reference [4] lists several alternative LiDAR based DTM features, which could be used in stone detection, too. Some of the features are common in GIS software, but most should be implemented for stoniness detection.

Hough method adapted to finding hemispherical objects is considerably slower, although there is recent publication about various possible optimizations, see e.g. [7]. These optimizations are mainly about better spatial partitioning.

Minimum Description Length (MDL) is presented in [16] with an application to detect planes from the point cloud. The approach is very basic, but can be modified for hemispherical objects rather easily. MDL formalism can provide an easy choice between two hypotheses: *a plain spot/a spot with a stone*.

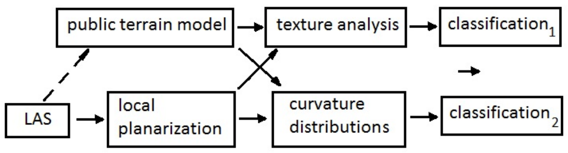


Fig. 1. Alternatives for stoniness detection. The public terrain model is nationwide, but does not provide finer details of local planarization.

The GIS approach to stone detection is missing but would have three steps:

1) Generating either a Triangulated Irregular Network (TIN) or DTM raster file from .las binary file. DTM raster files are publicly available from NLS. [1]
2) One or many texture features generated from DTM
3) texture classification

The texture features possible are the following:

- local height difference, see Laplace filtering in upper row in Fig. 4. An alternative is difference of a point cloud spline surface and a coarse DTM.
- various roughness measures, e.g. rugosity (related trigonometrically to the average slope), curvature, standard deviation of slope, standard deviation of curvature, mount leveling metric (opposite to a pit fill metric mentioned in [4])
- Multiscale curvature presented in [6]. It is used for dividing the point cloud to ground and non-ground returns, but could be modified to bring both texture information and curvature distribution information. The latter could be then used for the stoninesss prediction like in this study. The methods, possibly excluding TIN based interpolation, seem to be numerically more costly than our approach.

The actual texture classification method would be heavily influenced on the choices made above. Most of the features listed are standard tools in GIS systems or can be implemented by minimal coding. E.g. the mount leveling metric would require negating the height.

Terrain roughness studied in [4] is a concept which is close to stoniness. Authors mention that the point density increase from $\rho = 0.7/m^2$ to $\rho = 10/m^2$ did not improve the terrain roughness observations considerably. This is understandable since the noise at the vertical components of surface hits is at the range of the higher density. The paper states that algorithms producing the features have importance to success. This led us to experiment with various new algorithms.

Point cloud features based on neighborhoods of variable size are experimented with in [10]. Many texture recognition problems are sensitive to the raster scale used, thus we tested a combination of many scales, too.

In comparison to previous references, this approach is rather independent study based on four facts: point cloud density is low, ground hit percentage is high providing relatively even coverage, a direct approach without texture methods was preferred, and the method is for a single focused application. Also, we wanted to avoid complexities described in [8] and keep the method parameters tunable by cross-validation.

II. Local curvature approximation

This presentation uses the following concepts, most of which will be elaborated later in the text:

- sample polygon is a geographical area, either stony or not stony. See Fig. 4.
- points $q \in Q \subset \mathbb{R}^3$ of the LAS point cloud Q
- raster of size $n_1 \times n_2$, $n_1, n_2 \in \mathbb{N}$ and cell size $\delta \in \mathbb{R}^+$
- raster grid locations $k \in I \subset \{1, ..., n_1\} \times \{1, ..., n_2\}$, for indexing. I covers only the sample polygon area.
- raster grid points $c_k \in \mathbb{R}^2$. The grid is on the horizontal (x, y) plane, x corresponding to ETRS-TM35FIN east coordinate and y to north.
- approximate topological heights $z_k \in \mathbb{R}$ of the approximated ground surface
- surface points $p_k^T = (c_k^T\; z_k)$, $\{p_k\}_{k \in I} = P \subset \mathbb{R}^3$
- surface normals $\{\mathbf{n}_k\}_{k \in I} = N \subset \mathbb{R}^3$
- nearest cloud points $\{Q_k\}_{k \in I, Q_k \subset Q}$ from the location k, see Fig. 2
- surface triangularization $J \subset I^3$, see Fig. 2

[1]NLS laser data: http://www.maanmittauslaitos.fi/en/maps-5

- adjoining triangles V_k at location $k \in I$, see Fig. 2
  $V_k = \{(j_0, j_1, j_2) \in J \,|\, \exists\, l \in \{0,1,2\}, j_l = k\}$

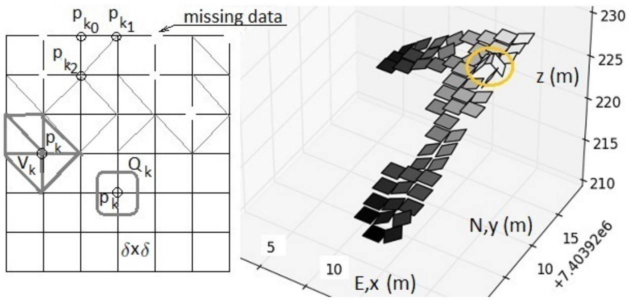


Fig. 2. Left: The triangulation of the grid avoids the incomplete squares. A triangle (k_0, k_1, k_2) and local cloud point set Q_k depicted. Right: A stone revealed by two adjacent tilted planes. That stone gets detected only with $\delta = 2\,m$.

The surface points P, normals N, and triangularization J form a surface model $\mathcal{S} = (P, N, J)$. Heights z_k and normals n_k are approximated based on local point cloud subsets Q_k. The aim is to compute a heuristic curvature κ_k at each grid location k and then use the resulting normalized histogram over a sample area for classification. The heuristic curvature κ is based on Gaussian curvature G and mean curvature H. The problem of approximating G and H from a ground surface triangulation J is presented in [5] and [13]. We gained more prediction accuracy by not generating the normal vectors from the triangular mesh (P, J) as in [5] but producing normals in a separate fitting process $Q_k \Rightarrow \mathbf{n}_k$ (see App. I), and then applying the single triangle formulation of [13].

The histogram of curvature over each polygon proved to be simple and efficient basis for classification. One does not need to construct the DTM surface for that, local sampling of the point cloud is enough. Also the texture analysis can be skipped.

The algorithm steps 1...3 are repeated once per each grid constant δ_m, $m = 1...6$, see Table 1. The steps are:

1) Estimating the ground height z_k and ground normal $\mathbf{n}_k$ by local plane fitting at a grid location k.
2) Curvatures calculated at the vertices of each triangle of the grid, see Fig. 2.
3) Local curvature estimation at location k based on voting among the adjoined triangles V_k.
4) Curvature histograms are normalized, vectorized ...
5) ...and used by logistic regression.

Vector voting methods listed in [9] decrease noise and achieve good approximative surface normals for symmetrically noisy data sets. Our target cloud has asymmetrical noise (vegetation hits are always above the ground), and hits under the ground (reflections) are extremely rare. Finding the local planes is very straight-forward adaptation of [11]. The plane fitting process resembles usual localized principal component analysis (PCA) except the distance weight is not symmetrical, but penalizes heavily points immediately below the local ground manifold while attempting to ignore the points well above the ground. After local planes are generated, local curvatures are calculated by adaptation of the process described in [13].

A more detailed presentation of steps 1-5 follows:
1: Approximating the local planar fit. The regular grid points c_k are given the approximate local heights z_k and normals $\mathbf{n}_k$ by a plane fitting process which uses local cloud points Q_k. The fitting algorithm is described in the App. I. The result is the surface model $\mathcal{S}$.

Sec. 3.2 of [2] presents a very similar approach, except the fitting function $g(.)$ used there is quadratic and intended for well-pruned (thin) and dense manifolds of points. To summarize this step: $c_k, Q_k \Rightarrow (p_k, \mathbf{n}_k)$. See the App. I for details.

2: This step uses the ground surface approximation $\mathcal{S}$ to estimate local curvatures κ_{ki}, $k \in I, i \in V_k$ at all vertices of all triangles J. Ch. 3 of [13] provides a simple and elegant barymetric interpolation of curvature properties over a triangle $j \in J$, when triangle corner points and the corresponding surface normals are given. Triangle set J is produced by taking care that every grid square with only 3 points gets a triangle, then filling the complete squares by splitting them to two triangles randomly. See Fig. 2. The incomplete squares are a result of too sparse or completely missing cloud points.

The exposition in [13] ends to expressions for Gaussian curvature K and mean curvature H of the ground relief $\mathcal{S}$ limited to a triangle. Curvatures are calculated using heuristics of App. II. To summarize this step: $\mathcal{S}, J \Rightarrow \kappa_{kj}$, $j \in V_k$.

3: Each grid location k gets several nominal curvature candidates, one per each adjoining triangle vertex $j \in V_k$. The final value is the mode of the candidates. The summary: $\{\kappa_{kj} | j \in V_k\} \Rightarrow \kappa_k$.

4: The final preliminary step before the supervised learning is the vectorization process. Each sample polygon $i \in 1..n$, $n = 88$ gets analyzed with a grid constant δ_r, $r = 1..6$ resulting in a histogram vector $\mathbf{x}_{ir}$ of all curvatures at the sample polygon i. A summary: $\delta_r, \{\kappa_k\}_{k \in I} \Rightarrow \mathbf{x}_{ir}$.

Table 1. Grid constants used.

Grid m	1	2	3	4	5	6
δ_m (m)	1.25	2.0	3.0	4.0	5.0	6.0

The histogram is more refined near the zero curvature, the exact definition of the bin limits is here:

$$\kappa_0 = 0.04\ 1/m,\ \kappa_6 = 1.8\ 1/m$$
$$\kappa_b = \left(\frac{1}{\kappa_0} + \frac{b}{6}\left(\frac{1}{\kappa_6} - \frac{1}{\kappa_0}\right)\right)^{-1}, b \in 1...5$$
$$B = \{-\kappa_b\}_{b=0..6} \cup \{0\} \cup \{\kappa_b\}_{b=0..6} \quad (1)$$

The bin limits B are then sorted to ascending order.

Each representation vector x_{ir} is normalized and appended to form the final sample vector:

$$\mathbf{x}_i = (1\,\mathbf{x}_{i1}/\|\mathbf{x}_{i1}\|, ..., \mathbf{x}_{i6}/\|\mathbf{x}_{i6}\|) \in \mathbb{R}^{d+1},\, i \in 1...n \quad (2)$$

The maximum dimensionality of the problem $d = 6 \times (7+1+7-1) = 75$ is based on the histogram bin definition in Eq. 1, the choice of the number of the bins and concatenation of 6 histograms. Subtraction term -1 comes from the fact that the histogram is defined by the bin limits, not by the bin centers. Actually, all possible subsets of six grids were tried, five best combinations are listed in Table 2.

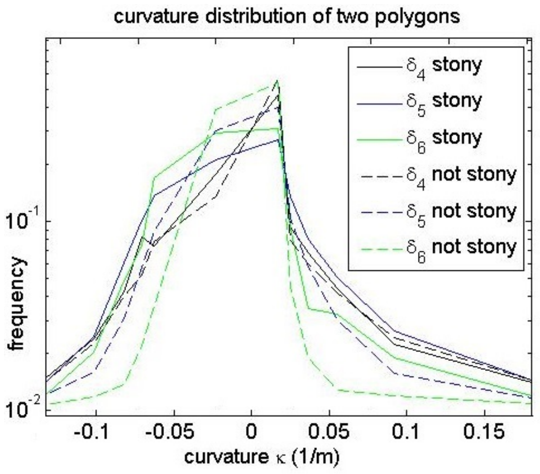


Fig. 3. Curvature histograms of the polygons 632 (non-stony, dotted line) and 7089 (stony, solid line) introduced at Fig. 4. Histograms are normalized and concatenated to form a sample vector.

Figure 3 depicts the distributions of two polygons on all grid sizes. The non-stony signal has distinctive amount of planar areas with approximately zero curvature. Stony areas have both positive and negative curvature extremes reducing the close to planar areas. The curvature range $\kappa \in [0.05, 0.1]\, 1/m$ represents typical ground contour not interfered with stones.

It is possible to formulate a similar interpolation for a bilinear case over one square of the grid following the same derivation procedure as in [13]. The triangular formulation recovers the data nearby the low point density areas more efficiently.

A. Comparison to DTM approach

Three different grids are visualized.

Conventional geographic information systems (GIS) have well-established DTM routines. With the available point density, their range reaches to grid size $\delta = 2\,m$. Tighter grids e.g. $\delta = 1\,m$ are numerically unstable.

For visual comparison, two test areas located West from Kemijärvi Finland were processed with the standard GIS tool ArcGIS 10.3 using ArcGIS/Topo-to-Raster.[2] method. Areas in question have different scales indicated by a line of $100\,m$ length. A visual comparison is in Fig. 4.

[2]ArcGIS Topo to Raster: http://resources.arcgis.com/en/help/main/10.1/index.html

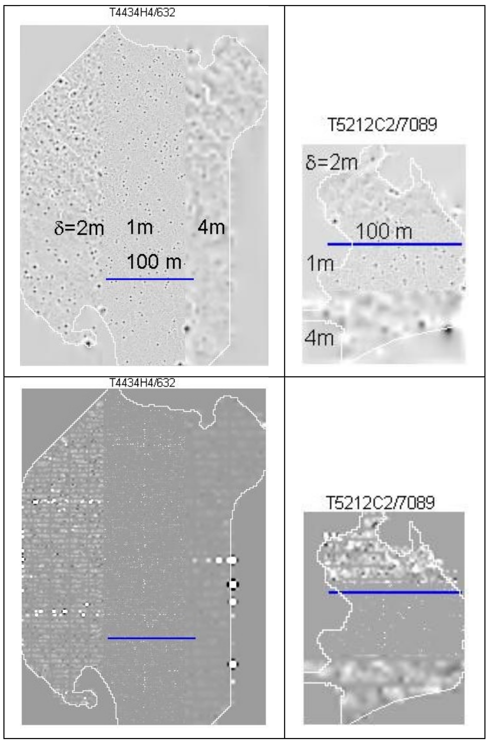


Fig. 4. Upper row: Laplace filtered DTM height relief generated by a GIS software. Lower row: curvatures from LAS cloud. Left column: A non-stony area 632. Right column: A stony area 7089. The scale indicated by a horizontal 100 m long line. The scales and three zones of different grid sizes are common to both columns.

Fig. 4 left column is a non-stony, and right column stony sample. The upper row is GIS topographic height model, the lower row is the local curvature rendered by our approach. The grid size $\delta = 1.0\,m$ is impossible to produce with standard terrain modeling tools. GIS results have been filtered by the 3×3 mean difference to make it visually comparable with curvatures. The GIS height signal detects stones at $4.0\,m$ range, but also picks up the basic contour signal. Both samples miss $r = 1.0\,m$ range stones.

The conventional GIS approach to stoniness classification would have the three steps (data $\rightarrow$ DTM $\rightarrow$ textures) presented in Sec. I-B. GIS tools have certain lack of control over the process which produces the DTM on which the further texture analysis must be founded. Also, the texture methods have trouble learning from the teaching polygons which have a slightly lax spatial relation to actual stony areas. It seems that the approach presented here exceeds the capabilities of many open source and commercial GIS software.

B. Method parameters

Strictly speaking, there are 15 parameters for weight function $g(.)$, 3 for histogram and 6 for grid constants. One parameter is a tolerance for close-to-planar case when Gaussian curvature $K \approx 0$. Preliminary numerical tests indicate that the weight parameterization seems to be rather independent problem to general classification performance. Same holds to the planarity limit and histogram definition. That leaves 6 grid constants as practically effective method parameters.

III. Results

A. Logistic regression

The label vector $y_i \in \{0,1\}, i = 1..n$ was acquired by field campaign done by a geology expert.

This is a qualitative response problem, so logistic regression was chosen to predict a label $\hat{y}$ from a given sample vector $\mathbf{x}$. The prediction coefficient $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$ is tuned by usual maximum likelihood approach to optimal value $\boldsymbol{\beta}^*_{T'}$ using a sample set $\{(\mathbf{x}_i, y_i)\}_{i \in T'}$, $T' \subset T = 1..n$ where T' is the training set and T is the full sample set:

$$f(\mathbf{x}_i, \boldsymbol{\beta}) = Pr(y_i = 1 \,|\, \mathbf{x}_i) = (1 + \exp\left[-\boldsymbol{\beta} \cdot \mathbf{x}_i >\right])^{-1}$$
$$\hat{y}^{(T')}(\mathbf{x}) = \begin{cases} 1 & f(\mathbf{x}, \boldsymbol{\beta}^*_{T'}) \geq 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The sample vectors $\mathbf{x}_i$ are defined in Eq. 2 and the label vector y_i in the text soon after Eq. 2. Vectors $\{x_i\}_{i \in \, T'}$ are standardized before solving the regression problem.

The area under curve (AUC) performance measure seems natural in this application area, where cost functions do exist but are not exactly known. The number of samples $n = 88$ is rather small due to the time constraints set for the field campaign for capturing the label data. [1] recommends in this case:

- to perform a leave-pair-out test over all possible positive-negative label pairs P, and
- to measure AUC by using the Heaviside function $H(.)$ for summation.

$$\begin{aligned} AUC &= \sum_{(i,j) \in P} H(\hat{y}_{ij} - \hat{y}_{ji})/|P| \\ \hat{y}_{ij} &= \hat{y}^{(T \setminus \{i,j\})}(\mathbf{x}_i) \text{ (learning without a pair } i,j) \\ P &= \{(i,j) \,|\, y_i = 1,\, y_j = 0,\, i,j \in J\} \\ H(\Delta\hat{y}) &= \big(1 + \text{sign}(\Delta\hat{y})\big)/2 \end{aligned}$$

B. Summary of results

There are 63 possible non-empty subsets of the six available grids, each leading to different vectorization of the samples. The best 4 subsets are listed in Table 2 ordered by the leave-pair-out AUC.

The publicly available DTM data from NLS has the grid size $\delta = 2\,m$. Grid combination $\{5, 6\}$ of this study could be based on the DTM data, it results in almost equal performance.

Table 2. Results with different combinations of grid sizes δ_m, $m = 1, .., 6$. The best result when using only the GIS range of grids is marked with (*).

Grids used, $m \in$	$\{1,4,5\}$	$\{5,6\}$	$\{1,3,5\}$	$\{1,4\}$
AUC	0.853	0.845*	0.837	0.837

It can be claimed that the Table 2 result (*) is an optimistic estimate for the DTM/GIS approach since information about normals is missing in public DTM. Normals should be geometrically estimated e.g. from 4 adjacent grid squares of grid constant $\delta = 2.5\,m$, and that size is close to numerically unreliable with this data density. Our method uses information from only one grid square at a time ans is numerically more stable.

IV. Conclusions

The prediction performance $AUC = 0.85$ is adequate for practical applications. The forest type of the test site is rather common and it is likely that almost similar ground sample densities can be expected especially in pine-woods elsewhere in Finland.

The method can be used as a generic stoniness indicator, but to use it independently off the context of this study, one should implement an approximation of the distributions of size and density of stones and boulders. Also, it is likely that the current method does not detect stones with radius $r < 2\,m$ well.

The method has linear computational efficiency due to space partitioning. The analysis speed is now c. $2\,km^2/h$ with the point density $0.8 - 1.2$ points$/m^2$, and using Python 3.4/Intel Core ir-3470. The speed gains are possible by avoiding the interpreted execution, by cheaper approximation of the curvature and by using distributed computation. The speed is practical already, though.

Texture analysis has many possible methods, some of which could perform well in this problem. Texture methods easily react to the general terrain type of the test site and are then less generalizable to new sites. In our opinion, one has to strive for independence from the terrain type and develop methods which detect stoniness decoupled from the environment. The more direct the method is, the better.

For some point cloud classification problems e.g. stoniness detection, texture based methods may not be the sole technique. Direct point cloud analysis methods have to be developed in parallel.

Acknowledgements

The feedback and constructive criticism of anonymous referees was received with high appreciation.

References

[1] A. Airola, T. Pahikkala, Willem Waegeman, Bernard De Baets and Tapio Salakoski. An experimental comparison of cross-validation techniques for estimating the area under the ROC curve. *Computational Statistics & Data Analysis*, 55:1824-1844, Apr 2010.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin and C. T. Silva. Point Set Surfaces. In *Proceedings of the conference on Visualization '01*, pages 21-28, 2001.

[3] ASPRS. LAS Specification Version 1.4 - R13. The American Society for Photogrammetry & Remote Sensing, Jul 2013.

[4] K. M. Brubaker, W. L. Myers, P. J. Drohan, D. A. Miller and E. W. Boyer. The Use of LiDAR Terrain Data in Characterizing Surface Roughness and Microtopography. *Applied and Environmental Soil Science*, 2013.

[5] M. Desbrun, M. Meyer, P. Schröder and Alan H. Barr. Discrete Differential-Geometry Operators in nD. pages 35-57, Springer-Verlag, 2003.

[6] J. S. Evans and A. T. Hudak. A multiscale curvature algorithm for classifying discrete return lidar in forested environments. *IEEE Transactions on Geoscience and Remote Sensing*, 45(4):1029-1038,2007.

[7] R. Hulík, M. Španěl, Z. Materna and P. Smrž, Continuous Plane Detection in Point-cloud Data Based on 3D Hough Transform. In *Journal of Visual Communication and Image Representation*, 25,(1):86-97, 2013.

[8] X. Lin and J. Zhang, Segmentation-Based Filtering of Airborne LiDAR Point Clouds by Progressive Densification of Terrain Segments. In *Remote Sensing*, 6(2):1294–1326,2014

[9] D. L. Page, Y. Sun, A. F. Koschan, J. Paik and M. A. Abidi. Normal Vector Voting: Crease Detection and Curvature Estimation on Large, Noisy Meshes. In *Graphical Models*, pages 199-229, 2002.

[10] M. Pauly, R. Keiser, M. Gross and Eth Zürich, Multi-scale Feature Extraction on Point-sampled Surfaces. In *Computer Graphics Forum*, 22(3):281-290, 2003.

[11] M. Pauly, R. Keiser, L. P. Kobbelt and M. Gross. Shape Modeling with Point-sampled Geometry. In *ACM Transactions on Graphics*, 22(3):641-650, Jul 2003.

[12] R. Sutinen, E. Hyvönen, M. Middleton and T. Ruskeeniemi. Airborne LiDAR detection of postglacial faults and Pulju moraine in Palojärvi, Finnish Lapland. In *Global and Planetary Change*, 115:24-32,2014.

[13] H. Theisel, C. Rössl, R. Zayer and Hans Peter Seidel. Normal Based Estimation of the Curvature Tensor for Triangular Meshes. In *PG04: Proceedings of the Computer Graphics and Applications*, 12th Pacific Conference on (PG2004), pages 288-297, IEEE Computer Society, 2004.

[14] G. Vosselman and L. Zhou. Detection of curbstones in airborne laser scanning data. In *Proceedings of Laser scanning '09*, pages 111-116, ISPRS 2009.

[15] J. F. Weishampel, J. N. Hightower, A. F. Chase, D. Z. Chase and R. A. Patrick. Detection and morphologic analysis of potential below-canopy cave openings in the karst landscape around the Maya polity of Caracol using airborne LiDAR. In *Journal of Cave and Karst Studies*, pages 187-196, 2011.

[16] M. Y. Yang and W. Förstner. Plane Detection in Point Cloud Data. Technical Report 01-2010, University of Bonn, 2010.

APPENDIX I
SURFACE APPROXIMATION

The input and output of the algorithm are:

- input: the set of nearby cloud points $Q_k \subset \mathbb{R}^3$ at grid location k, and the grid point c_k.
- output: local surface point p_k and normal $\mathbf{n}_k$ defined by a local planar approximation $\mathcal{P}_k$ of the surface $\mathcal{S}$. $\mathcal{P}_k = \{\mathbf{x} \in \mathbb{R}^3 \mid d(\mathbf{x}, p_k, \mathbf{n}_k) = 0\}$, $p_k \in \mathcal{P}_k$ and $\mathbf{n}_k$ is the plane normal. $d(.)$ is the geometric distance between the plane $\mathcal{P}$ and a cloud point q.

Optimization concerns parameter $\mathbf{w}_k^T = (z_k, u, v)$ where z_k is the local height associated to the grid point c_k, and u, v are orientation parameters of the local plane normal $\mathbf{n}_k$. The problem of finding the local plane $\mathcal{P}_k$ is now:

$$\mathbf{w}_k^* = \operatorname*{argmin}_{\mathbf{w}_k} f(\mathbf{w_k}) = \sum_{q \in Q_k} g[d(q, p_k, \mathbf{n}_k)] \tag{4}$$

$$d(q, p, \mathbf{n}) = (q - p) \cdot \mathbf{n} / \|\mathbf{n}\|_2$$

$$p_k = (c_k^T, z_k)^T$$

$$\mathbf{n}_k = (u, v, 1)^T$$

$$g(.) = \text{see Fig 5 and Table 3} \tag{5}$$

Iteration starts with $\mathbf{w}_{init} = (\min_{q \in Q_k}(q_z), 0, 0)^T$ and ends when the change of the target function $f(\mathbf{t})$ is small enough. The second order polynomial in Fig 5 depicts the normal least squares linear regression, which would often lead the surface approximation $\mathcal{S}$ to diverge from the best possible fit. The shape of $g(.)$ enables the iteration to converge from the initial $\mathbf{w}_{init}$ to a local ground alignment $\mathbf{w}^*$.

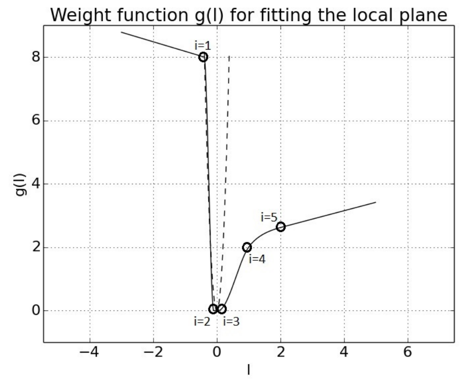


Fig. 5. The penalty function $w = g(l)$. l is the distance of point p from a local plane and w is the weight given to the distance l.

$g \in C^1(\mathbb{R}, \mathbb{R}^+)$ is a piecewise third order polynomial, which has the interval properties listed in Table 3. The table defines $g(.)$ uniquely inside the range $l \in [l_1, l_5]$. The linear extrapolation is applied outside that range. Within a range $l \in [l_i, l_{i+1}], i = 1..4$ and with end values $g_i = g(l_i), g'_i = \frac{dg}{dl}(l_i)$:

$$\Delta l_i = l_{i+1} - l_i$$

$$g(l) = (g_i, g_{i+1}, g'_i \Delta l_i, g'_{i+1} \Delta l_i) \cdot \mathbf{h}((l - l_i)/\Delta l_i) \tag{6}$$

$$\mathbf{h}(t) = (h_0(t), h_0(1-t), h_1(t), h_1(1-t))$$

$$h_0(t) = (2t - 3)t^2 + 1$$

$$h_1(t) = (t - 2)t^2 + t \tag{7}$$

The parameters of the distance weight function $g(.)$ are presented in Table 3.

Table 3. Weight function definition. Indices i refer to points at Fig. 5. Note that the orthogonal distance l_i has a physical dimension (m).

i	1	2	3	4	5
$l_i\,(m)$	-0.4	-0.1	0.04	1.0	2.0
$g(l_i)$	8.0	0.0	0.0	2.0	2.6
$\frac{dg}{dl}(l_i)$	-0.3	0.0	0.0	1.4	0.27

Parameters were chosen by the following criteria:

1) the plane $\mathcal{P}_k$ coincides with the local mode of the distances $d_q = d(q, p, \mathbf{n})$, $q \in Q_k$, $k \in I_1 \cup I_2$, where I_+ represents locations of a positive sample polygon and I_- of a negative sample
2) convergence from the initial value $\mathbf{w}_{init}$ quaranteed by linear extrapolation
3) acceptable convergence speed

Appendix II
Approximate local surface curvature

Curvature tensor approximation for triangulated surfaces is presented in detail in Ref. [5]. Ch. 3 of [13] has derivation of Gaussian curvature $K = \kappa_1\kappa_2$ and median curvature $H = (\kappa_1 + \kappa_2)/2$ where κ_l, $l \in \{1, 2\}$ are the two eigenvalues of the curvature tensor.

This is a reproduction of Ch. 3 of [13] adapted to the notation used in this paper. Triangle vertex points p_0, p_1, p_2 and the corresponding surface normals $\mathbf{n}_0, \mathbf{n}_1, \mathbf{n}_2$ come from the planar fits (see App. II) and are given in relation to a triangle $j \in J$. The following computation is indifferent to the orientation of vertices and the scaling of the normals. The triangle index is omitted.

Note that the successors $(.)'$ and precedessors $'(.)$ in the triangle indexing are presented by prime markings:
$0' = 1, 1' = 2, 2' = 0,\ '(i') = i$.

$$
\begin{array}{rll}
\mathbf{u}^T = & (u_0, u_1, u_2) & \text{barycentric coordinates} \\
 & u_0 + u_1 + u_2 = 1 & \\
\tilde{q}(\mathbf{u}) = & (p_0\, p_1\, p_2)\mathbf{u}, & \text{triangle surface point} \\
\tilde{\mathbf{n}}(\mathbf{u}) = & (\mathbf{n}_0\, \mathbf{n}_1\, \mathbf{n}_2)\mathbf{u} & \text{interpolated normal} \\
D = & |(\mathbf{n}_0\, \mathbf{n}_1\, \mathbf{n}_2)| & \text{scaling determinant} \\
\mathbf{r}_i = & p_{'i} - p_{i'} & \text{edge vectors} \\
\mathbf{m} = & \mathbf{r}_i \times \mathbf{r}_{i'} & \text{triangle normal (any } i\text{)} \\
\mathbf{h} = & \sum_{i=1}^{3} \mathbf{n}_i \times \mathbf{r}_i & \\
M(\mathbf{u}) = & \tilde{\mathbf{n}}(\mathbf{u}) \cdot \mathbf{m} &
\end{array}
$$

Then the two eigenvalues κ_1, κ_2 of the curvature tensor are:

$$
\begin{aligned}
K(\mathbf{u}) &= \frac{D}{||\tilde{\mathbf{n}}(\mathbf{u})||^2 M(\mathbf{u})} \\
H(\mathbf{u}) &= \frac{\tilde{\mathbf{n}}^0(\mathbf{u}) \cdot \mathbf{h}}{2\, M(\mathbf{u})} \\
\kappa_l(\mathbf{u}) &= H(\mathbf{u}) \pm \sqrt{H(\mathbf{u})^2 - K(\mathbf{u})}
\end{aligned} \tag{8}
$$

The following heuristics gives a single value, which represents the curvature:

$$
\kappa(\mathbf{u}) = \begin{cases} \operatorname{sign}(\kappa_1 \mathbf{u})\sqrt{|K(\mathbf{u})|}, & \operatorname{sign}(\kappa_1(\mathbf{u})) = \operatorname{sign}(\kappa_2(\mathbf{u})) \\ H(\mathbf{u}), & \text{otherwise} \end{cases} \tag{9}
$$

The curvature $\kappa(\mathbf{u})$ has to be evaluated at every corner: $\kappa_{jl} = \kappa(\mathbf{u}_l)$ where $\mathbf{u}_0 = (1, 0, 0), \mathbf{u}_1 = (0, 1, 0), \mathbf{u}_2 = (0, 0, 1)$. Note that this appendix considered the vertex curvatures of triangle $j \in J$.

The heuristics in Appendices I-II have not been completely optimized. Further improvements are still possible.

Publication R2

Video based Swimming Analysis for Fast Feedback

P. Nevalainen, A. Kauhanen,C. Raduly-Baka, M.-J. Laakso, J. Heikkonen, Video based Swimming Analysis for Fast Feedback. In: Maria De Marsico, Gabriella Sanniti di Baja, Ana L. N. Fred (Eds.), Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2016, 457–466, ScitePress, 2016.

Video based swimming analysis for fast feedback

Paavo Nevalainen[1], Antti Kauhanen[2], Csaba Raduly-Baka[1],
Mikko-Jussi Laakso[1], Jukka Heikkonen[1]
[1] *Department of Information Technology, University of Turku, FI-20014 TURKU, FINLAND*
[2] *Sport Academy of Turku region, Kaarinankatu 3 20500 TURKU, FINLAND*
ptneva@utu.fi,antti.kauhanen@turku.fi,{csaba.raduly-baka,milaak,jukhei}@utu.fi

Keywords: athletics, swimming, motion tracking, camera calibration, signal smoothing, movement cycle registration

Abstract: This paper proposes a digital camera based swimming analysis system for athletic use with a low budget. The recreational usage is possible during the analysis phase, and no alterations of the pool environment are needed. The system is of minimum complexity, has a real-time feedback mode, uses only underwater cameras, is flexible and can be installed in many types of public swimming pools. Possibly inaccurate camera placement poses no problem. Both commercially available and tailor made software were utilized for video signal collection and computational analysis and for providing a fast visual feedback for swimmers to improve the athletic performance. The small number of cameras with a narrow overlapping view makes the conventional stereo calibration inaccurate and a direct planar calibration method is proposed in this paper instead. The calibration method is presented and its accuracy is evaluated. The quick feedback is a key issue in improving the athletic performance. We have developed two indicators, which are easy to visualize. The first one is the swimming speed measured from the video signal by tracking a marker band at the waist of the swimmer, another one is the rudimentary swimming cycle analysis focusing to the regularity of the cycle.

1 INTRODUCTION

This paper describes the swimming analysis system being developed at the Impivaara public swimming center in Turku, Finland. Starting a new site for swimming analysis requires usually considerable resources and our economical approach with 5-7k€ budget for hardware and software licenses should be of interest to any swimming coach considering a basic computerized real-time feedback at a local site.

Budget reasons forced us to use 3 cameras only and video coverage of 18 m. Another major constraint was to get the system up and running with no special initial procedure and without disturbing recreational swimmers. The system can be expanded in the future by a fourth camera at the grey dot depicted in Fig 1. This setup will cover the whole 25 m pool length.

We use the video image series of the light-reflective marker on the waist of the swimmer to record the movement of the swimmer. The marker moves along the tracking plane, which resides 200 mm aside towards the cameras from the centerline of the swimming lane. The distance has been chosen so that it approximates the dimensions of the pelvis of an average-sized adult male and female. The real movement of the marker is naturally a non-planar one, but the planar approximation is a useful first step to simplify the swimmer movement analysis. The tracking plane of swimming lane 7 is depicted in Fig. 1. The tracking plane has c. 1 m depth at the shallow end and c. 2 m depth in the deep end of the pool. All the calibration measurements were constrained on this plane, and the calibration result is a geometric mapping from the image pixels to global coordinates of the tracking plane. Two lanes with numbers 7 and 8 were calibrated. Lane 8 is used occasionally for swimming analysis purposes, but camera views do not cover the whole length of the tracking plane as seen in Fig. 4.

The camera positions are constrained to windowsills at the sides of the pool at the depth of 560 mm. The image mapping was constructed directly in relation to the tracking plane, and this method does not require usual camera model, camera locations and orientations. The stereo calibration method proved inferior because of very limited overlap between cones of visibility, see Fig. 4.

The design emphasizes the possibility to a fast feedback. Thus there are features which are designed to operate in real-time during the session of athletic performance. There are also some features which are

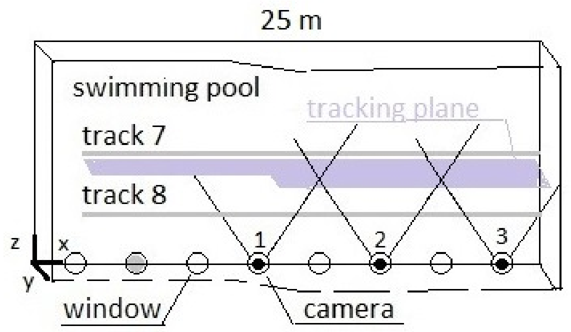


Figure 1: The general layout of the site seen from above. The tracking plane of lane 7 is emphasized.

based on the post-session phase. The forms of the feedback implemented are:

1. Real-time marker movement tracking embedded on the video.
2. Different performances presented side-by-side for visual comparison and evaluation. It can be real-time and post-session.
3. Stroke variation visualization, which is designed so that it can be monitored by the athlete in the pool. It can be real-time or post-session.
4. Geometric transformation of the video stream from pixels to global coordinates. It can be real-time or post-session.

The geometric mapping algorithm can project the raw video to a real-time 25 fps mono-color visualization on the tracking plane. The full quality color video has processing speed of c. 5 fps and cannot be performed in real-time. The geometric mapping will be a crucial part for a seamless swimmer-focused view after the swimmer detection has been implemented.

The marker tracking routine introduces both stochastic and algorithmic noise to the signal. After the pixel signal is transformed to global coordinates, the signal needs to be smoothed to eliminate the noise. The Kalman smoothing uses a basic dynamical model of a swimmer body. Smoothing requires the record of the whole performance as input and thus is a post-session step.

So far the coaching routine with verbal and video feedback has been established, but already the procedure is used on weekly basis and it requires no extra technical personnel on site.

The rest of the paper is organized as follows. Sec. 2 is a short presentation about the current research. Sec. 3 documents the architecture of the system, tracking and swimming cycle registration. Sec. 4 presents the used in-plane calibration method in detail, since this aspect is heavily dictated by the budget limitations yet opens possibilities for future research as well. Sec. 5 is about the real-time tracking visualization.

The swimmig cycle registration and comparison presented in Sec. 6 has been an important early facility for the coaching. The post-session analysis phase of Sec. 7 is one adaptation to the budget limitations. Sec. 8 summarizes the design choices made to achieve the real-time system response. Sec. 9 has conclusions and discussion about the possible future developments.

2 LITERATURE REVIEW

There are several approaches to swimming analysis. The oldest one is using wire. (Jean-Claude, 2003) reports about measuring the force in the wire while some object is dragged behind, another method is measuring the swimmer speed directly using the wire. The mechanical method is used especially to verify the video installments.

Video analysis is the dominant mode of performance analysis nowadays, see e.g. a review of the field in (Kirmizibayrak et al., 2011). A typical approach is:

- to produce the continuous video stream from multiple cameras
- and trace anchor points of the body (marked or nonmarked) and then
- combine the acquired information to a biomechanical or 3D visualization model.

There are several commercial tools available, many of them summarized in (Kirmizibayrak et al., 2011). Typical examples are Dartfish (Dartfish, 2015) and Sports Motion (Sportsmotion, 2015).

Wearable accelerometers are relatively new tools. These are developing smaller and lighter, also the operation time is increasing due the lower power requirements and increased battery capacity. (Dadashi et al., 2013) shows an arrangement with only one accelerometer to record the swimming performance over the full length of the pool. The data link is usually radio-linked in bursts like e.g. (James et al., 2011), or in the end of the performance, like in (Dadashi et al., 2013).

A typical large scale system design can be found from (Mullane et al., 2010). They provide an excellent analysis of what feedback should be provided real-time and what at post-session phase.

Swedish Center for Aquatic Studies has AIM (Athletes in Motion) system which can combine views from submerged and above-water cameras, see (Haner et al., 2015). The calibration process resembles our approach albeit they use striped poles while we use chessboard pattern. AIM has been developed by efforts of Chalmers and Lund Universities.

Chalmers University has a multiple accelerometer arrangement, which is coupled with a video analysis and a biomechanical model. Accelerometers can be placed by suction cups to various areas of the body. One study shows how a relatively low frequency still provides adequate biomechanical modeling (Siirtola et al., 2011).

Another video technique is the virtual camera technique, where a moving viewpoint is synthetized between two adjacent cameras. It is possible to interpolate the view between stationary spots like in (Makoto et al., 2002).

Head is a popular choice of strapping the athlet with a sensor device. One can wear a colored cap, or swimming glasses with an accelerometer, see (Pansiot et al., 2010).

Relatively new approach is the video analysis without markers (Ceseracciu, 2011). From athlete point of view it is much less intrusive and enables automation of the analysis process.

The trend in research seems to be towards 3D visualization and increasing usage of biomechanical models. Actual analysis is quite developed and remaining goals are at quick performance feedback and well visualized and conceptually simple performance measures.

As a summary, existing systems are well-developed and serve the coaching activities well. Often the implementation is rather involved requiring technical assistance, set-up times and high initial and running costs. Our aim was to produce a cheap and simple non-intrusive alternative with stable basis for further improvement.

3 SYSTEM DESCRIPTION

The system consists of:

- one 2-core 3.2 GHz 64 bits computer with. 2 TB of disc space
- 3 permanently placed 50 fps cameras at the side wall of the 25 m pool. The maximum image size is 750×2044. The camera placement is dictated by the construction of the pool.
- one movable extra camera for above-water usage and one movable underwater camera. Usage of the extra cameras is just for visual observation and verbal feedback only.
- movement marker and band at the hip of the swimmer.

The cameras and computer record and store over 50 fps high resolution digital video in uncompressed format. The image size is 750×2044 pixels. An individual pixel of the geometrically transformed image corresponds to 4.0×4.0 mm^2 and 2.3×2.3 mm^2 on lanes 7 and 8, correspondingly.

The uncompressed data from three cameras amounts to about 1GB for a 10 second clip. All cameras are synchronized so that they capture images at the same time. The time stamps are stored in the video files and they can be used in determining how to stitch the tracking results.

Marker tracking algorithm utilizes OpenCV package (Bradski, 2000). Camera calibration was done with self-developed software.

Process is divided to real-time and post-session phases. Figure 2 illustrates the various steps of the process. The recorded video is stored in a raw uncompressed file format specific to the camera manufacturer and is later accessed by the post-session phase.

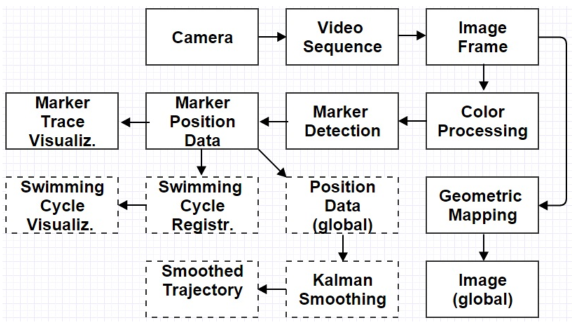


Figure 2: The processes and data flow. Post-session steps are indicated by the dashed outline.

4 CAMERA CALIBRATION

Camera calibration is a preliminary measurement process delivering either the camera model (mapping from pixels to normal vectors of the pinhole camera idealization) or the direct geometric mapping from pixels to global positions.

Three calibration methods, stereo-camera (Bouguet, 2008), mono-camera (Bouguet, 2008) and our own direct planar calibration were tested.

The stereo-calibration is an industry standard method since it is able to produce depth information (3D) and is not limited to the tracking plane G of Fig. 1. It calibrates the full camera view just like the mono-camera method. It also provides an early quality check in the form of relative camera positions depicted in Fig. 3. The position error was c. 85 mm even after the best possible calibration measurements described in (Bouguet, 2008).

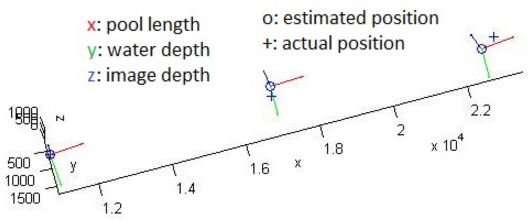


Figure 3: The camera positions from stereo-calibration.

Reason for the low accuracy is the difficult geometry of the camera positions dictated by the location of the windowsills of the pool, see Fig. 1. The amount of overlap of the camera views is only c. 22%, see Fig. 4 with refraction included, whereas the overlap ratio in a usual stereo calibration analysis is above 50%.

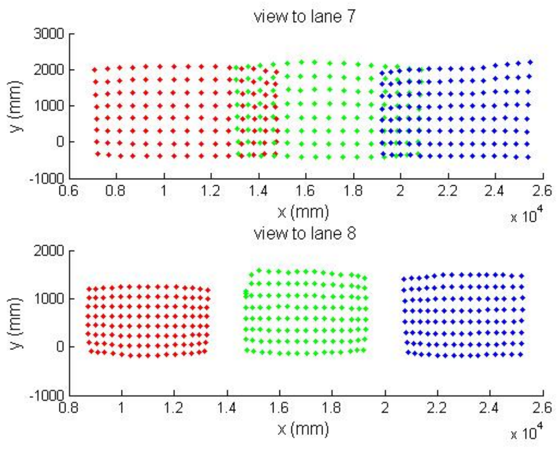


Figure 4: The area of visibility per each camera on the tracking plane. Colors (red/green/blue) correspond to cameras (1/2/3). Only c. 22% of the view is overlapping at lane 7. The lane 8 is closer to cameras, and there is no overlapping anymore.

The small overlap also rules out the homography approach described in (Chum et al., 2005) applied to all the sample points at the whole tracking plane at once. Otherwise that method would have been excellent, since it is able to use the expected location errors.

The mono-camera approach is very close to stereo-calibration, except the location and orientation of each camera is a separate subject of the matching process, when a fit is made to the tracking plane data. The mono-camera method is also close to the direct planar calibration presented in Sec. 4.1. The main difference is that the direct calibration requires no camera model (not even the camera location) and that the mapping from pixels to global locations can be arbitrarily chosen. The mono-camera calibration produces better mapping quality at the image borders than the direct planar calibration. The difference is aesthetic only, since the accurate zone of the direct calibration can be made large enough to accommodate all the swimmer motions. Also the mono-camera approach has been omitted from this presentation.

The stereo and mono calibrations were done with Matlab Camera Calibration Toolbox, see (Bouguet, 2008). The theory of the toolbox is given at (Zhang, 1999) and (Heikkilä and Silven, 1997).

The most accurate method was the direct planar calibration proposed in Sec. 4.1. This method can be categorized as an ad hoc approach answer to two constraints: sparsely placed camera array and potential for real-time video transformation. Nearest reference is (Luo et al., 2006), which uses a camera model and requires the co-planarity of the camera image plane and the tracking plane. Our method requires no camera model. The direct planar method is presented in the following.

4.1 Direct Plane Calibration

The geometric calibration was done for lanes 7 and 8 of ten available lanes. The calibration data for the direct method was gathered by floating a calibration chessboard along the surface at the tracking plane and recording its position at each picture, see Fig. 5. The chessboard had buoys at the top and weight at the bottom. The global position x_0 of the board was measured within 10 mm accuracy std.

To ease the presentation, some definitions are needed. An image $I = (P,J)$ of size $n \times m$ is a pair of set of pixels $p \in P = [1,n] \times [1,m] \subset \mathbb{Z}_+^2$ and their intensities $J = \{j(p) | p \in P\}$, where an individual pixel p has intensity $j(p)$. Images come in three varieties, source images $I_s = (P_s, J_s)$, geometrically transformed target images $I_t = (P_t, J_t)$, and calibration images.

The tracking plane $G = \{\mathbf{g} \in \mathbb{R}^3 \mid (\mathbf{g} - \mathbf{g}_G) \cdot \mathbf{n}_G = 0\}$ is defined by one insident point $\mathbf{g}_G = (0,0,z_0)^T$, where $z_0 = 5050$ mm in case of lane 7. The 200 mm aside of the center of the swimming lane. The normal vector $\mathbf{n}_G$ is a unit vector aligned with the global z axis. The marker is assumed to move along this tracking plane. Fig. 1 depicts the tracking plane G and the global coordinates x, y, z.

A chessboard corner pixel and its corresponding global positions form a measurement pair $(p, \mathbf{g}) \in U$. The calibration data set U is cumulated over all calibration images. One measurement image is depicted at the upper part of Fig. 5. The lower part shows the

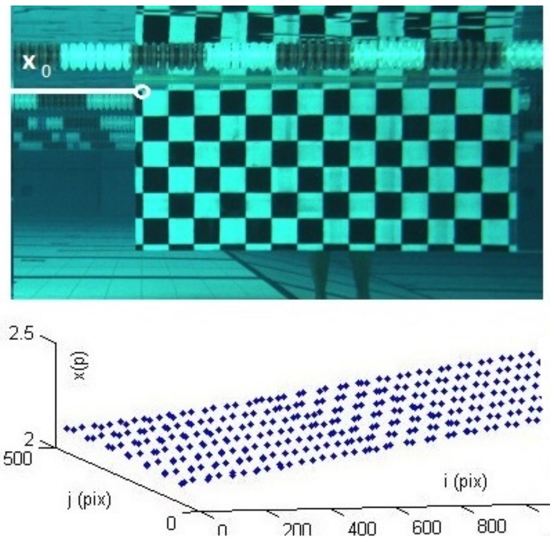


Figure 5: Direct calibration on the tracking plane. Above: an individual x_0 position of the calibration board at camera 2 view. Below: the cumulated observation set U of camera 1 consisting of corner point pixels p and global positions $\mathbf{g} \in \mathbb{R}^3$ from all images of lane 7 at camera 1. Only the x component of the global position $\mathbf{g}$ depicted.

measurement set U (only x component of $\mathbf{g}$ shown). The pixel samples of U cover only a part of the image pixels P whereas the end result of the direct plane calibration maps all pixels of the source image onto the tracking plane $P_s \rightarrow G$. In that sense this is an interpolation problem.

In the following, only the mapping of the x component is detailed. The y component has a similar treatment and is omitted for brevity. There would be some advantage to use a coupled mapping $p \rightarrow (x,y)$ e.g. a mono-camera model for interpolation but even this rudimentary approach with two independent mappings produced encouraging results.

A piecewise bilinear smoothing function:

$$x = f(p, \alpha_x^*) \tag{1}$$

with shape parameters $\alpha_x^* \in \mathbb{R}^d$ and automatic tiling heuristics is used to set a map $p \rightarrow x$ from pixels p to the global coordinate x. The shape parameter dimension $d \in \mathbb{N}_+$ varies case by case because of the implementation (D'Errico, 2006) but is in this application $d \approx 80$. The regularization parameter $\lambda_x \in \mathbb{R}_+$ controls the smoothness. The definition below uses functionals A_x for error penalty and B for non-smoothness penalty:

$$\begin{aligned} \alpha_x^* &= \underset{\alpha}{\operatorname{argmin}} A_x(f(.,\alpha)) + \lambda_x B(f(.,\alpha)) \\ A_x(f(.,\alpha)) &= \sum_{(p,\mathbf{g})\in U} (f(p,\alpha) - x)^2 \\ B(f(.,\alpha)) &= \sum_{(p,\mathbf{g})\in U} (\underset{q\in N_p}{\operatorname{mean}} f(q,\alpha) - f(p,\alpha))^2 \end{aligned} \tag{2}$$

In the above, N_p is the set of neighboring pixels of p at pixel radius $r = 2$. The function $f(.,.)$ is implemented as Matlab *gridfit.m* with 'bilinear' and 'laplacian' options, see (D'Errico, 2006). Values $\lambda_x = 120$, $\lambda_y = 180$ were chosen to keep the non-smoothness measure $B(.)/A(.)$ tolerable.

4.2 Precomputed Mapping

Let us combine Eqs. 2 and 1 for further treatment. The source image pixels p_s are mapped to tracking plane G by:

$$F_s(p_s) = (f(p_s,\alpha_x^*), f(p_s,\alpha_y^*), z_0) \in G, p_s \in P_s \tag{3}$$

The image of I_s on G is now $F_s(I_s)$. The target image I_t is mapped to tracking plane G by:

$$F_t(p_t) = \mathbf{g}_0 + \gamma \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} p_t^T \in G, p_t \in P_t \tag{4}$$

The target image pixel size $\gamma = 4.0$ mm for lane 7 and $\gamma = 2.3$ mm for lane 8. Image location $\mathbf{g}_0$ is specific for each camera view on each swimming line. Pixel $p = (i,j)$ has row and column indices as depicted in Fig. 6. It is now possible to interpolate the intensity value of p_t using Shepard interpolation of Eq. 6 at the tracking plane G.

First, some definitions. Let $|P_t|$ be the number of pixels in the image I_t and $M \in \mathbb{N}_+^{|P_t|\times 4}$ be a reference pixel matrix. One row $M_{p_t.}$ specified in Eq. 5, holds 4 reference pixels from P_s for a pixel $p_t \in P_t$. The nearest neighbor operator $\mathrm{NN}_k(x,X)$ selects k nearest neighbors of x from a set X. Sets are treated as vectors whenever there is a unique enumeration of the set. The location of p_t is $\mathbf{g}_t$. N_t is the set of 4 nearest neighbors of the sourcce image locations ar G.

$$\begin{aligned} \mathbf{g}_t &= F_t(p_t) \\ N_t &= \mathrm{NN}_k(\mathbf{g}_t, F_s(P_s)) \subset G, k = 4 \\ M_{p_t i} &= F_s^{-1}(N_{ti}) \subset P_s, i = 1...4 & (5) \\ W_{p_t i} &= \{s(\|\mathbf{g}_i - \mathbf{g}_t)\|) \mid \mathbf{g}_i \in N_t\}^0, i = 1..4 & (6) \\ s(r) &= 1/\max(r, 0.05\text{ mm}) \end{aligned}$$

where $W \in \mathbb{R}_+^{|P_t|\times 4}$ is a radial interpolation weight matrix with a correspondence to same indexing as matrix M. Function $s(r)$ is the radial weight used. The normalization in Eq. 6 happens by L_1 norm: $\mathbf{w}^0 = \mathbf{w}/\sum_i |w_i|$ for a general weight row $\mathbf{w}$.

Considering pixel intensities J_t and J_s as vectors indexed by pixels, the transformation becomes:

$$J_t(p_t) := \sum_{i=1}^{k} J_s(M_{p_t i})\, W_{p_t i} \tag{7}$$

By selecting $k = 1$ one gets the real-time transformation:

$$J_t(p_t) := \; J_s(M_{p_t 1}) \tag{8}$$

Eq. 8 corresponds to the Nearest Neighbor referencing which can be computed at 25 fps (measured with Matlab). The case $k = 4$ of Eq. 7 can be computed at 5 fps and thus is not usable in real-time. The quality of $k = 1$ case is adequate to a video stream, see Fig. 6. If quality par the original is required, one can use Eq. 7. The balance between speed and quality can be tuned further by choosing $k = 2$ or $k = 3$.

The geometric image mapping is efficient and simple, see Eqs. 8 and 7. The formulation used also makes it possible to combine the three separate video signals accurately to one single video. This feature will be implemented when an automated swimmer targeting is added to the system.

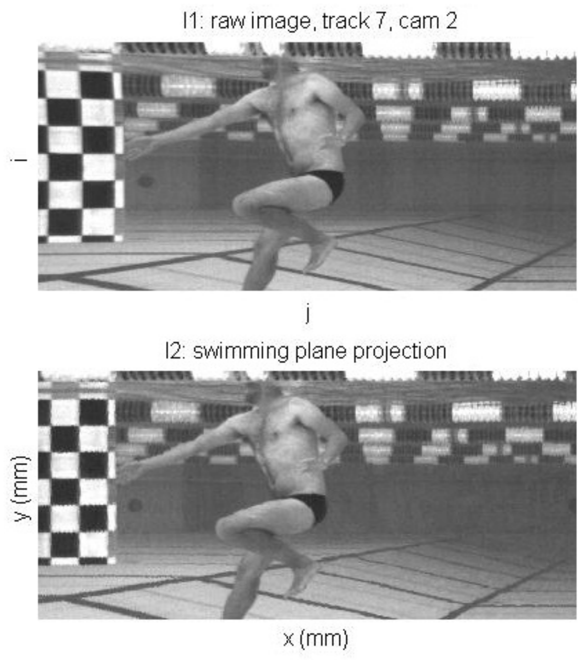


Figure 6: Quality of the fast mapping, a detail at the opposite wall. Above: the source image I_s with pixel coordinates i and j. Below: the target image I_t with global coordinates x and y.

4.3 Error analysis

The measurement points U in Fig. 5 are in approximate horizontal rows. There is c. 150 mm vertical gap between rows and c. 50 mm average horizontal distance between points. This requires the interpolant to have rather high penalty for non-smoothness.

The pixel detection was done with Matlab *detectCheckerBoard.m* function, theory of which is contained in (Zhang, 2000). The pixel detection error is $p \approx (1,1)$. The mechanical placement accuracy of the measured points $(p, \mathbf{g}) \in U$ is $\Delta\mathbf{g} \approx (10, 10, 10 + 0.01z)^T$ mm as an approximate std. The error is unbiased and the final accuracy of $F_s(p_s)$ is much better.

A pixel-wise geometric mapping error measure $e(p)$ is formulated by Eq. 9 and depicted at Fig. 7:

$$e(p) = \|\mathbf{g} - F_s(p)\|, \; (p, \mathbf{g}) \in U \tag{9}$$

Since the sample set U is of rather good quality and since the function F_s is rather smooth, the error stays almost constant even if the tuning of the shape parameters α_x, α_y in Eq. 2 is subjected to cross-validation over subsets of U. The error is largest in occasional points at the border and grows rapidly when extrapolating. The border areas are seldom occupied by a swimmer, though, and the problem is more of aesthetical nature. The border error can be eliminated in the future by applying a different interpolant instead of one in Eq. 3.

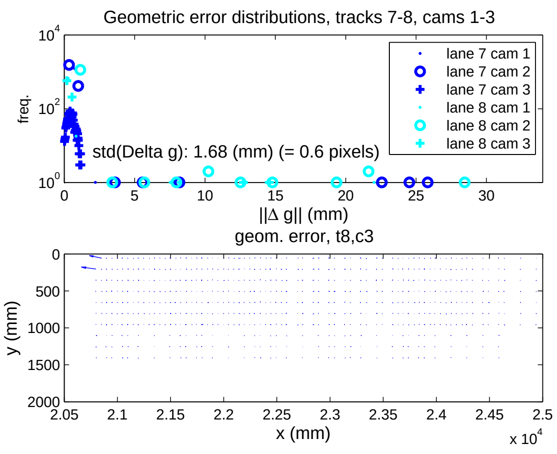


Figure 7: The geometric mapping error $e(p)$, $p \in P_s$ defined by Eq. 9. Large errors happen occasionally at the borders of the sampled area U.

5 REAL-TIME TRACKING

The swimmer tracking method is based on marker tracking using the blob tracking facilities of OpenCV

(Bradski, 2000). The marker is placed on a colored flexible band worn by the athlete in order to improve accuracy and reduce noise. The band is selected and installed so that it will not hinder the performance of the swimmer. The yellow color of the band is chosen to increase its visibility and identifiability in the environment. The band can be occluded by the swimmer's hand, or by bubbles in the water.

The color of the band is selected based on the color hue distribution of the environment. Tests on the site showed that the environment colors (water, swimmer, light, walls, etc) are between the interval $90^\circ - 270^\circ$, leaving the rest of the hue circle open. We selected a yellow color band. The choice leaves room for one or two extra markers, if needed in some future analysis.

The pixel trajectory of the marker is being visualized for the user in real-time, see Fig. 8. The pixel trace is then being mapped to global coordinates on the tracking plane using the geometric mapping described in Sec. 4. Tracking performs well with the current 50 fps speed allowing the real-time rendering.

The visualized points are referring to image pixel coordinates and are not suitable for speed analysis. The swimmer speed is calculated by converting these points using the geometric mapping described in Sec. 4.

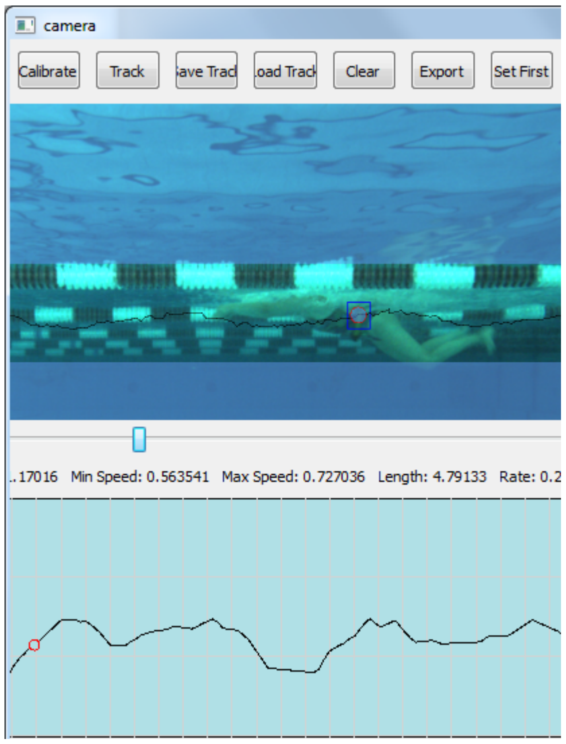
The real-time swimmer trace is provided as an overlay curve at the video area, see Figure 8. The current frame position is highlighted. The recorded session can be replayed immediately.

6 SWIMMING CYCLE REGISTRATION

A swimming cycle is the state history over a time interval during which the swimmer state returns relatively close to the initial state. The maximum performance requires rather monotonic strokes, yet the rhythm may vary based on metabolical optimum. Detecting the regularity of swimming strokes is of importance. Cycles seem to have a distinctive deceleration phase just before each hand stroke. This enables a simple cycle registration by finding a local spike heuristically in phase signal at times $t_i \in \mathbf{R}_+$, $i \in \mathbb{N}_+$. Using a peace-wise linear parameter τ:

$$t(\tau) = t_i(\tau - i) + t_{i+1}(i + 1 - \tau),\ i = \lfloor \tau \rfloor,\ \tau \in \mathbb{R}_+ \quad (10)$$

one can compare the shapes of two cycles i and j directly in a duration-invariant way on their own relative time scale $t_i + \tau$. Let us define the duration of a cycle i as $T_i = t_{i+1} - t_i$. Now, the dissimilarity d_{ij} between



..17016 Min Speed: 0.563541 Max Speed: 0.727036 Length: 4.79133 Rate: 0.2

Figure 8: Presenting the tracking results to the user. The emphasised square is for the user only and it does not correspond to the tracking plane. The track is based on pixel information for reducing response time.

two cycles i and j can be defined:

$$d_{ij} = [\int_0^1 (v_x(t(i+\tau)) - v_x(t(j+\tau)))^2 \, d\tau]^{1/2} + \\ + \lambda |T_i - T_j| \quad (11)$$

The horizontal velocity v_x in Eq. 11 is based on pixel information with a moving average smoothing, since we noticed the raw pixel signal is enough for the cycle detection. The last summand of Eq. 11 sets a weight on the duration difference between two strokes. The duration difference penalty is open to experimentation, currently we use value: $\lambda = 4$. We have used the vertical velocity $v_x(t)$ as the target signal for similarity analysis. The target signal could be also the horizontal velocity or a vector combination of both, in which case a vector norm should be used in Eq. 11.

The swimming cycle registration is a post-session process, which will be implemented as a real-time feature in the future. A similarity matrix is cumulated from last few strokes (last 5 in cases depicted in Fig. 9). The visualization is designed to be seen directly from the pool. The colors are scaled so that black is a serious deviation from allowed, white means identical strokes. Each stroke is compared to others and no judgement is made towards the quality of the swimming performance in general. The gray-scale used is an arbitrary choice at the moment.

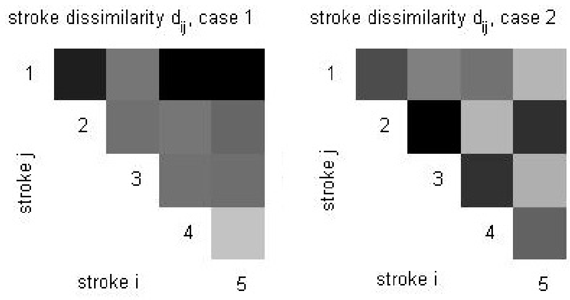


Figure 9: Swimming stroke regularity visualization. Rows and columns are individual strokes. White means zero difference and black a difference $d_{ij} = 0.2\ m/s$.

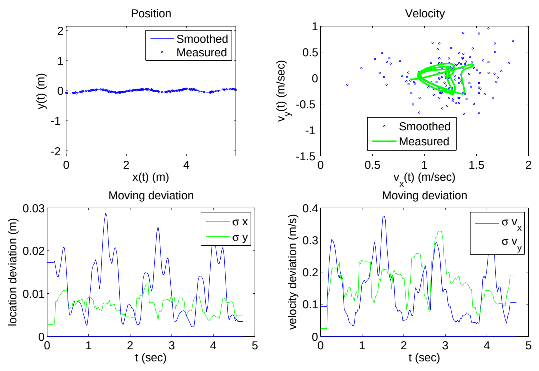


Figure 10: The smoothed position and velocity of the marker tracking. Above: Measured and smoothed signals. The current tracking brings large procedural noise component to velocity. Below: The moving deviation estimates. Only one camera view was used in this demonstration.

7 POST-SESSION PHASE

The post-session phase occurs when there is a pause in the athletic performance. First, the recorded video is stored on the hard disc. The trainer opens the video file with the tracking software and it is able to provide feedback to the coach and swimmer in a reasonable time (at most a few minutes).

The software allows the overlaying of multiple tracking results of different athletes. The trainer can use these data overlays to compare a trainee with a reference (trained) swimmer performance.

The speed graphs acquired by geometric transform of the original pixel trace are displayed in a separate area of the screen under the video frame. The graphs span the whole observed length.

A number of quantitative measures are displayed on the current swimmer performance, like average speed, distance, time, minimum and maximum speeds. A specific time period can be highlighted in the speed graph, to restrict the numeric display to measurements on this area.

There will be further experiments on visualizing various swimming characteristics. Preference will be given to the real-time feedback.

7.1 Kalman Smoothing

Kalman smoothing (J. Hartikainen and and Särkkä, 2011), is applied to pixel trace to get smoothed plots of the position and velocity components over time. Fig. 10 depicts the smoothed position and velocity history. Further swimming style analysis and movement registration operations can be based on this signal.

The marker observations are described as coming from a linear dynamical system of Eq. 12 with Gaussian noise $\mathbf{w} \sim N(\mathbf{0}, \lceil\sigma_x^2, \sigma_y^2\rfloor)$ as the driving force component. Our numerical choice was: $\sigma_x = 10$ N, $\sigma_y = 20$ N. Other constants of Eq. 12 are specified in Eq. 13:

$$\mathbf{m}\,\ddot{\mathbf{g}}(t) + \mathbf{c}\,\dot{\mathbf{g}}(t) + \mathbf{k}\,\mathbf{g} = \mathbf{w}(t) \qquad (12)$$

The system is further discretized to given non-regular observation times and transformed to a discrete-time linear dynamical model with Gaussian noise term, see details from (J. Hartikainen and and Särkkä, 2011).

The numerical values of the swimmer model of Eq. 12 are chosen for an average swimmer, and the dampening parameters approximate the observed speed resistance of swimming. The model will be improved later e.g. using Gaussian process formulation instead of Kalman, adding more biomechanical authentity to the model and physically interesting latent forces, see e.g. (Hartikainen et al., 2012):

$$\begin{aligned} \mathbf{m} &= \mathbf{I}_2 \times 60\ kg && \text{(mass)} \\ \mathbf{c} &= \lceil 11.7\ 16.3 \rfloor\ kg/s && \text{(dampening)} \quad (13) \\ \mathbf{k} &= \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \times 0.05\ kg/s^2 && \text{(spring)} \end{aligned}$$

The average error (std.) of the procedure is at the moment $\Delta x = 0.018$ m, $\Delta y = 0.008$ m, $\Delta v_x = 0.2$ m/s, $\Delta v_y = 0.2$ m/s for positions and velocities along x and y axes. Improvements will be made by applying a better tracking method less sensitive to bubbles.

Kalman smoothing is a post-processing task, too.

8 SYSTEM PERFORMANCE

A principal objective of the system is to provide immediate trainer feedback. To achieve that, the implementation is based on the following principles:

- The frames are processed and tracked in their uncalibrated shape (containing all camera distortions, refraction etc.). The physical meaning for the tracking signal can be attached only after the geometric mapping, which is a postprocessing step, see Fig. 2.
- The tracked area is limited manually, see the highlighted area in Fig. 8. Usually swimmers occupy only a narrow band on the screen. Limiting the tracking to this area of interest reduces computation significantly. At the moment, the tracking box (see Fig. 8) is selected manually, but there will be few prerecorded tracking areas for different swimming styles in the future.
- The dissimilarity measure of Eq. 11 is also based on the raw pixel information. The formula is computationally cheap, and it has to be evaluated on a separate processor once when swimmer passes a camera. Full real-time indicator will be implemented when a second computer and a monitor will be added to the system.
- The geometric mapping of video images uses reduced quality to deliver real-time performance.
- Seamless (combined from 3 cameras) geometrically accurate visualizations like video, stroke regularity indication and physical speed analysis are all left as a post-processing step. At this phase, the geometric mapping has been done to images and data already.

9 CONCLUSIONS

We have presented a simple video-based swimming analysis system which is easy to install, is of low cost and is simple to calibrate without any technical assistance. It can be installed to a wide variety of pool types. It is maintenance free and based on our experience so far, it can be operated by one person only. In ordinary use no technical assistance is needed.

The proposed system provides swimming speed analysis and instant visual feedback. The system is a good basis for further expansion e.g. with swimming gait analysis, biomechanical modeling etc.

The current system can be easily upgraded by a fourth camera at the location indicated by a grey circle in Fig. 1. The video monitoring would then span whole the pool length. A second video screen will be added in the future to serve the athletes better.

There are many off-the-shelf analysis systems with a wide spectrum of functionality available today. Usually these systems are much more complex and expensive than one presented here. Our choice was to implement the real-time pixel trace of the marker and swimming cycle regularity visualization.

The tracking system needs to be improved in the near future. At the moment it falls off-the-track too often, especially when a hand moment occludes the already lost marker.

The current system has been used by Finnish national swimming teams both on senior and junior level since autumn 2014. Automated tracking has made it possible to give faster and more accurate feedback to athletes. Thus it has been possible to test a large number of athletes in relatively short time during national team camps, when previously only a few of the top swimmers were able to get the service due to time investment required using the older version of the system. According to national team coach the system has been a major asset in developing technical skills of national team athletes. The findings have also been used in national coaches' education to provide insight into swimming performance.

The proposed direct planar calibration method used is aimed for efficient real-time video stream transformation. The efficiency is possible due to the restriction to 2D tracking plane projection only. There is potential for the same formulation to be generalized for 3D motion capture at the overlapping view zones (2×2 m at the current system, 3×2 m after one camera will be added). The proposed calibration method may be of use in other applications where conditions in camera placement rule the stereo-calibration out and where planar observations suffice.

The most important future goals are a reliable markerless tracking and implementing a record database with automated input from the site and a support for rudimentary searches and comparisons.

The swimming gait registration based on the profile shape of the body of the swimmer is a potential development.

Automated detection of different phases of the swimming performance remains the last goal. It is the hardest since there are a lot of different swimming styles each with somewhat differing phases, and female and male swimming costumes differ.

Acknowledgements

The project is a joint venture of University of Turku IT department and Sports Academy of Turku region and it has been funded by city of Turku, National Olympic Committee, Finnish Swimming Federation, Urheiluopistosäätiö and University of Turku.

REFERENCES

Bouguet, J. Y. (2008). Camera calibration toolbox for matlab. 3, 4

Bradski, G. (2000). Opencv. *Dr. Dobb's Journal of Software Tools*. 3, 7

Ceseracciu (2011). *New frontiers of markerless motion capture: application to swim biomechanics and gait analysis*. PhD thesis, Padova University. 3

Chum, O., Pajdla, T., and Sturm, P. (2005). The geometric error for homographies. *Comput. Vis. Image Underst.*, 97(1):86–102. 4

Dadashi, F., Millet, G., and Aminian, K. (2013). Inertial measurement unit and biomechanical analysis of swimming: an update. *Sportmedizin*, 61:21–26. 2

Dartfish (2011-2015). Dartfish video analysis tool. http://www.sportmanitoba.ca/page.php?id=116. 2

D'Errico, J. (2006). Surface fitting using gridfit. Technical report, MATLAB Central File Exchange. 5

Haner, S., Svärm, L., Ask, E., and Heyden, A. (2015). Joint under and over water calibration of a swimmer tracking system. In *Proceedings of the International Conference on Pattern Recognition Applications and Methods*, pages 142–149. ScitePress. 3

Hartikainen, J., Seppänen, M., and Särkkä, S. (2012). State-space inference for non-linear latent force models with application to satellite orbit prediction. *CoRR*. 8

Heikkilä, J. and Silven, O. (1997). A four-step camera calibration procedure with implicit image correction. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1106–1112. 4

J. Hartikainen and, A. S. and Särkkä, S. (2011). Optimal filtering with kalman filters and smoothers, a manual for the matlab toolbox ekf/ukf. Technical report, Dept. of Biomedical Eng. and Comp.Sci., Aalto University School of Science. 8

James, D. A., Burkett, B., and Thiel, D. V. (2011). An unobtrusive swimming monitoring system for recreational and elite performance monitoring. In *Procedia Engineering, 5th Asia-Pacific Congress on Sports Technology (APCST)*, volume 13, pages 113–119. 2

Jean-Claude, C., editor (2003). *Biomechanics and Medicine in Swimming IX*. IXth International World Symposium on Biomechanics and Medicine in Swimming, Université de Saint-Etienne. 2

Kirmizibayrak, J., Honorio, J., Xiaolong, J., Mark, R., and Hahn, J. K. (2011). Digital analysis and visualization of swimming motion. *The International Journal of Virtual Reality*, 10(3):9–16. 2

Luo, H., Zhu, L., and Ding, H. (2006). Camera calibration with coplanar calibration board near parallel to the imaging plane. *Sensors and Actuators A: Physical*, 132:480486. 4

Makoto, H. S., Kimura, M., Yaguchi, S., and Inamoto, N. (2002). View interpolation of multiple cameras based on projective geometry. In *In: International Workshop on Pattern Recognition and Understanding for Visual Information*. 3

Mullane, S. L., Justham, L. M., West, A. A., and Conway, P. P. (2010). Design of an end-user centric information interface from data-rich. In *Procedia Engineering*, volume 2, pages 2713–2719. 8th Conference of the International Sports Engineering Association (ISEA). 2

Pansiot, J., Lo, B., and Guang-Zhong, Y. (2010). Swimming stroke kinematic analysis with bsn. In *Body Sensor Networks (BSN), 2010 International Conference on*, pages 153–158. 3

Siirtola, P., Laurinen, P., Roning, J., and Kinnunen, H. (2011). Efficient accelerometer-based swimming exercise tracking. In *IEEE Symp. on Computational Intelligence and Data Mining (CIDM)*, pages 156–161. IEEE. 3

Sportsmotion (2011-2015). motion analysis system. http://www.sportsmotion.com/. 2

Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. In *in ICCV*, pages 666–673. 4

Zhang, Z. (2000). A flexible new technique for camera calibration. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, page 13301334. 6

Publication P3

Detecting terrain stoniness from airborne laser scanning data

P. Nevalainen, M. Middleton, R. Sutinen, J. Heikkonen, and T. Pahikkala. Detecting terrain stoniness from airborne laser scanning data. IPTA 2015 The 5th International Conference on Image Processing Theory, Tools and Applications, 11/2015.

remote sensing

Article

Detecting Terrain Stoniness From Airborne Laser Scanning Data †

Paavo Nevalainen [1,*], Maarit Middleton [2], Raimo Sutinen [2], Jukka Heikkonen [1] and Tapio Pahikkala [1]

[1] Department of Information Technology, University of Turku, FI-20014 Turku, Finland; jukhei@utu.fi (J.H.); Tapio.Pahikkala@utu.fi (T.P.)

[2] Geological Survey of Finland, P.O. Box 77, Lähteentie 2, 96101 Rovaniemi, Finland; Maarit.Middleton@gtk.fi (M.M.); Raimo.Sutinen@gtk.fi (R.S.)

* Correspondence: ptneva@utu.fi; Tel.: +358-40-351-8236

† This paper is an extended version of our paper published in Paavo Nevalainen, Ilkka Kaate, Tapio Pahikkala, Raimo Sutinen, Maarit Middleton, and Jukka Heikkonen. Detecting stony areas based on ground surface curvature distribution. In The 5th International Conference on Image Processing Theory, Tools and Applications, 2015.

Academic Editors: Jie Shan, Juha Hyyppä, Lars T. Waser and Prasad S. Thenkabail

Received: 29 June 2016; Accepted: 17 August 2016; Published: 31 August 2016

Abstract: Three methods to estimate the presence of ground surface stones from publicly available Airborne Laser Scanning (ALS) point clouds are presented. The first method approximates the local curvature by local linear multi-scale fitting, and the second method uses Discrete-Differential Gaussian curvature based on the ground surface triangulation. The third baseline method applies Laplace filtering to Digital Elevation Model (DEM) in a 2 m regular grid data. All methods produce an approximate Gaussian curvature distribution which is then vectorized and classified by logistic regression. Two training data sets consisted of 88 and 674 polygons of mass-flow deposits, respectively. The locality of the polygon samples is a sparse canopy boreal forest, where the density of ALS ground returns is sufficiently high to reveal information about terrain micro-topography. The surface stoniness of each polygon sample was categorized for supervised learning by expert observation on the site. The leave-pair-out (L2O) cross-validation of the local linear fit method results in the area under curve $AUC = 0.74$ and $AUC = 0.85$ on two data sets, respectively. This performance can be expected to suit real world applications such as detecting coarse-grained sediments for infrastructure construction. A wall-to-wall predictor based on the study was demonstrated.

Keywords: aerial laser scan; point cloud; digital elevation model; logistic regression; stoniness; natural resources; micro-topography; Gaussian curvature

1. Introduction

There is an increased attention towards classification of the small scale patterns of terrain surface. Recognition of micro-topography may help in arctic infrastructure planning [1], terrain trafficability prediction [2], in hydraulic modeling [3], and in detecting geomorphologic features like in [3,4], and terrain analysis and modelling.

In Finland, a nationwide airborne light detection and ranging (LiDAR) mapping program has provided the means for detecting ground objects with the ground return density $\rho \approx 0.8\ \mathrm{m}^{-2}$. Since one needs at least one point per stone, and to define the stone radius one needs at least 4 points per stone, this leads to an absolute theoretical detection limit of stone radius $r_{min} = 0.6 \ldots 1.2$ m. The real limit is naturally somewhat higher. The actual stone sizes fall into this critical range (as discussed in Section 2.2) making the stoniness detection a difficult problem.

One aspect of the ground surface is the presence of stones and boulders, which can be characterized by the stone coverage and by stone size distribution. Mass-flow deposits are recognized by irregular distribution of boulders and stones on their surface. Mass-flow deposits may have regional significance in aggregate production if they occur in fields as they do in the Kemijärvi region in Finland. Mass-flow sediments are often moderately sorted sediments with low fine grained fraction (clay and silt content, <0.006 mm) being less than 12 % [5]. In addition they contain boulders and stones in their sediments which may be crushed for aggregates. Therefore, they are potential aggregates for infrastructure construction. Mass-flow deposits can be detected in a two-step process: First candidate polygons are found by analyzing geomorphological features in a process which can be automated, then surface stone detection based on airborne LiDAR data is performed to limit the set of candidates. Various other geomorphological features like paleolandslides [6], fluvial point bars [7], neotectonic faults [3] and Pulju moraines [8] in Finland and several other types of glacial landforms elsewhere (see summary in [9]) have already been mapped using LiDAR data.

The intent of this paper is to document various methods, which analyze airborne laser scanning data (ALS) or digital elevation model (DEM) to detect stony areas. Our hypothesis is that a direct approach may be able to detect a signal of a target feature like stoniness better than methods using DEM. This is because DEM is a general smoothed representation of the ground surface for generic purposes [10]. This paper focuses on binary classification of the stoniness of sample areas. The approach results in a classifier, which is subjected to 20 m × 20 m point cloud patches to produce a binary mask about stoniness covering whole Northern Finland. Stoniness is just one example of micro-topographic features, which could be detected from public ALS data. Even the positive samples of the data sets focus on stony mass-flow deposits, algorithms are developed for general stoniness detection, which can be later targeted to various specific purposes depending on the available teaching data. It is our hope that the research community finds our results and methods useful in the future.

This paper is an expansion of [11], which studied only one polygon set *data2014* using curvature estimation based on local linear fit (LLC). In comparison to [11], this paper uses an additional data set *data2015*, additional public DEM data format and two additional methods: local curvature estimation based on triangulated surface model computed from LiDAR (LTC) and Laplace filtering of a DEM grid (DEC). LTC uses triangulated irregular network (TIN) produced by a solid angle filtering (SAF). An overview of the relation of computational methods and various data formats can be seen in Figure 1.

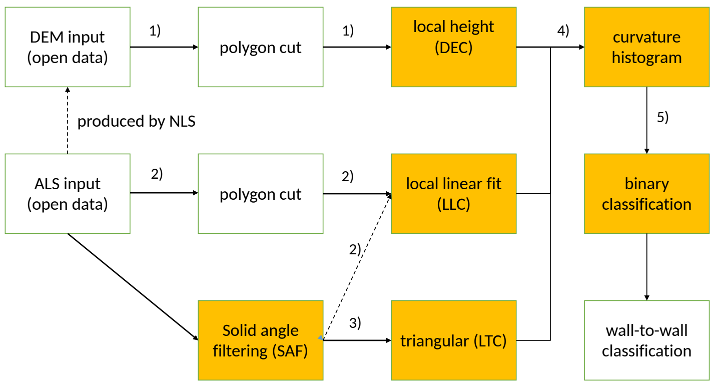


Figure 1. The process flow, methods covered in this paper are highlighted. Data formats: (1) 2 m raster; (2) point cloud; (3) task-specific TIN model; (4) curvature value sets; (5) sample vectors. LLC can optionally use either original point cloud (2) or vertex points (3) produced by SAF TIN model. Wall-to-wall classification is a possibility provided by the resulting binary classifier.

The structure of the paper is as follows: a summary of current research applicable to micro-topographic feature detection is in Section 1. Data sets and data formats are explained in Section 2.2. The solid angle filtering (SAF) can be used by two presented methods, it has been described in Section 2.4. Three methods are documented in Sections 2.5–2.7. Methods are compared in Section 3. Possible improvements are discussed in Section 4 and further application areas considered in Section 5.

Current Research

A good presentation of general-purpose ALS classification is in [12]. Our work relates to some of the contour and TIN -based ground filtering algorithms mentioned in [13], since all of our methods either directly or indirectly use or produce a tailor-made ground model. Methods described in [13] are usually more generic accommodating to infrastructure signatures etc. It is possible, that methods described in our paper have to be combined with existing generic ground model algorithms, where an assembly of methods would use e.g., a voting arrangement at the approximity of constructed environment.

Solid angle filtering (SAF) in Section 2.4 resembles the despike algorithm presented in [14]. Two problems are mentioned in [14]: Unnecessary corner removals (rounding off the vertices of e.g., block-like structures) and effects of negative blunders (false points dramatically below the real surface level). Our routine was specifically designed to eliminate these problems. SAF can also be used in canopy removal. An interesting new technique in limiting the ground return points is min-cut based segmentation of k-nearest neighbors graph (k-NNG) [15]. The graph is fast to compute with space partitioning, and it could have served as a basis for stoniness analysis directly e.g., by fast local principal components analysis (PCA) and local normal estimation with vector voting procedure, as in [16]. The literature focuses mostly on laser clouds of technological environment, where the problem of eliminating the canopy (noise) and finding the ground returns (a smooth technical surface) are not combined. Our experiments with local normal approximation and vector voting were inferior to results presented in this paper. There is great potential in local analysis based on k-NNG, though.

There seems to be no research concerning the application of ALS data to stoniness detection in forest areas. Usually target areas have no tree cover [17], objects are elongated (walls, ditches, archaeological road tracks, etc.) [17,18] and often multi-source data like photogrammetry or wide-spectrum tools are used. curbstones which separate the pavement and road in [18]. Their data has the sample density $\rho = 5/\,\mathrm{m}^2$ which produces geometric error of size 0.3 m which is larger than the observed shapes (curbstones) and thus not practical. Effects of foliage and woody debris are discussed in [19]. They mention that even a high-density ALS campaign is not able to get a dense sampling of the ground surface in a non-boreal forest (Pennsylvania, U.S.). They reported ground return ratio is 40% with the ground sample density $\rho = \ 4/\,\mathrm{m}^2$, which is much higher than $\rho \approx 0.8/\,\mathrm{m}^2$ in our study. The distribution of the local ground sample density was not reported in [19] but is probably much higher than in our case.

DEM in Figure 1) is a standard data type used by geographic information systems (GIS). Many implementations and heuristics exist (see e.g., [20]) to form DEM from ALS format *.las* defined by [21]. Usually, the smallest raster grid size is dictated by the sample density and in this case DEM grid size $\delta = 2\,\mathrm{m}$ is possible, and $\delta = 1\,\mathrm{m}$ already suffers from numerical instability and noise.

A rare reference to DEM based detection of a relatively small local ground feature (cave openings) in forest circumstances is presented in [22]. In that paper the target usually is at least partially exposed from canopy and the cave opening is more than 5 m in diameter. On the other hand, the forest canopy was denser than at our site in general. Another application is detecting karst depressions, where slope histograms [23] and local sink depth [24] were used to detect karst depressions. There are similarities with our study, e.g., application of several computational steps and tuning of critical parameters (e.g., the depression depth limit in [23]), although the horizontal micro-topology feature size is much larger than in our study (diameter of doline depressions is 10–200 m vs. 1.5–6 m diameter of stones in our study). The vertical height differences are at the same range, 0.5–1.5 m in our study and in [23,24], though. A similar study of [25] uses higher density LiDAR data with $\rho = 30\,\mathrm{m}^{-2}$ to detect karst depressions of size 26 m and more. The vertical height difference (depth) was considerably larger than in in [23,24]. The high density point cloud and a carefully designed multi-step process results in quantitative analysis of sinkholes in [25], unlike in our study, where the stoniness likelihood of a binary classifier is the only output.

One reference [19] lists several alternative LiDAR based DEM features, which could be used in stone detection, too. These include fractal dimension, curvature eigenvectors, and analyzing variograms generated locally over multiple scales. Some of the features are common in GIS software, but most should be implemented for stoniness detection.

Hough method adapted to finding hemispherical objects is considerably slower than previous ones, although there is a recent publication about various possible optimizations, see e.g., [26]. These optimizations are mainly about better spatial partitioning.

Minimum description length (MDL) is presented in [27] with an application to detect planes from the point cloud. The approach is very basic, but can be modified to detect spherical gaps rather easily. MDL formalism can provide a choice between two hypotheses: *a plain spot/a spot with a stone*. Currently, there is no cloud point set with individual stones tagged to train a method based on MDL. MDL formalism could have been used without such an annotated data set, but we left this approach for further study. In addition, probably at least 4..8 returns per stone is needed and thus a higher ground return density than is currently available.

This paper presents two methods based on ALS data and one method using DEM and acting as a baseline method. The DEM method was designed according to the following considerations: It has to be easy to integrate to GIS and it would start from a DEM raster file, then generate one or many texture features for the segmentation phase. The possible texture features for this approach are the following:

- local height difference, see Laplace filtering Section 2.7. This feature was chosen as the baseline method since it is a typical and straightforward GIS technique for a problem like stoniness detection.
- various roughness measures, e.g., rugosity (related trigonometrically to the average slope), local curvature, standard deviation of slope, standard deviation of curvature, mount leveling metric (opposite to a pit fill metric mentioned in [19]).
- multiscale curvature presented in [28]. It is used for dividing the point cloud to ground and non-ground returns, but could be modified to bring both texture information and curvature distribution information. The latter could then be used for the stoninesss prediction like in this study. The methods, possibly excluding interpolation based on TIN, seem to be numerically more costly than our approach.

Possible GIS -integrated texture segmentation methods would be heavily influenced on the choices made above. Most of the features listed are standard tools in GIS systems or can be implemented by minimal coding. An example is application of the so called mount leveling metric to stoniness detection, which would require negating the height parameter at one procedure.

Terrain roughness studied in [19] is a concept which is close to stoniness. Authors mention that the point density increase from $\rho = 0.7/\mathrm{m}^2$ to $\rho = 10/\mathrm{m}^2$ did not improve the terrain roughness observations considerably. This is understandable since the vertical error of the surface signal is at the same range as the average nearest point distance of the latter data set. The paper states that algorithms producing the terrain roughness feature have importance to success. This led us to experiment with various new algorithms.

Point cloud features based on neighborhoods of variable size are experimented with in [29]. Many texture recognition problems are sensitive to the raster scale used, thus we tested a combination of many scales, too. According to [16], curvature estimation on triangulated surfaces can be divided to three main approaches:

- surface fitting methods: a parametric surface is fitted to data. Our local linear fit LLC falls on this category, yet does not necessarily require triangularization as a preliminary step.
- total curvature methods: curvature approximant is derived as a function of location. Our local triangular curvature LTC is of this category of methods.
- curve fitting methods.

LLC has a performance bottleneck in local linear fit procedure described in Section 2.5. This problem has been addressed recently in [30], where an algebraic-symbolic method is used to solve a set of total least squares problems with Gaussian error distribution in a parallelizable and efficient way. That method would require modification and experimentations with e.g., a gamma distributed error term due to asymmetric vegetation and canopy returns.

The vector voting method presented in [16] decreases noise and achieves good approximative surface normals for symmetrically noisy data sets of point clouds of technological targets. Our target cloud has asymmetrical noise (vegetation returns are always above the ground), and returns under the ground (e.g., reflection errors) are extremely rare. Usually vector voting methods are used in image processing. They are based on triangular neighborhood and any similarity measure between vertices, focusing signal to fewer points and making it sometimes easier to detect. Neighborhood voting possibilities are being discussed int Section 4.

General references of available curvature tensor approximation methods in case of triangulated surfaces are [31,32]. A derivation of Gaussian curvature κ_G and mean curvature κ_H is in [33]:

$$\kappa_G = \kappa_1 \kappa_2 \quad (1)$$

$$\kappa_H = (\kappa_1 + \kappa_2)/2, \quad (2)$$

where κ_l, $l \in \{1, 2\}$ are the two eigenvalues of the curvature tensor. Perhaps the best theoretical overview of general concepts involved in curvature approximation on discrete surfaces based on discrete differential geometry (DDG) is [34].

We experimented with methods which can produce both mean and Gaussian curvatures, giving access to curvature eigenvalues and eigenvectors. Our experiments failed since the mean curvature κ_H seems to be very noise-sensitive to compute and would require a special noise filtering post-processing step. Difficulties in estimating the mean curvature from a noisy data have been widely noted, see e.g., [29].

In comparison to previous references, this paper is an independent study based on the following facts: point cloud density is low relative to the field objects of interest (stones), ratio of ground returns amongst the point cloud is high providing relatively even coverage of the ground, a direct approach without texture methods based on regular grids was preferred, individual stones are not tagged in the test data, and the methods are for a single focused application. Furthermore, we wanted to avoid complexities of segmentation-based filtering described in [35] and the method parameters had to be tunable by cross-validation approach.

2. Materials and Methods

Test data is presented in Section 2.2. It is available online, details are at the end of this paper.

Figure 1 presents the process flow of stone detection. Two data sources at left are introduced in Section 2.2, tested methods (DEC, LLC, LTC) are detailed in Sections 2.5–2.7. The vectorization of samples varies depending on the method in question, details can be found in Section 2.8. The solid angle filtering SAF of Section 2.4 is a necessary preprocessing step for LTC, but could be used also before LLC for computational gain.

2.1. Study Area

The study area is a rectangle of 1080 km^2 located in the Kemijärvi municipality, in Finnish Lapland, see Figure 2. The 675 sample polygons cover approx. 10.7 km^2 of the area. Mass-flow sediment fields such as the Kemijärvi field, have regional significance for aggregate production as there is an abundance of closely spaced mass-flow formations within a relatively short distance from the road network.

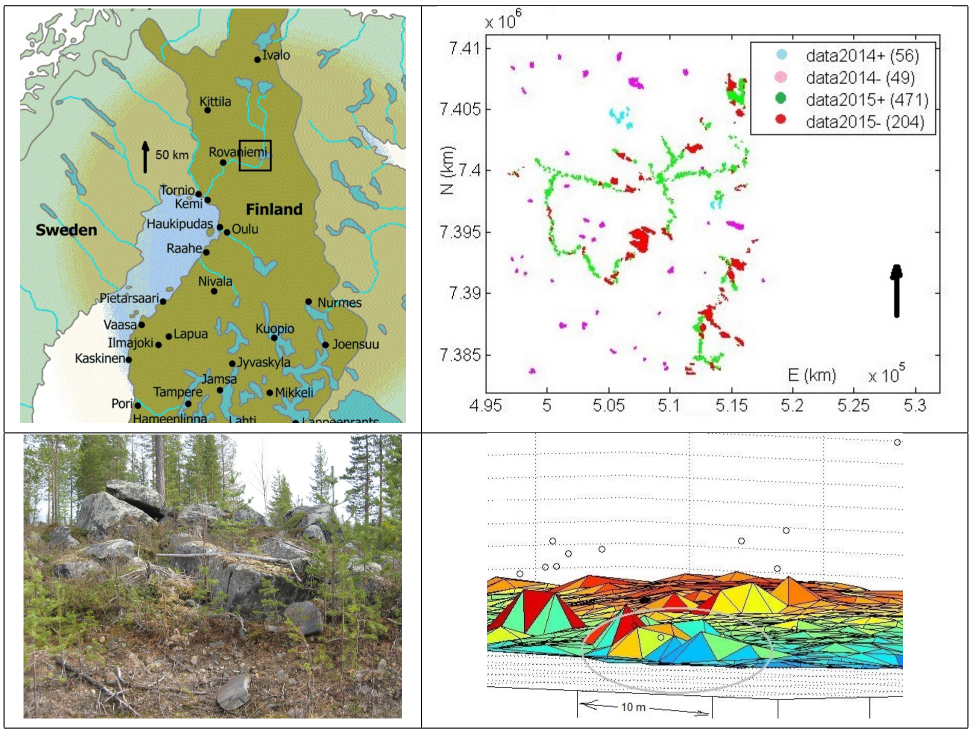


Figure 2. Upper left: The site near Kemijärvi Finland. The research area covered by 120 open data *.las* files covering 1080 km^2. **Upper right**: the relative location of sample polygons. Amount of sample sets in parenthesis. **Lower left**: A view of a sample site in boreal forest. **Lower right**: approximately the same view as at lower left after solid angle filtering (see Section 2.4) of the point cloud. The stone formation has been circled. Location is at UTM map T5212C3, polygon 11240.

2.2. Materials

Table 1 gives a short summary of the data sets: the first set *data2014* is rather small with positive samples occupied by large boulders. The second set *data2015* has an imbalance of many positive samples with smaller stones vs. fewer negative samples. Data sets are depicted in Figure 2. The acquisition of data sets differ: the classification of *data2014* was based on cumulated field photographs and the land survey annotations of the general topographic map (stone landmarks). There is no stone size distribution data available for *data2014*, though. The set *data2015* was classified and the approximative stone size and coverage statistics recorded by a geology expert. The second data set seems to present more difficult classification task—the areas are more varied and stone size probably smaller than in the first set. The advantage of having two data sets of different origin is that the resilience and generality of the methods can be better asserted.

Table 1. Some characteristics of the two data sets.

Data Set	Stony Samples	Area km^2	Non-Stony Samples	Area km^2	Acquisition
data2014	56	1.7	49	1.7	cumulated observations
data2015	471	4.7	204	6.0	field campaign

The data preprocessing consists of the following three steps:

1. All hummocky landforms (i.e., hills) with a convex topographic form were delineated from the ALS derived digital elevation model and its tilt derivative with an Object-Based Image Analysis algorithm developed in eCognition software, see [1]. This step produced *data2014* and *data2015* polygon sets (see Table 1 and Figure 2).
2. A 10 m × 10 m space partitioning grid was used to cut both the point cloud (ALS) and DEM to polygon samples.
3. Point cloud was cut to 2 m height from initial approximate ground level. The mode of heights in 2 m × 2 m partitions was used as the ground level.

ALS LiDAR data was produced by National Land Survey (NLS) (NLS laser data: http://www.maanmittauslaitos.fi/en/maps-5) in fall 2012 with a Leica ALS50-II laser scanner (Leica Geosystems, St. Gallen, Switzerland), the flight altitude was 2000 m. Last-return data has approx. 0.09–0.1 m vertical ground resolution and average footprint of 0.6 m. ALS data has several additional information fields per cloud point, see e.g., [21]. We used only x-y-z components of the data. Approximately 25% of the data are canopy returns, the rest is ground returns. Reflection errors causing outlier points occur approximately once per 0.5×10^6 returns.

DEM data is 2 m regular grid data available from NLS. It is nationwide data aimed for general purposes (geoengineering, construction industry). Its vertical accuracy is 0.3–1.0 m std. Both ALS and DEM data were cut to polygon samples by using 10 m × 10 m space partitioning slots. See two example polygon shapes in Figure 3. The further processing focused only to the point cloud limited by each polygon sample.

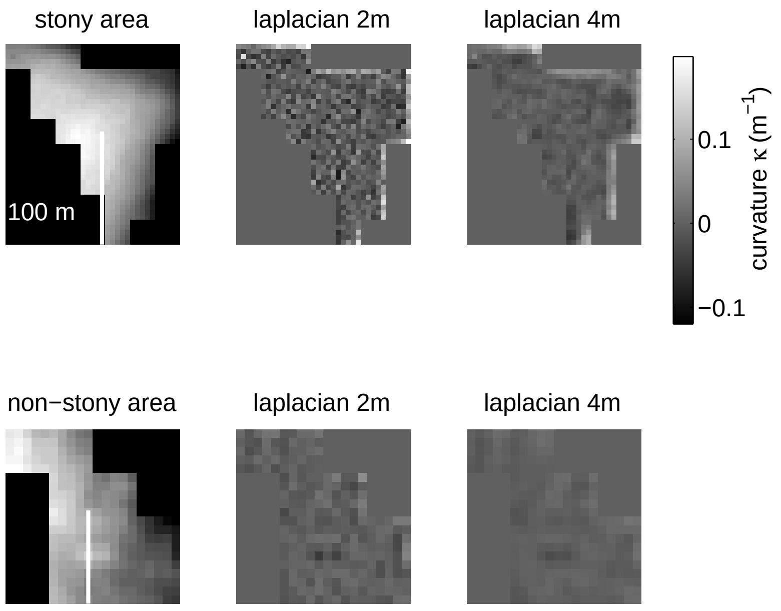


Figure 3. A stony (upper row) and a non-stony (lower row) sample polygon. Original polygons are approximated by 10 m × 10 m batches. The ground height (DEM 2 m) and its Laplace discrete operator signals with 2 m and 4 m radius are depicted. The border noise has been removed from actual analysis. The 100 m scale is aligned to North.

Stones are bare and the vegetation is thin due to high Northern latitudes.The point cloud on this site has approx. 25 % returns to forest canopy and approx. 75 % ground returns, so the ground signal is rather strong. The reflection errors were extremely rare, approx. 1 per 10^6 returns. Together the sample sets represent rather well the Kemijärvi study area.

2.3. Materials online

Sample data sets including some polygon point clouds, DEM data of two map pages, sample vectors, some field images, SAF algorithm document, a short description of the data set and the problem are available at: http://users.utu.fi/ptneva/ALS/.

2.4. Solid Angle Filtering (SAF)

The proposed solid angle filtering is a novel method to produce an alternative TIN DEM sensitive to stones and boulders on the ground. Filtering starts by forming an initial TIN either from a full point cloud or after an industry standard preliminary canopy and tree point elimination. The cut was made 2 meters above the local mode of the point cloud height. The 2D projection of TIN satisfies Delaunay condition at all times during the iterative process of point elimination. The prominent 'pikes' in the intermediate TIN are removed in random order while the Delaunay triangulation is updated correspondingly. The implementation requires a dynamical Delaunay algorithm, which facilitates incremental removal of points. We used an industry standard approach described in [36] with $O(k)$ computational complexity per removed point, where k stands for the average number of nearest neighbors of a point.

A second iterative phase removes 'pits' in a similar fashion. The prominence of pikes and pits is measured by solid angle Ω_k, which is the spatial angle of the surrounding ground when viewed from a TIN vertex point p_k. Appendix A provides the technical definition of computing Ω_k.

Each state of TIN is achieved by dropping one point which fails the following inequality:

$$\Omega_{min} \le \Omega_k \le \Omega_{max}, \tag{3}$$

where solid angle limits $\Omega_{min} = 1.80\,\mathrm{sr}$ (steradians) and $\Omega_{max} = 12.35$ sr correspond to solid angles of two spherical cones with opening angles 89 ° and 330 °, respectively. The choice affects the prediction performance: if both limits are close to a planar situation of $\Omega \approx 2\pi$, there is a loss of points. If there are no limitations ($\Omega_{min} \equiv 0$, $\Omega_{max} = 4\pi$), data is dominated by the noise from canopy and tree trunks. The solid angle limits were defined by maximizing the Kolmogorov-Smirnov (K-S) test [37] difference using 95 % confidence limit. Figure 4 depicts the difference between solid angle distributions at positive and negative sample sets at the choice we made. A pike at approx. 2.5 sr indicates stones.

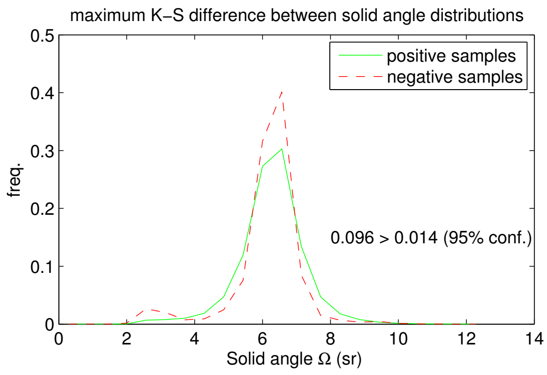


Figure 4. The solid angle distribution of positive and negative samples among the *data2015* data set. Averages can be distinguished well but variation among samples is high.

The resulting ground surface triangularization (see lower left part of Figure 2) resembles the end product of the 3D alpha shape algorithms [38], when alpha shape radius is chosen suitably. It produces an alternative TIN model which hopefully contains a signal needed in stone detection. In this paper this method is used as a preprocessing step for LTC method (see Section 2.6) and for LLC method (see Section 2.5, and Figure 1).

2.5. Curvature Estimation Based on Local Linear Fit (LLC)

LLC method is based on a curvature approximation method described in [33]. The method requires surface normals to be available at the triangle vertices. LLC provides these normals by a local linear fit to the point cloud at a regular horizontal grid. Since the resulting curvature function is not continuous at the border of the triangles, a voting procedure is needed to choose a suitable value for each grid point.

Finding the local planes is a similar task to finding the moving total least squares (MTLS) model in [39]. The differences are the following:

- a space partitioning approach is used instead of a radial kernel function to select the participant points. This is because ground surface can be conveniently space partitioned horizontally unlike in [39], where the point cloud can have all kinds of surface orientations.
- the point set is not from constructed environment. Canopy returns create a 3D point cloud, thus the loss function cannot be symmetrical, but must penalize points below the approximate local ground plane.

The LLC process has 6 steps, which are expounded in Appendix B. Step 1 is cutting the foliage dominated part of the point cloud, step 2 approximates ground with local linear planes at regular grid points. Step 3 spans the grid with triangles avoiding spots with missing data. Step 4 defines the curvature within each triangle. Step 5 combines the curvature values of the neighboring triangles to each grid point. Step 6 is about forming a histogram over the whole grid of the sample polygon.

LLC is a multi-scale method like [28]. Steps 1 through 6 are repeated with differing grid lengths $\delta_j, j \in [1,6]$ of the grid, see Table 2. There is a potential danger for overfitting, so the qualities of grid sizes are discussed here from that point of view.

Table 2. Square grid sizes used in local linear fit of LLC method.

Grid Version	**1**	**2**	**3**	**4**	**5**	**6**
Grid constant δ_m (m)	1.25	2.0	3.0	4.0	5.0	6.0

The smallest grid size $\delta_1 = 1.25$ m has approx. 85 % of the grid slots with only 1 to 3 points as shown in the left part of Figure 5 and so it represents a practical low limit of the local planar fit. A practical upper limit is $\delta_6 = 6$ m because only the largest boulders get registered on this scale. Such large boulders are few, as shown in the right part of Figure 5.

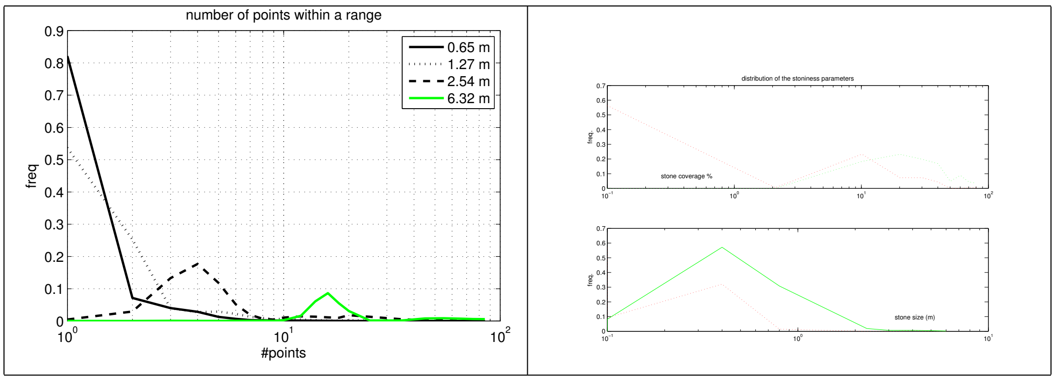


Figure 5. Approximative properties of *data2015* data set. Similar qualities of *data2014* are not available. **Left**: The number of stones at a spatial partition when the partitioning range (the grid size δ) changes. A sensible approximation of e.g., local ground inclination is possible only when there are at least 3 points per grid square. **Right**: The difference between positive and negative samples is mainly in stone size distribution. The practical detection limit in size is approx. 1.0 m.

2.6. Local Curvature Based on Ground Triangularization (LTC)

The input is a triangulated surface model generated by SAF method described in Section 2.4 and Figure 1. LTC produces the curvature estimates directly to the vertex points, so local linearization step 3 of the LLC method of Section 2.5 is not needed. Vectorization (step 4 of LLC) has to be done, but only once, since there are no grids nor multi-grids in this method. The idea is to calculate the value κ_k of a Gaussian curvature operator at each ground point p_k based on the Gauss-Bonnett theorem as described in [31].

The right side of the Figure A1 depicts a point p_k and its neighboring points p_i and p_j. The neighboring triangles T_k of a vertex point k define a so-called spherical excess, which is the difference between the sum of triangular surface angles β_{ikj} and the full planar angle 2π. Now, one can write an estimator for the Gaussian curvature κ_k at point p_k based solely on local triangularization formed by the Delaunay process:

$$\begin{aligned} \beta_{ikj} &= \quad \operatorname{acos}(p_i - p_k, p_j - p_k) \\ A_k &= \quad \textstyle\sum_{t \in T_k} A_t/3 \qquad (4) \\ \kappa_k &\approx \quad \left(2\pi - \textstyle\sum_{(i,k,j) \in T_k} \beta_{ikj}\right)/A_k, \qquad (5) \end{aligned}$$

where β_{ikj} is the angle at vertex k in a surface triangle $t = (i,k,j) \in T_k$, $\operatorname{acos}(.,.)$ is the angle between two vectors defined in Equation (A1) in Appendix A, and A_k is a characteristic surface area associated with the vertex k. The characteristic area has been defined approximately by taking one third of the area A_t of each adjoining triangle t. There are locally more stable but also more complicated ways to calculate A_k, see e.g., [31,32]. The choice made in Equation (4) causes noise because the area is approximate but seems to allow effective histogram vectors.

2.7. Curvature Based on Filtering DEM by a Modified Discrete Laplace Operator (DEC)

The third method is traditional, fast and easy to implement in the GIS framework and thus provides a convenient baseline for the previous two methods. Local height difference is converted to local curvature approximant. Curvature histograms are then vectorized as in previous methods.

DEM data with a regular grid with the grid size $\delta = 2.0$ m was utilized. Data is publicly available over most of Finland. The discrete 2D Laplace operator with radius $r_{horiz} = 2.0$ m is well suited for detecting bumpy features like stones at the grid detection limit. It simply returns the difference between the average height of 4 surrounding grid points and the height of the center point. A modified Laplacian filter with $r_{horiz} = 4.0$ m (length of two grid squares) was used to estimate the local height difference on the larger scale, see Figure 6. A postprocessing transformation by Equations (7) and (8) was applied to produce correspondence to Gaussian geometric curvature κ_k at point k. A geometric justification for the transformation is depicted in Figure 6. A stone is assumed to be a perfect spherical gap with perfect horizontal surrounding plane. The mean curvature κ_H can be approximated from the observed local height difference of Equation (6) by using the geometric relation Equation (7), see Figure 6. $\bar{z}(r_{horiz})$ is the average height at the perimeter of horizontal radius r. The local height difference $\bar{Z}$ is the key signal produced by Laplacian filter. Gaussian curvature is approximately the square of the mean curvature, when perfect sphericality is assumed, see Equation (8). The sign of the Gaussian curvature approximant κ_{Gk} at point c_k can be decided on the sign of the height difference $\bar{Z}$ at vertex k. The index k of the vertex point p_k is omitted for brevity. Equation (7) comes from rectangular triangle in Figure 6.

$$\begin{aligned} \bar{Z} &= \quad z - \bar{z}(r) && \text{local height difference} \qquad (6) \\ 1/\kappa_H^2 &\approx \quad r_{horiz}^2 + (1/\kappa_H - \bar{Z})^2 && \text{approximate mean curvature condition} \qquad (7) \\ \kappa_H &\approx \quad 2\bar{Z}/(\bar{Z}^2 - r_{horiz}^2) && \text{mean curvature solved from Equation (7)} \\ \kappa_G &\approx \quad -\operatorname{sign}(\bar{Z})\kappa_H^2 && \text{approximate Gaussian curvature} \qquad (8) \end{aligned}$$

Figure 6 shows grid squares of size 2 m $\times$ 2 m. Points A are used to calculate the average height $\bar{z}(2.0\,\mathrm{m})$ and points B the average height $\bar{z}(4.0\,\mathrm{m})$.

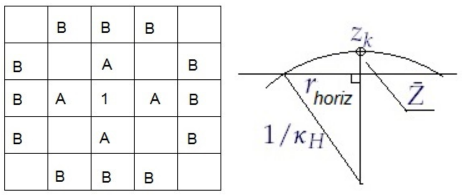


Figure 6. Left: The Laplace difference operator returns the height difference between the center point (1) and the average of points A. The modified Laplace difference operator does the same but using points B. These two kernels define each an average circumferential height difference $\bar{Z}$. **Right**: The geometric relation between $\bar{Z}$ and approximate mean curvature κ_H. Horizontal line represents average ground level at the circumference.

Many sample polygons are relatively small. The above described difference operator produces numerical boundary disturbance, see Figure 3. This can be countered by limiting the perimeter average $\bar{z}(r)$ only to the part inside the polygon, and then removing the boundary pixels from the histogram summation.

The next step is to build the sample histograms as with other methods. Histogram vectors from two filters are concatenated to produce a sample vector of a polygon. The details of forming the histogram are given in Section 2.8.

2.8. Vectorization

All three methods produce histograms of Gaussian curvature $\kappa_G = \pm 1/r^2$, where r is the local characteristic curvature radius and the curvature sign has been chosen in Equation (8) so that potential stone tops have negative curvature and "pit bottoms" have positive curvature. An ideally planar spot k has curvature radius $r_k \equiv \infty$ and curvature $\kappa_G \equiv 0$. Recall that minimum detectable stone radius is approx. $r_{min} = 0.6...1.2$ m, which leads to a Gaussian curvature interval of $\kappa_G \in= [-1.8, 1.8]\ \mathrm{m}^{-2}$. This range was spanned by histogram bins. LLC and DEC are rather insensitive to bin choice, so a common ad hoc choice was made for these methods, see Table 3. The LTC method proved sensitive to bin choices, so the values were derived using a subset of 10 positive and 10 negative samples in leave-pair-out cross-validation. This set was excluded from later performance measurements.

Table 3. Curvature histogram bins.

Method	Positive Half of the Bin Values
LLC and DEM	0.010, 0.030, 0.060, 0.13, 0.25, 0.50, 1.0, 2.0
LTC	0.031, 0.12, 0.25, 0.44 ,0.71, 1.13, 1.8

The histogram creates a vector representation $\mathbf{x}_i$, $i = 1..n$ for all sample polygons i. The LTC method produces one histogram vector, the DEC method produces two vectors (for $r = 2$ and $r = 4$ m) and LLC produces 6 vectors (for 6 different grids), which are then concatenated to form the final sample vector $\mathbf{x}_i$.

Figure 7 provides a summary of average curvature distributions produced by each of the three methods. The planar situation with $\kappa_G \equiv 0$ is the most common. Occurrences with characteristic radius $r < 1$ m are very rare. Useful information is contained within the range $\kappa_G \in [-1, 1]\ \mathrm{m}^{-2}$. With LLC method, grid size $\delta = 2$ m is able to detect greater curvatures and grid size $\delta = 5$ m is the last useful grid size. DEM is remarkably similar to 2 m LLC grid, which was to be expected.

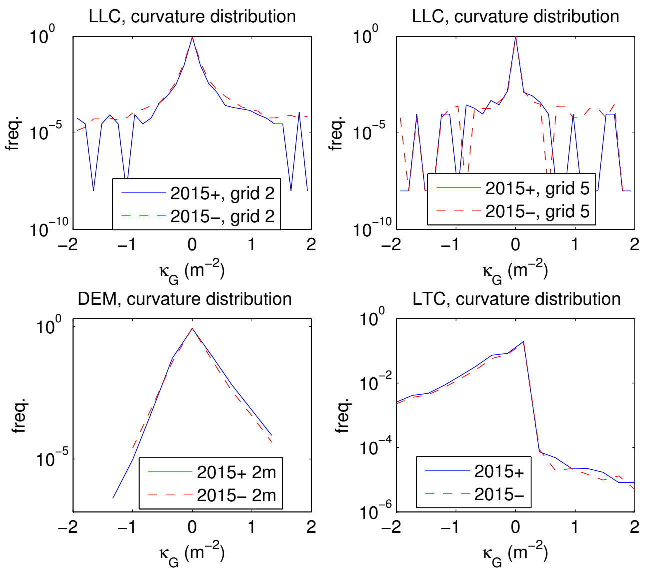


Figure 7. Curvature distributions produced by each method. **Upper left**: LLC and grid size 2m. **Upper right**: LLC and grid size 4 m. Larger grid size results in narrow band around $\kappa = 0$. **Lower left**: DEM curvatures are characterized by kurtosis. **Lower right**: the LTC distribution.

Negative samples (dashed lines) have about the same amount of exactly planar samples, but are a little bit more slightly curved ($|\kappa_G| \approx 0.4$ m^2) samples. This probably results from the basic ground curvature distribution. If the presented three methods were to be adapted elsewhere, the changing background curvature spectrum may result in changes in the prediction performance.

The LTC method is able to detect the negative curvature around the stone causing the curvature distribution to be asymmetric. Unfortunately, this method is also very noise-sensitive reducing its performance.

Each method requires more testing and especially test data with known stone properties (relative coverage of the surface, radius and height distribution, individually located and labeled stones).

2.9. Logistic Regression

The label vector $y_i \in \{-1, 1\}$, $i \in D = 1...n$ was acquired by field campaign done by a geology expert. D is the index set of the full sample set, size $n = |D|$ varies depending e.g., on the different sensitivity to sparse point cloud of each method. The sample vectors $\mathbf{x}_i \in \mathbb{R}^d$ are produced by histogram vectorization described in Section 2.8. Dimensionality d varies depending on the method. We use the affine form $\mathbf{z}_i = (1, \mathbf{x}_i)$ to shorten the following treatise. Vectors $\{\mathbf{z}_i\}_{i \in D}$ are also standardized before solving the regression problem.

This is a qualitative response problem, so logistic regression was chosen to predict a label $\hat{y}$ from a given sample vector $\mathbf{x}$. The prediction coefficient $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$ is tuned by usual maximum likelihood approach to optimal value $\boldsymbol{\beta}^*_{D'}$ using a sample set $\{(\mathbf{z}_i, y_i)\}_{i \in D'}$, $D' \subset D$ where D' is the training set used:

$$f(\mathbf{z}_i, \boldsymbol{\beta}) = Pr(y_i = 1 \,|\, \mathbf{z}_i) = (1 + \exp[-\boldsymbol{\beta} \cdot \mathbf{z}_i])^{-1}$$
$$\hat{y}^{(D')}(\mathbf{z}) = \begin{cases} 1 & \text{when } f(\mathbf{z}, \boldsymbol{\beta}^*_{D'}) \geq 1/2 \\ -1 & \text{otherwise} \end{cases} \tag{9}$$

The area under curve (AUC) performance measure is natural in this application area, where cost functions do exist but are not exactly known. The sample set *data2014* is rather small and the sample set *data2015* has imbalanced samples of positive and negative cases and the n-fold cross-validation may produce too optimistic estimates, as mentioned in [40]. It is recommended in [40] in this case:

- to perform a leave-pair-out (L2O) test over all possible positive-negative label pairs P, and
- to measure L2O area under curve AUC by using the Heaviside function $H(.)$ for summation.

$$\begin{aligned} P &= \{(i,j) \mid y_i = 1, y_j = -1, i,j \in D\} && \text{all possible (+,–) pairs} \\ AUC &= \textstyle\sum_{(i,j)\in P} H(\hat{y}_{ij}(\mathbf{z}_i) - \hat{y}_{ij}(\mathbf{z}_j))/|P| && \text{leave-pair-out AUC} \\ \hat{y}_{ij}(\mathbf{z}) &= \hat{y}^{(D\setminus\{i,j\})}(\mathbf{z}) && \text{prediction without a pair } i,j \text{ based on } \mathbf{z} \\ H(\Delta\hat{y}) &= (1+\operatorname{sign}(\Delta\hat{y}))/2 && \text{Heaviside over the prediction difference} \end{aligned} \tag{10}$$

2.10. Method Parameters and Design Choices

Some parameters were optimized by nested cross-validation or K-S test and thus settled. Some parameters are ad-hoc built-in parameters, values of which are chosen mostly during the coding process. These should be taken into consideration if the methods are utilized under slightly different conditions. A list of the open parameters, their potential range and some of the discrete design choices available, follows. Table 4 is a summary of the treatise.

Table 4. Effective method parameters, a summary.

Method	Parameters	Binary Choices
SAF	2	0
LLC	3–15	63
LTC	0	1
DEC	2	1

LLC: There are 11 non-zero shape parameters of the planar distance weight function $g(.)$ presented in [11]. The validation of the choices made will be a separate publication. There are some minor design choices, an example is Equation (B2): one can use either median or mean rule in composing curvature from surrounding triangle vertices, and results do not change noticeably. Median rule was used to reduce occasional outliers. Another example is the 6 grid sizes in Table 2. The number of possible subsets of grids to be used equals $2^6 - 1 = 63$.

LTC: There is a design choice of using the local surface area $A_t/3$ in Equation (4) or a more complex definition given in [32]. This is listed as one binary choice in Table 4.

DEC: There is a binary choice of either choosing Laplacian filter signal $\bar{Z}$ or the Gaussian approximant κ_k of Equation (8) based on the signal $\bar{Z}$.

2.11. General Wall-to-Wall Prediction

Methods presented in Sections 2.4–2.9 were applied only to given polygon areas, since teaching is possible only where the response value is known. But after the parameters of predictor have been settled, the area to be inspected can be a generic one. As a demonstration and speed test, we applied methods to a 1080 km^2 area divided to 20 m $\times$ 20 m pixels with approx. 320 points from a point cloud of density $\rho = 0.8\,\text{m}^{-2}$. Pixels have 6 m overlapping margins to increase the sample area to 32 m $\times$ 32 m (approx. 820 points) to avoid partially populated histograms, which would not be recognized correctly by the classifier. See Figure 8 for the DEC method wall-to-wall result.

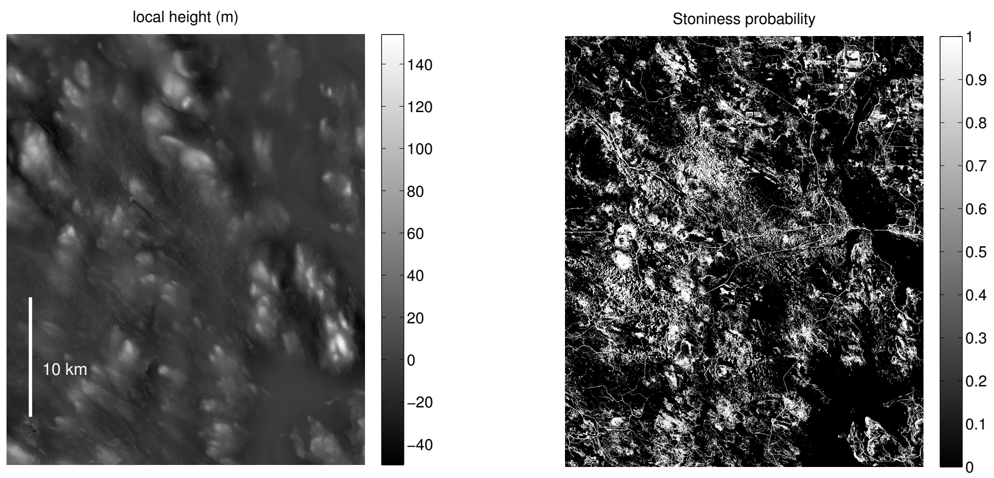


Figure 8. Left: The local height from DEM files, 30 km × 36 km area depicted. The scale is oriented northwards. The general location of the rectangle can be seen in upper left part of the Figure 2. **Right**: Stoniness probability by DEC method. The scale is probabilty of having stones on a particular pixel. Roads and waterways are classified as stony areas. LLC and LTC methods are much less sensitive to roads and constructed details.

3. Results

Binary classification of stoniness was done by logarithmic regression over curvature histogram vectors cumulated over each sample polygon area. Three methods were used; they differ on how the curvature approximants were produced(Table 5):

- Local linear fitting (LLC) divides the polygon into 6 different grids. Each grid square is fit by a plane approximating the local ground height of the center of the plane and the plane orientation. Curvatures are computed from these center points and their orientation normals.
- Curvature from DEM (DEC) uses traditional DEM data. Curvatures are approximated by the observed local height difference delivered by a modified discrete Laplace operator.
- Curvature by local triangulation (LTC) has a TIN computed by SAF method of Section 2.4. The curvature is then computed triangle by triangle as in LLC.

Area under curve AUC [41] was measured using both data sets and all three methods, see Section 2.9. The AUC measure describes the discriminative power of a predictor over various possible cutoff point choices. A proper cutoff depends on the costs involved and is not known at the moment, justifying the accommodation of AUC. Leave-pair-out variant of AUC was used, see considerations for this choice in Section 2.9.

Table 5. Leave-pair-out AUC results based on three methods used: digital elevation model, local linear fit and local triangular curvature for two polygon sample sets.

Data Set	DEC	LLC	LTC
data2014	0.85	0.82	0.79
data2015	0.68	0.77	0.66

LLC proved best for *data2015*. This data set is large and perhaps more representative of the locality, and the performance $AUC = 0.77$ can be considered adequate for practical application such as pre-selecting possible gravel deposit sites for infrastructure construction. Its performance is also on par with many hard natural resource prediction tasks based on open data, see e.g., [2].

Data set *data2014* is somewhat exceptional, since it contains larger boulders and seems to be an easy prediction task for wide array of methods. Both DEC and LLC performed well. The same holds to several other tested methods which have not been included to this report, e.g., neighborhood voting based on solid angle values.

DEC performance is mediocre with *data2015* set because the average stone size depicted in Figure 5 right side is actually below the theoretical detection limit of a regular 2 m grid. Established DEM computation routines are a trade-off of many general-purpose goals and some of the stone signal seems to be lost in the case of the *data2015* set.

LLC, eventhough it was cross-validated with *data2014*, performed adequately here. There were high hopes about local curvature based on triangularization (LTC), but it performed the worst. This is because LTC computes the curvature directly from a TIN, and the process produces a lot of noise. LTC has been included in this report mainly because the method is fast to compute and there seems to be potential to reduce noise in the future by neighborhood voting methods.

The processing speed (see Table 6) has a linear dependence with the area analyzed. This is because the analysis is done by space partitions of constant point cloud sample size n. All the steps have a linear $O(n)$ time complexity except the Delaunay triangulation in SAF. The point removal phase in SAF is of $O(kn)$ complexity, where $k \approx 5...20$ is the amount of nearest neighboring points. The experiments were run on a desktop computer with Intel Core i5-3470 CPU (3.20 GHz) running Ubuntu Linux 14.10. LLC implementation requires several intermediary file operations, which makes it slow. All implementations are experimental prototypes and many speed improvements are still possible.

Table 6. Analysis speed computed from average of two runs over the data set *data2015*.

Analysis Speed	DEC	LLC	LTC
km^2/h	200	0.5	4.0

4. Discussion

A traditional approach for terrain micro-topography classification is to use DEM model as a basis for a wide array of texture methods. The low end has a simple texture feature computation followed by segmentation tuned manually by an expert. The high end has several texture features extracted, and preferably at least two DEM models of different grid size as a basis for analysis.

This paper presents a way to use the existing ALS LiDAR material to construct an alternative task-specific terrain surface representation which hopefully contains more information e.g., concerning the presence of stones. All methods presented are conceptually simple, although documenting and coding LLC and LTC brings up a multitude of details and ad hoc choices, see e.g., the number of method parameters in Table 4. Each method has potential for further improvement by a more thorough parameter tuning. SAF and LLC have enough method parameters that tuning by new field campaign data at more southern boreal forests could succeed. More southern boreal forest provides a challenge since the ratio of the ground returns is only 30%–60% instead of 70% at our Northern test site.

LLC and DEC perform well enough to be practically usable. The direct utilization of ALS data seems to work on this site.

Because LTC is based on a TIN model, there is available additional geometrical information like mean curvature, curvature eigenvalues and eigenvectors etc. for more complex micro-topographic features. If it is possible to reduce the noise and keep the computation costs at the current low level, a combination of these features could be a basis for fruitful multi-layer texture analysis.

Many terrain micro-topography classification tasks e.g., registering post-glacial landslides, karst depresssion detection and fault lines detection e.g., can be done with DEM and by texture methods, but there may be a need to add stoniness or curvature related features to improve the classification. The current data sets do not provide accurate quantitative information about stoniness for regression methods. The wall-to-wall stoniness result with 20 m × 20 m pixels produced by current binary classification (see Figure 8) can be utilized as an additional feature in other prediction problems in the future. The wall-to-wall pixel size must be increased when the ground return ratio decreases in the southern dense forests.

Three individual methods or a combination of them can be modified to produce estimates of the relative stone coverage and stone size distribution. This step can be taken only if data sets come available with individual stones and their properties tagged out. Furthermore, more sophisticated probabilistic and minimum description length (MDL) based methods would then be possible. The stone coverage and size distribution information in Figure 5 is approximative only, so current data sets cannot be used for development of quantitative stoniness models.

It is hard to estimate how far to southern forests the three methods can be extended. The ground return density varies with boreal forests, and detection results from spatially rather sparse accessible spots should be somehow extended to nearby areas using other available public data and Machine Learning methods. This line of research requires specific field test sets, though.

There are several other micro-topological problems, e.g., classifying and detecting undergrowth, marshland types, military structures, unauthorized inhabitation and geomorphology e.g., frost phenomena. Some of these require comparison of two snapshots from ultra-light vehicle (UAV) scans, and e.g., a combination of SAF and MDL might perform well in this scenario. We believe SAF has wide adaptivity to several purposes by tuning its 2 parameters (minimum and maximum spatial angle allowed) and MDL can be built to detect a specific shape (a hemisphere in cases of stones, a box, a prism or a cylinder in case of other applications). As mentioned before, MDL would require denser point clouds with $\rho \approx 1.6...3.2\,\mathrm{m}^{-2}$.

5. Conclusions and Future Research

Results in Section 3 show that LLC performs better than DEC but is numerically much more expensive. LLC seems to be robust and useful when the computation costs can be amortized over several subsequential analyses. LTC performed worst but there is room for improvement as discussed in Section 4. Both LLC and DEC are ready to be applied to industrial purposes after prototyping implementations are upgraded to production code.

Current results are bound to stoniness of mass-flow deposits what comes to teaching data, but each method should work in generic stoniness detection, if such a need arises and general teaching data sets become available.

Using direct ALS information either as an alternative data source or supplementary one may help solving a variety of micro-topography detection problems better in the future. The research efforts will be focused on the following topics:

- Extending the analysis to more dense forests, where stoniness detection occurs only at benevolent cicumstances (forest openings, sparse canopy, hilltops). In this environment the acquired stoniness signal has to be combined to a wide array of open data features to extend prediction to unobservable areas. The corresponding field campaigns will be more elaborate.
- Taking into account the stone coverage and size distribution. It is likely that a multi-grid method like LLC might perform well in this prediction task (given suitable teaching data), whereas DEC may be restricted by the general purpose nature of DEM and its modest grid size.
- Topography and vegetation classification of marshlands. Marshlands have similar high ground return ratio as the current case site. SAF can be tuned by cross-validation to produce a tailored TIN and an improved LTC method with added curvature properties (mean curvature, curvature eigenvectors) could detect various micro-topographic marshland features. It is our assumption that the histogram approach would work also with marshland classification, given a suitable teaching polygon quality produced in field campaigns.
- Using min-cut based segmentation of k-NNG graph of ALS data as described in [15] instead of simple Delaunay triangulation. One has to modify the algorithm to include neighborhood voting

to reduce noise. This could be a fruitful approach, since it could suit to 3D analysis of forest tree species, providing more motivation for the implementation.

- Utilizing all relevant LiDAR attribute fields, like return intensity, return number, the scan angle etc. (see [21]).

Acknowledgments: Department of Information Technology of University of Turku covered the travel and publication costs. English language was revised by Kent Middleton.

Author Contributions: Maarit Middleton designed and conceived the sample polygon sets; Raimo Sutinen conceived the research problem; Paavo Nevalainen designed the methods and analyzed the data; Paavo Nevalainen wrote the paper; Jukka Heikkonen contributed to the paper; Tapio Pahikkala contributed to the performance analysis.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ALS	Aerial laser scan
AUC	area under curve
DDG	Discrete differential geometry
DEC	Curvature based on DEM
DEM	Digital elevation model
DTM	Digital terrain model
GIS	Geographic information system
k-NNG	k-nearest neihgbors graph
K-S	Kolmorogov-Smirnov test
L2O	Leave-pair-out
LiDAR	Light detection and ranging
LLC	curvature based on local linear fit
LTC	local curvature based on ground triangulation
MDL	Minimum description length principle
MTLS	Moving total least squares
NLS	National Land Survey of Finland
PCA	principal components analysis
SAF	Solid angle filtering
TIN	triangulated irregular network
UAV	Ultra-light vehicle

Appendix A

The computation of a solid angle Ω_k takes place at the approximity of a point p_k, namely at the set of the adjoining triangles T_k, see detail A) of Figure A1. Detail B) presents a tetrahedron defined by points p_k, p_i, p_j, p_l, where $p_i, p_j \in T_k \backslash p_k$ are from the outskirt of the triangle set T_k and p_l is an arbitrary point directly below point p_k. There are several ways to implement the solid angle calculation, a formula based on a classical l'Huillier's theorem [42] is presented here:

$$\begin{array}{rclll}
\mathrm{acos}(\mathbf{a},\mathbf{b}) & = & \cos^{-1}(\mathbf{a}^0 \cdot \mathbf{b}^0) & \text{angle between two vectors} & \text{(A1)} \\
\mathbf{x}^0 & := & \mathbf{x}/\|\mathbf{x}\|_2 & \text{vector normalization} & \\
\alpha_i & = & \mathrm{acos}(p_l - p_k, p_j - p_k) & \text{compartment angles } i,j,k & \\
\alpha_j & = & \mathrm{acos}(p_i - p_k, p_l - p_k) & & \\
\alpha_l & = & \mathrm{acos}(p_j - p_k, p_i - p_k) & & \\
\alpha_0 & = & \alpha_i + \alpha_j + \alpha_l & \text{basic product term} & \\
\omega_{ilj} & = & 4\tan^{-1}\sqrt{\tan\frac{\alpha_0}{4}\Pi_{\nu\in\{i,j,l\}}\tan\frac{\alpha_0-2\alpha_\nu}{4}} & \text{compartment angle} & \text{(A2)} \\
\Omega_k & = & \sum_{ilj\in T_k}\omega_{ilj} & \text{solid angle at point } p_k & \text{(A3)}
\end{array}$$

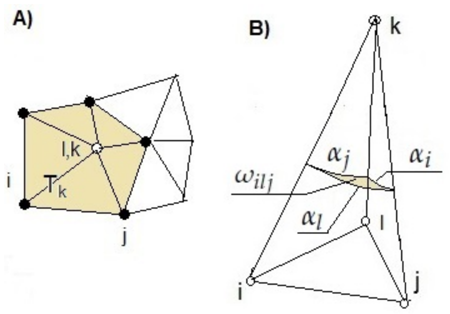


Figure A1. Solid angle filtering. (**A**) The set of adjoining triangles T_k of a point p_k seen from above; (**B**) A compartment *ijl* of the vertex point p_k presented in detail. A solid angle Ω_k is a sum of compartment angles ω_{ilj} of Equation (A2). Point p_l is an arbitrary point directly below the vertex point p_k.

Equation (A2) approaches so called Heron's planar trigonometric formula [42] when angles α_i, ... approach zero. The practical implementation requires a combination of space partitioning to a manageable point cloud patches of approx. 700...2000 points and a 2D Delaunay triangulation with an efficient point removal method. We use a batch version (python *scipy.spatial.delaunay*) with our own industry standard routine for deletion. This combination is simple to implement and excels in practise as [36] mentions, even though there exists faster incremental deletion arrangements with 2..3 times slower construction phase.

Appendix B

Step 1: Data can be preprocessed in three different ways before the LLC step. Alternatives are listed here in the order of increasing computational efficiency and decreasing amount of points:

(a) raw 3D ALS data
(b) same as (a) with tree and foliage returns cut from approx. 2 m height from approximative ground level
(c) TIN model produced e.g., by solid angle filtering of Section 2.4

The local linear fitting step 2 finds similar ground model per each alternative, only the speed of convergence varies. All three alternatives seem to result in about the same quality when measured by predictor performance.

Step 2: The fitting of the plane resembles a normal regression problem with an ad hoc nonlinear loss function, which penalizes residuals below the plane to force the ground fit. By applying the fitting process to planes of varying sizes one gets an assembly of plane orientations and plane centers. The neighboring planes can then be used to approximate local curvature. Since sample density is low, some of the plane fits cannot be performed. Therefore, it is numerically more resilient to use triangulation over neighboring planes and define curvature over each triangle using formulation developed in [33]. Another approach would be to produce a triangular mesh and estimate curvature based on it, as in [31,32]. There is a large filtering effect in this approach, since vertex normals depend on surrounding vertices. Local linear fit seemed to pick up the stoniness signal better, especially since we used multi-scale grids.

The grid division has been depicted in Figure B1. At each grid slot, one has to find the best fitting plane $\mathcal{P}(p, \mathbf{n})$, where $p \in \mathbf{R}^3$ is the center point of the plane $\mathcal{P}$ and $\mathbf{n}$ its normal. The initial state for the plane is: $\mathcal{P}_0 = \mathcal{P}(\text{lowest point of local sample}, (0,0,1)^T)\ \delta$.

The plane $\mathcal{P}$ represents a good local ground linearization provided that the weight function $g(l)$ of the orthogonal distance l penalizes heavily points below the approximated ground level. The details of weight function have been published in [11]. The practical considerations in selecting the weight function shape are at rapid and guaranteed convergence, whereas the influence to prediction performance comes from the actual ground returns and their geometry.

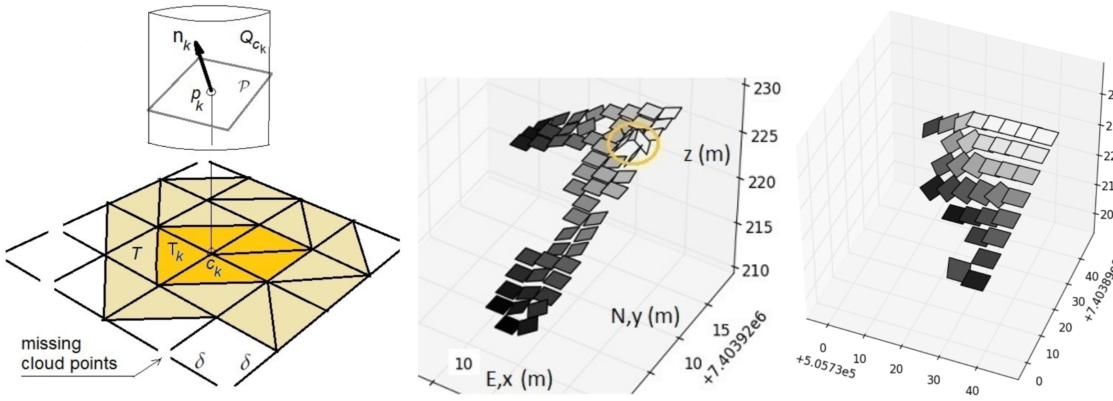


Figure B1. Left: An individual local plane $\mathcal{P}(p_k, \mathbf{n}_k)$ at grid point c_k and its parameters (local plane center point p_k and normal $\mathbf{n}_k$). A triangulation T of the grid avoids squares with incomplete data. A local cloud point set Q_{c_k} and neighboring triangles $T_k \subset T$ of a grid slot c_k are also depicted. **Center**: a stone revealed by two adjacent tilted planes. This stone provides a signal with the grid size $\delta = 2$ m. Note the amount of missing planes due to a lack of cloud points. **Right**: The grid of size $\delta = 4$ m at the same spot. The stone does not appear, local variation has disappeared but the grid is almost full approximating the sample polygon shape.

The optimal fit at each grid slot concerns now coordinate components $\mathbf{w}^T = (p_z, n_x, n_y)$ of p and $\mathbf{n}$. The local plane $\mathcal{P}$ is found by a numerical minimization:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{q_j \in Q_c} g[(q_j - p) \cdot \mathbf{n}] \tag{B1}$$

Step 3: Triangulation is based on the grid centers, where the surface normal is known. Because of relatively low point density, some grid locations are bound to have no points and thus have to be omitted from triangulation, see Figure B1. The triangularization T outlined in Figure B1 is generated randomly, see Figure B1. The end result dictates the adjoining triangle sets $T_k \subset T$ of each grid point k. The size $|T_k|$ of the adjoining triangle sets varies depending on how dense or sparse point cloud is nearby point k: $1 \leq |T_k| \leq 8$.

Step 4: The curvature is approximated on each vertex of each triangle as in [33]. There are several other similar formulations e.g., using rectangular grids, but those are not so suitable in the presence of sparse and missing cloud points. The end result is set of candidate curvatures $\kappa_{tk}, t \in T_k$ per each grid point p_k.

Step 5: Now the task is to combine the final curvature approximant at a grid point p_k by taking a median of values available at the adjoining vertices of all surrounding triangles:

$$\kappa(p_k) = \underset{t \in T_k}{\operatorname{median}} \, \kappa_{tk} \tag{B2}$$

Step 6: We used the normalized histograms of $\mathbf{h} = \text{hist}_{k \in K} \kappa(p_k)$, where K is the set of grid centers and histogram operator hist(.) and its properties are documented in Section 2.8.

References

1. Middleton, M.; Nevalainen, P.; Schnur, P.; Hyvnen, T.; Sutinen, R. Pattern recognition of mass-flow deposits from airborne LiDAR. In Proceedings of the 36th EARSeL Symposium, Bonn, Germany, 20–24 June 2016.
2. Pohjankukka, J.; Nevalainen, P.; Pahikkala, T.; Hyvönen, E.; Middleton, M.; Hänninen, P.; Ala-Ilomäki, J., Heikkonen, J. Predicting water permeability of the soil based on open data. In Proceedings of the 10th Artificial Intelligence Applications and Innovations (AIAI 2014), Rhodes, Greece, 19–21 September 2014

3. Palmu, J.P.; Ojala, A.; Ruskeeniemi, T.; Sutinen, R.; Mattila, J. LiDAR DEM detection and classification of postglacial faults and seismically-induced. *GFF* **2015**, *137*, 344–352.
4. Johnson, M.D.; Fredin, O.; Ojala, A.; Peterson, G. Unraveling Scandinavian geomorphology: The LiDAR revolution. *GFF* **2015**, *137*, 245–251.
5. Sutinen, R.; Middleton, M.; Liwata, P.; Piekari, M.; Hyvönen, E. Sediment anisotropy coincides with moraine ridge trend in south-central Finnish Lapland. *Boreas* **2009**, *38*,638–646.
6. Sutinen, R.; Hyvönen, E.; Kukkonen, I. LiDAR detection of paleolandslides in the vicinity of the Suasselk posglacial fault, Finnish Lapland. *Int. J. Appl. Earth Obs. Geoinf. A* **2014**, *27*, 91–99.
7. Alho, P.; Vaaja, M.; Kukko, A.; Kasvi, E.; Kurkela, M.; Hyyppä, J.; Hyyppä, H.; Kaartinen, H. Mobile laser scanning in fluvial geomorphology: mapping and change detection of point bars. *Z. Geomorphol. Suppl.* **2011**, *55*, 31–50.
8. Sutinen, R.; Hyvönen, E.; Middleton, M.; Ruskeeniemi, T. Airborne LiDAR detection of postglacial faults and Pulju moraine in Palojärvi, Finnish Lapland. *Glob. Planet. Chang.* **2014**, *115*, 24–32.
9. Gallay, M. *Geomorphological Techniques*, 1st ed.; Chapter Section 2.1.4: Direct Acquisition of Data: Airborne Laser Scanning; British Society for Geomorphology: London, UK, 2012.
10. Li, Z.; Zhu, C.; Gold, C. *Digital Terrain Modeling*; CRC Press, Roca Baton, FL, USA, 2004.
11. Nevalainen, P.; Kaate, I.; Pahikkala, T.; Sutinen, R.; Middleton, M.; Heikkonen, J. Detecting stony areas based on ground surface curvature distribution. In Proceedings of the 5th International Conference Image Processing, Theory, Tools and Applications IPTA2015, Orléans, France, 10–13 November 2015.
12. Waldhauser, C.; Hochreiter, R.; Otepka, J.; Pfeifer, N.; Ghuffar, S.; Korzeniowska, K.; Wagner, G. Automated Classification of Airborne Laser Scanning Point Clouds. *Solv. Comput. Expens. Eng. Probl. Proc. Math. Stat.* **2014**, *97*, 269–292.
13. Meng, X.; Currit, N.; Zhao, K. Ground Filtering Algorithms for Airborne LiDAR Data: A Review of Critical Issues. *Remote Sens.* **2010**, *2*, 833–860.
14. Haugerud, R.; Harding, D. Some algorithms for virtual deforestation (VDF) of LiDAR topographic survey data. *Int. Arch. Photogramm. Remote Sens.* **2001**, *34*, 219–226.
15. Golovinskiy, A.; Funkhouser, T. Min-Cut Based Segmentation of Point Clouds. In Proceedings of the IEEE Workshop on Search in 3D and Video (S3DV) at ICCV, Kyoto, Japan, 27 September–4 October 2009.
16. Page, D.L.; Sun, Y.; Koschan, A.F.; Paik, J.; Abidi, M.A. Normal Vector Voting: Crease Detection and Curvature Estimation on Large, Noisy Meshes. *Graph. Model.* **2002** *64*, 199–229.
17. Johnson, K.M.; Ouimet, W.B. Rediscovering the lost archaeological landscape of southern New. *J. Archaeol. Sci.* **2014**, *43*, 9–20.
18. Vosselman, G.; Zhou, L. Detection of curbstones in airborne laser scanning data. ISPRS, 2009, pp. 111–116.
19. Brubaker, K.M.; Myers, W.L.; Drohan, P.J.; Miller, D.A.; Boyer, E.W. The Use of LiDAR Terrain Data in Characterizing Surface Roughness and Microtopography. *Appl. Environ. Soil Sci.* **2013**, *2013*, 1–13.
20. Hengl, T.; Reuter, H. (Eds.) *Geomorphometry: Concepts, Software, Applications*; Elsevier: Amsterdam, The Netherland, 2008; Volume 33, p. 772.
21. ASPRS. *LAS Specification Version 1.4-R13*; Technical Report; The American Society for Photogrammetry & Remote Sensing(ASPRS): Bethesda, ML, USA, 2013.
22. Weishampel, J.F.; Hightower, J.N.; Chase, A.F.; Chase, D.Z.; Patrick, R.A. Detection and morphologic analysis of potential below-canopy cave openings in the karst landscape around the Maya polity of Caracol using airborne LiDAR. *J. Cave Karst Stud.* **2011**, *73*, 187–196.
23. Telbisz, T.; Látos, T.; Deák, M.; Székely, B.; Koma, Z.; Standovár, T. The advantage of lidar digital terrain models in doline morphometry copared to topogrpahic map based datasets—Aggtelek karst (Hungary) as an example. *Acta Carsol.* **2016**, *45*, 5–18.
24. de Carvalho Júnior, O.A.; Guimaraes, R.F.; Montgomery, D.R.; Gillespie, A.R.; Gomes, R.A.T.; de Souza Martins, É.; Silva, N.C. Karst Depression Detection Using ASTER, ALOS/PRISM and SRTM-Derived Digital Elevation Models in the BambuiGroup, Brazil. *Remote Sens.* **2014**, *6*, 330–351.
25. Kobal, M.; Bertoncelj, I.; Pirotti, F.; Kutnar, L. LiDAR processing for defining sinkhole characteristics under dense forest cover: A case study in the Dinaric mountains. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2014**, *XL-7*, 113–118.
26. Hulík, R.; Španěl, M.; Materna, Z.; Smrž, P. Continuous Plane Detection in Point-cloud Data Based on 3D Hough Transform. *J. Vis. Commun. Image Represent.* **2013**, *25*, 86–97.

27. Yang, M.Y.; Förstner, W. *Plane Detection in Point Cloud Data*; Technical Report TR-IGG-P-2010-01; University of Bonn, Bonn, Germany, 2010.
28. Evans, J.S.; Hudak, A.T. A multiscale curvature algorithm for classifying discrete return lidar in forested environments. *IEEE Trans. Geosci. Remote Sens.* **2007**, *45*, 1029–1038.
29. Pauly, M.; Keiser, R.; Gross, M.; Zrich, E. Multi-scale Feature Extraction on Point-sampled Surfaces. *Comput. Graph. Forum* **2003**, *22*, 281–290.
30. Palancz, B.; Awange, J.; Lovas, T.; Fukuda, Y. Algebraic method to speed up robust algorithms: Example of laser-scanned point clouds. *Surv. Rev.* **2016**, 1–11, doi:10.1080/00396265.2016.1183939.
31. Meyer, M.; Desbrun, M.; Schröder, P.; Barr, A.H. Chapter Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. *Visualization and Mathematics III*; Springer Berlin Heidelberg: Berlin, Heidelberg, 2003; Volume 7, pp. 35–57.
32. Mesmoudi, M.M.; Floriani, L.D.; Magillo, P. *Discrete Curvature Estimation Methods for Triangulated Surfaces*; Lecture Notes in Computer Science; Springer: New York, NY, USA, 2010; Volume 7346, pp. 28–42.
33. Theisel, H.; Rössl, C.; Zayer, R.; Seidel, H.P. Normal Based Estimation of the Curvature Tensor for Triangular Meshes. In Proceedings of the 12th Pacific Conference on (PG2004) Computer Graphics and Applications, Seoul, Korea, 6–8 October 2004; pp. 288–297.
34. Crane, K.; de Goes, F.; Desbrun, M.; Schröder, P. *Digital Geometry Processing with Discrete Exterior Calculus*; ACM SIGGRAPH 2013 Courses; ACM: New York, NY, USA, 2013.
35. Lin, X.; Zhang, J. Segmentation-Based Filtering of Airborne LiDAR Point Clouds by Progressive Densification of Terrain Segments. *Remote Sens.* **2014**, *6*, 1294–1326.
36. Devillers, O. On Deletion in Delaunay Triangulations. *Int. J. Comp. Geom. Appl.* **2002**, *12*, 193–205.
37. Massey, F. The Kolmogorov-Smirnov Test for Goodness of Fit. *J. Am. Stat. Assoc.* **1956**, *46*, 66–78.
38. Bernardini, F.; Mittleman, J.; Rushmeier, H.; Silva, C.; Taubin, G. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. Visual. Comput. Graph.* **1999**, *5*, 349–359.
39. Pauly, M.; Keiser, R.; Kobbelt, L.P.; Gross, M. Shape Modeling with Point-sampled Geometry. *ACM Trans. Graph.* **2003**, *22*, 641–650.
40. Airola, A.; Pahikkala, T.; Waegeman, W.; Baets, B.D.; Salakoski, T. An experimental comparison of cross-validation techniques for estimating the area under the ROC curve. *Comput. Stat. Data Anal.* **2010**, *55*, 1824–1844.
41. Fawcett, T. An Introduction to ROC Analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874.
42. Zwillinger, D. *CRC Standard Mathematical Tables and Formulae*, 32th ed.; CRC Press, Boca Raton, FL, USA, 2011.

Publication R4

Real-Time Swimmer Tracking on Sparse Camera Array

P. Nevalainen, M. H. Haghbayan, A. Kauhanen, J. Pohjankukka, M.-J. Laakso, J. Heikkonen, Real-Time Swimmer Tracking on Sparse Camera Array. In Ana L. N. Fred, Maria De Marsico, Gabriella Sanniti di Baja, Pattern Recognition Applications and Methods ICPRAM 2016, Revised Selected Papers. Lecture Notes in Computer Science, 10163, 156–174, 2017.

Real-Time Swimmer Tracking on Sparse Camera Array

Paavo Nevalainen[1], M. Hashem Haghbayan[1], Antti Kauhanen[2], Jonne Pohjankukka, Mikko-Jussi Laakso[1], and Jukka Heikkonen[1]

[1] Department of Information Technology, University of Turku
FI-20014 TURKU, FINLAND
[2] Sport Academy of Turku region, Kaarinankatu 3 20500 TURKU, FINLAND
{ptneva,mohhaq,mikko-jussi.laakso,jukka.heikkonen}@utu.fi
antti.kauhanen@turku.fi

Abstract. A swimmer detection and tracking is an essential first step in a video-based athletics performance analysis. A real-time algorithm is presented, with the following capabilities: performing the planar projection of the image, fading the background to protect the intimacy of other swimmers, framing the swimmer at a specific swimming lane, and eliminating the redundant video stream from idle cameras. The generated video stream is a basis for further analysis at the batch-mode. The geometric video transform accommodates a sparse camera array and enables geometric observations of swimmer silhouette. The tracking component allows real-time feedback and combination of different video streams to a single one. Swimming cycle registration algorithm based on markerless tracking is presented. The methodology allows unknown camera positions and can be installed in many types of public swimming pools.

Keywords: athletics, swimming, body motion tracking, camera calibration, background subtraction, video processing, silhouette registration, movement cycle registration.

1 Introduction

Video systems in swimming coaching have three contradictory requirements. They should be economical to implement and operate while they have to produce adequate visualization and there should be some analysis capacity for real-time numerical feedback. This paper proposes a pipeline of video processing algorithms which are implemented already or intended to be implemented in the near future. The pipeline aims in early reduction of the video data amount to be stored and computed. It consists of a fast planar projection based on camera calibration, background filtering and swimmer tracking. The planar projection makes it possible to combine different camera streams to a single one and gives a solid physical framework for further analysis. Since swimming speed is an essential feature, speed accuracy estimation is included. The calibration based on planar projection is not conventional one, thus a comparison to an ideal pinhole model with existing camera placement uncertainty is documented.

The background filtering serves as a preparatory step in athlete tracking but also protects the intimacy of the general public. This is needed since the site (Impivaara Swimming Center, Turku, Finland) is open to the general public while the coaching sessions take place. Video recordings are used as handouts for athletes and are sometimes used as a public scientific resource.

The swimmer tracking helps to record only the swimmer and his immediate surroundings (bubble clouds, wave form at the water surface). This amounts to storing only 2 % of the original raw video data. The registration and vectorization of the swimmer silhouette in real time remains as a possible further development. This paper documents one silhouette vectorization algorithm implemented as a post-processing step. The vectorization quality has been compared to cycle registration and to the horizontal velocity records of the real-time swimmer tracking. The final aim of our project is an establishment of a video database with combination of bio-mechanical indicators, silhouette dynamics, silhouette state vector, time stamps for automatically and manually detected events and handout videos.

Starting a new site for swimming analysis requires usually considerable resources and our economical approach should be of interest to any swimming coach considering a basic computerized real-time feedback at a local site.

The rest of the paper is organized as follows. Sec. 2 is a short presentation about the contemporary research. Sec. 3 describes the site and the computing equipment. Sec. 4 presents an economical and simple calibration method based on planar projection. Also a brief comparison to mono-camera and ideal pin-hole model is provided. Sec. 5 is a proposal for real-time markerless tracking of the swimmer and it may have relevance to the swimming research community. The swimming cycle registration based on markerless tracking is proposed in Sec. 6. Sec. 7 has conclusions and discussion about the possible future developments.

2 Literature Review

The oldest approach in swimming tracking uses mechanical wire. [20] reports about measuring the force in the wire while some object is dragged behind, another method is measuring the swimmer speed directly using the wire. The mechanical method is used especially to verify the video installments.

Currently, performance analysis is based on video analysis, see e.g. a review in [19]. A typical approach is:

- to produce video stream from multiple cameras
- and detect marked or nonmarked anchor points of the body, and
- combine the trace information to a biomechanical model and visualization tools.

Several commercial tools are available, see e.g. a summary at [19]. Known examples are Dartfish [18] and Sports Motion [17].

A combination of other sensors are used in coaching, e.g. wearable accelerometers with data cache for whole length of the pool [6] and pressure pads [23] at hands.

A typical mature system is documented at [25]. An excellent analysis of real-time and post-session coaching feedback is provided. Swedish Center for Aquatic Studies has AIM (Athletes in Motion) system capable of combining views from submerged and above-water cameras, see [15]. The calibration process is close to our approach although they use striped poles while our approach is based on chessboard pattern. AIM has been developed by Chalmers and Lund Universities.

Another possibility is the virtual camera technique. A synthetical moving viewpoint is computed between adjacent cameras. The view between two stationary cameras can be interpolated as in [11]. This approach is possible in our site to eliminate the projection error between neighboring cameras. A concise method for virtual view generation is described in [26].

Video analysis without markers [12] simplifies the coaching sessions. It is much less intrusive and provides a smooth coaching process. This is also our approach. Although [12] aims for 3D body capture, their arrangement cannot be applied to a sparse camera-row easily. Our approach is more modest, 2D silhouette capture and for swimming style analysis.

A real-time human silhouette detection system is documented in [4]. In their application, the background is stable and can be recorded before the session. Our environment has potential moving objects (non-athlete swimmers sharing the same site during atheltic sessions). The aim of [4] is to estimate the center of the body of elderly people at domestic conditions. This data is then used to activity detection and classification.

Silhouette-based gait detection is the topic of [5]. The technique divides the standing or walking target to subparts for analysis. The approach is directly applicable on our field. The added difficulty comes from presence of bubbles and light scattering underwater.

The trend in research is towards 3D visualization and biomechanical models. Analysis of the recorded data is quite developed but there is room for improvement what comes to quick performance feedback by understandable performance measures.

There are mature systems which already serve the coaching activities well. The implementation seems to be rather involved requiring technical assistance, set-up times and high initial and running costs. Our approach is economical, we seek a non-intrusive rudimentary implementation with basis for further improvement.

3 The site and the system description

3.1 The site

We had only 3 cameras allocated for the project. The cameras cover 18 m over the 25 m pool length. Cameras had to be placed to underwater window sills at depth of 560 mm at the pool side. A fourth camera can be added to the grey dot depicted in Fig 1 in the future. This setup will cover the whole 25 m pool length.

The tracking plane is positioned c. 200 mm aside towards the cameras from the centerline of the swimming lane 7. The distance has been chosen so that it approximates the dimensions of the pelvis of an average-sized adult male and female.

The image mapping constructs pixel intensities directly in relation to the tracking plane. This method does not use any camera model, camera locations nor orientations. The World Coordinate System (WCS) is depicted in Fig. 1. Axis y stands for depth. The horizontal length x is oriented from the turning point towards the starting point.

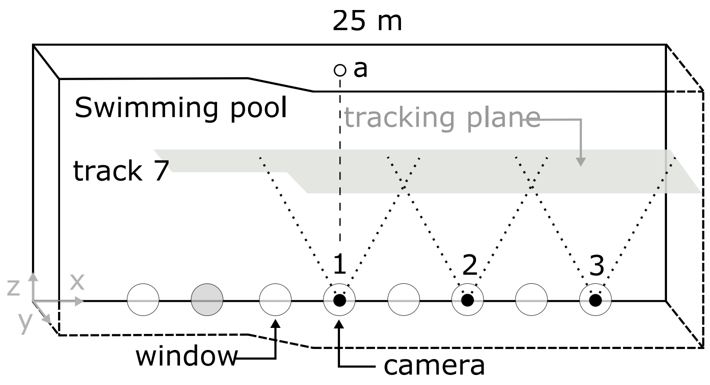


Fig. 1. The general layout of the site seen from above. The tracking plane of lane 7 is emphasized. Orientation point a of camera 1 is depicted.

3.2 System

The system consists of:

- one 2-core 3.2 GHz 64 bits computer with. 6 TB of disc space
- 3 permanently placed 50 fps cameras (Basler acA2000-50gc GigE). The image size is 750×2044. Pixel size at the tracking plane is fixed to $4 \times 4\, mm^2$. The frame time difference $\Delta t = \ 50^{-1} sec$ will be used later in this paper.

The uncompressed data from three cameras amounts to about 1GB for a 10 second clip. All cameras are synchronized so that they capture images at the same time. The time stamps are stored in the video files and they can be used in determining how to stitch the video streams together.

4 Planar Projection

Stereo and mono camera calibration methods adapted to underwater conditions are the usual choice for the calibration phase. However, on this site the actual camera locations were not precise preventing the mono-camera calibration and the camera array was too sparse for stereo camera calibration. Instead, a direct planar calibration was used, where pixels and their WCS position on the projection plane G (see Figs. 2 and 1) were sampled in such a quantity, that the rest of the pixel mapping was accomplished by interpolation.

A preliminary measurement process delivering the direct geometric mapping from pixels to global positions is documented first. The method can be categorized as an ad hoc approach answer to two demands: sparsely placed camera array and real-time video transformation. Nearest reference is [21], which uses a camera model and requires the co-planarity of the camera image plane and the tracking plane. Our method requires no camera model but can optionally use one as an interpolant.

The calibration data was gathered by floating a calibration checkerboard along the surface at the tracking plane and recording its horizontal position at each picture. The chessboard had buoys at the top and weight at the bottom. The global position x_0 of the board was measured within 10 mm accuracy std.

Next, a quick computation scheme for target pixel intensities will be derived. It is divided to a pre-computation of constants of Eq. 5 and real-time computation of Eq. 6. The following treatise refers to Fig. 2. Informally, one needs mappings F_s and F_t describing the relation from source image $\mathcal{I}_s$ and target image $\mathcal{I}_t$ to WCS. Then a functional composition $F_t \circ F_s^{-1}$ will provide the proper source image intensity $I_t(p_t)$ for a target image pixel p_t. The following definitions use somewhat unconventional definition of image as a set of pixels and a function $I(.)$ for pixel intensities because the pixel set P_s of source image is not a conventional rectangle, but merely a general subset of the full frame.

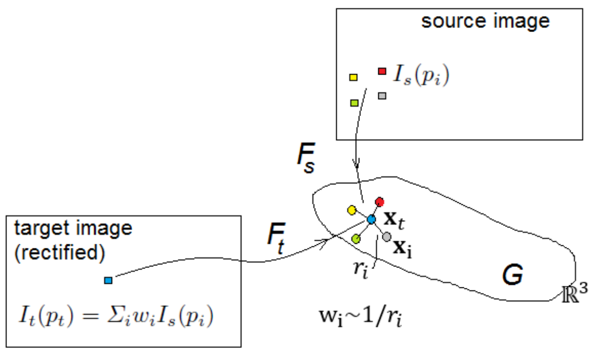


Fig. 2. Relation of source image, target image and the projection plane.

The tracking plane $G = \{\mathbf{g} \in \mathbb{R}^3 \mid (\mathbf{g} - \mathbf{g}_G) \cdot \mathbf{n}_G = 0\}$ is defined by the unit normal vector $\mathbf{n}_G$ aligned with WCS z axis and one plane point $\mathbf{g}_G = (0, 0, z_G)^T$, where $z_G = 5050\,\text{mm}$ in case of lane 7. See Fig. 1, which depicts the tracking plane G and the WCS axes x, y, z. A source image $\mathcal{I}_s = (P_s, I_s)$ has a set P_s of pixels with intensities $I_s(p)$, $p = (i, j, 1)^T \in P_s$, where i and j are conventional image pixel grid indices. Whereas source image intensities I_s are given, the target image $\mathcal{I}_t = (P_t, I_t)$ requires the geometrically rectified intensity map I_t to be computed. One can cover a conveniently chosen part of the plane G with the following mapping $F_t : P_t \to G$:

$$F_t(p_t) = \begin{pmatrix} 0 & \gamma & x_{min} \\ \gamma & 0 & y_{min} \\ 0 & 0 & z_G \end{pmatrix} p_t, \tag{1}$$

where $\gamma = 4.0\,mm$ is an arbitrarily chosen scale factor and $g_{min} \in G$ defined in Eq. 2 is a camera view specific upper left corner (UL) point. The lower right (LR) corner point g_{max} has similar definition Eq. 3. Corner points UL and LR, and the target image pixel sizes $n_t, m_t \in \mathbb{N}_+$ can be chosen freely as long as they lead to connected combined image and fulfill the constraint posed by Eq. 3:

$$g_{min} = \ (x_{min}\, y_{min}\, z_G)^T \tag{2}$$

$$g_{max} = F_t\left((n_t,\, m_t,\, 1)^T\right). \tag{3}$$

The target image size $P_t = [1, n_t] \times [1, m_t] \in \mathbb{N}_+^2$ and the the UL position g_{min} are free parameters, which are to be fixed by a practical choice so that the mapping quality is good and all camera views combine to a continuous total view. Fig. 3 shows a choice made for all three camera views.

A camera board has a checkerboard pattern. Each corner pixel p on the checkerboard on each calibration image and its corresponding global position $g \in G$ form a measurement pair $(p, \mathbf{g}) \in U$, where $p \in P_U \subset P_s$. The calibration data set U is cumulated over 37 calibration images with c. 3000 measurement pairs. The pixel samples P_U of U cover only a part of the source image pixels P_s whereas the end result of the direct plane calibration have to map whole source image onto the tracking plane using $F_s : \ P_s \to G$. In that sense this is an interpolation problem.

[2] uses two bilinear functions $f_x, f_y : \mathbb{R}^2 \to \mathbb{R}$ to map WCS coordinates x and y separately:

$$F_s(p_s) = (f_x(p_s), f_y(p_s), z_G)^T. \tag{4}$$

Specifically, for the calibration measurement set U it holds that $(p, \mathbf{g}) \in U \to F_s(p) \approx \mathbf{g}$. It is worth to mention that the above mapping F_s could have been any relatively smooth function with some sort of regularization control.

4.1 Pre-computation and post-computation

The initial problem of finding the target image content has been accomplished, except it is inconvenient to inverse F_s. Instead, one needs to match source image pixels to target image on the projection plane. There are many possibilities to this. One can e.g. use Shepard interpolation as in [2] and pre-compute the necessary neighborhood sets and weights. In that respect, $k = 4$ nearest neighbors of a point $F_t(p_t)$ amongst the point set $F_s(P_s)$ are used. The interpolation neighborhood matrix $M \in \mathbb{N}_+^{|P_t| \times k}$ and neighborhood weight matrix $W \in \mathbb{R}_+^{|P_t| \times k}$ can be pre-computed and used for the real-time pixel intensity computation:

$$I_t(p_t) = \sum_{i=1}^{k} W_{p_t i} \, I_s(M_{p_t i}), \; p_t \in P_t. \tag{5}$$

A special case $k = 1$ leads to the nearest neighbor approximation, which is very fast and requires no weights W. By denoting $C = M_{p_t 1}$ as a table of the nearest neighbor pixels of $p_t \in P_t$ at P_s, one can write the final real-time transformation as:

$$\mathcal{I}_t := (P_t, I_s(C(P_t))) \tag{6}$$

The pre-computed image mapping is simple and efficient enough for to be used in real-time. The usage of the projection plane G also makes it possible to combine each camera signal accurately to one single video, as demonstrated in Fig. 3. What comes to implementation, two image intensities rest on two memory blocks, and C is an index array.

4.2 Error analysis

The measurement set U has c. 150 mm vertical gap and c. 50 mm average horizontal distance between points. This requires the interpolant to have rather high penalty for non-smoothness.

The measurement errors for each observation $(p, \mathbf{g}) \in U$ were as follows:

- pixel p detection was done with Matlab *detectCheckerBoard.m* function, theory of which is contained in [7]. The pixel detection error is $p \approx (1, 1)$ std.
- the mechanical placement accuracy of a measured point $\mathbf{g}$ is $\Delta\mathbf{g} \approx (10, 10, 10 + 0.01\,y)^T$ mm as an approximate std.

The so called back-projection (see e.g. [22]) plot shows that the geometric mapping error is un-biased but not Gaussian. The back-projection map is not included to this paper. The final accuracy of $F_s(p_s)$ is much better than of initial data U. The geometric mapping error measure $e(p)$ is given in Eq. 7 and depicted at Fig. 3:

$$e(p) = \|\mathbf{g} - F_s(p)\|, \; (p, \mathbf{g}) \in U \tag{7}$$

Since the sample set U is of rather good quality and since the function F_s is rather smooth, the error stays almost constant even if the tuning of the interpolation is subjected to cross-validation over subsets of U. The positioning error with std. $\sigma_g \approx 1.8$ mm (see [2]) is largest in occasional points at the border and grows rapidly when extrapolating. The border areas are seldom occupied by a swimmer, though, and the problem is more of aesthetical nature. The border error can be eliminated in the future by applying a different interpolant instead of one in Eq. 4. Future calibrations will use a laser device for the horizontal position measurement, hopefully improving the visible errors occurred during the recording of the calibration data set U revealed by Fig. 3.

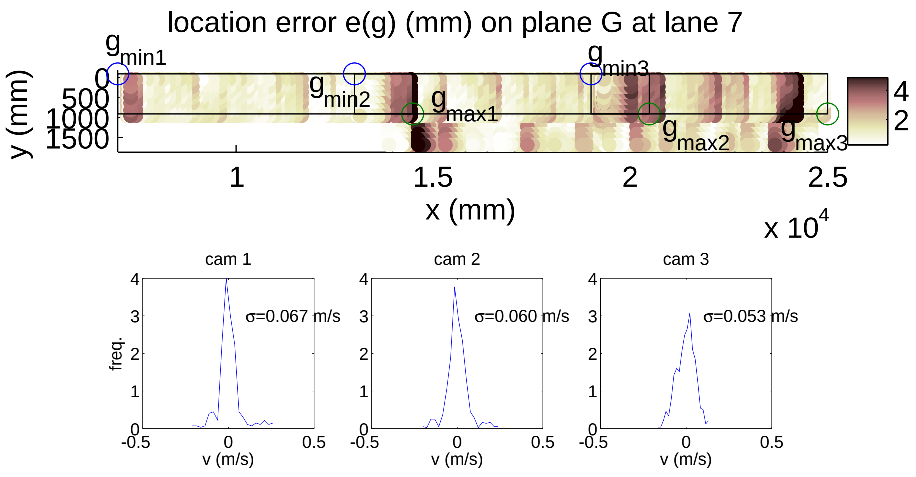


Fig. 3. Above: Error of the interpolation as a difference $e(p) = \|F_s(p) - \mathbf{g}\|, (p, \mathbf{g}) \in U$ at the calibration data set U. Also the choice of target image views P_t for each camera depicted. The color bar shows errors within the tracking area limited by view frames $(\mathbf{g}_{min\,i}, \mathbf{g}_{max\,i}), i = 1, 2, 3$. Maximum error outside the tracking area is 9 mm. Below: Speed error for a point moving at $v = 1.6m/s$. Velocity distribution is measured to all directions. The std. is 0.07 m/s.

The speed error with the result std. $\sigma_v \approx 0.07m/s$ has been estimated by assuming an observable point moving at a typical velocity $v = 1.6\,m/s = 8$ pixels/frame. An observation cannot occur at a shorter time interval than one frame, thus more pessimistic finite differences must be used in estimation. Instead of usual chain differentiation of velocity $v(p) = F_s'(p)\dot{p}\,\gamma$ where pixel speed $\dot{p}$ is assumed to be constant $\dot{p} = 8/\Delta t$, one needs to employ a probabilistic variables $p \sim \mathcal{U}(P_s)$ and $\dot{p}\Delta t \sim \mathcal{N}(8, 2 \times 0.5^2)$, where variance term comes from two additive measurements of consecutive frames with std. 0.5 pixels (≈ 2 mm according to [2]). The resulting distribution function of speed v is presented at Fig. 3:

$$v \sim \frac{F_s(p_r) - F_s(p)}{r}\dot{p}\,\gamma,\ \|p_r - p\| = r = 8,\ p_r \in P_s. \tag{8}$$

Both the location and speed errors are somewhat larger in practise, since there are always observation or registration errors involved.

4.3 Effect of inaccurate camera placement to traditional calibration

Alternative calibration methods were evaluated and tested. The best alternative was mono-camera model used as an interpolant of planar projection Eq. 4, but even that failed to reach the same accuracy as direct planar calibration. The traditional mono-camera calibration was next and the stereo-calibration worst - this was due to small overlapping of the camera view cones, see Fig. 1.

The traditional camera calibration requires an arrangement where the camera locations and orientations are known or can be measured accurately. Unfortunately, the geometric accuracy (see Table 1) technically possible on the site was not enough for traditional calibration to be competitive. Since the approaches differ so much, it is good to have a criterion for required camera positioning accuracy before mono or stereo calibration become an option.

A treatise of the application of mono-camera model combined with the plane projection follows. There are three distinct parts: pinhole camera model, coordinate transformation due the camera orientation, and projection to the tracking plane.

Pinhole camera model $Q_1^{-1} : \ P \rightarrow S_2 \subset \mathbb{R}^3$ from pixels P to camera-related projection directions on the unit sphere S_2. Literature has the camera model defined usually the opposite way as Q_1 but uses the concept in both meanings. We used Matlab Toolbox [24] and an underwater calibration chessboard.

Camera orientation of a camera c is defined by WCS orientation matrix $\mathbf{N}_c = \{\mathbf{n}_i\}, \mathbf{n}_i \in S_2$, $i = 1..3$, to be derived on the next page. The WCS coordinate base is denoted by $\{\mathbf{e}_i\}$ and camera coordinate base by $\{\mathbf{e}_{ci}\}, i = 1..3$, see Figures 4 and 1.

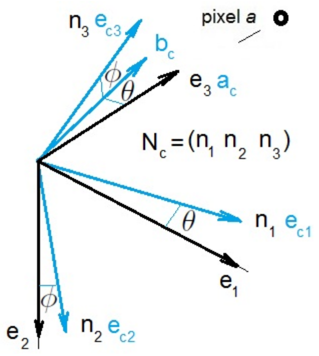


Fig. 4. The WCS orientation base $\mathbf{E}$, an orientation $\mathbf{E}_c$ specific to a camera c and column vectors of the coordinate transform matrix $\mathbf{N}_c$ depicted at the camera c. The pixel a is the global z axis marker for camera c.

The orientation is measured by an alignment marker pixel $a \in P$, which indicates the global z axis direction in the camera coordinates, see Figures 1 and 4. The following definitions are needed:

$$\begin{aligned} \mathbf{a}_c &= Q_1^{-1}(a, \boldsymbol{\alpha}) \text{ (geom. axis 3 in camera coords)} \\ \mathbf{P}_{c23} &= \mathbf{e}_{c3}\,\mathbf{e}_{c3}^T + \mathbf{e}_{c2}\,\mathbf{e}_{c2}^T \ \text{ (a projection matrix)} \\ \mathbf{b}_c &= (\mathbf{P}_{c23}\,\mathbf{a}_c)^0 \ \text{ (projecting } \mathbf{a}_c \text{ to } (\mathbf{e}_{c2}, \mathbf{e}_{c3}) \text{ plane)}, \end{aligned} \tag{9}$$

where $\boldsymbol{\alpha}$ are the pinhole model parameters achieved e.g. by Matlab toolbox [24] session and $(.)^0$ denotes vector normalization.

The above definition relies on the fact that cameras have been placed very carefully to have very small rotational error around the optical axis. This was achieved by special horizontal reference markers at the opposite side of the pool. The achieved optical axis rotation accuracy is c. $0.1 \times 10^{-3}\, rad$.

A rotation matrix $\mathbf{R}_{\mathbf{uv}}$ rotates a unit vector $\mathbf{u}$ to a unit vector $\mathbf{v}$:

$$\mathbf{R}_{\mathbf{uv}} = \mathbf{H}(\mathbf{u}+\mathbf{v})\mathbf{H}(\mathbf{u}), \tag{10}$$

where the so called line reflection matrix $\mathbf{H}(\mathbf{u}) = -\mathbf{I} + 2\mathbf{u}\mathbf{u}^T/\|\mathbf{u}\|_2$, see e.g. [28]. The rotation matrix $\mathbf{R}_{\mathbf{uv}}$ has the following necessary properties: $\mathbf{H}(\mathbf{u},\mathbf{u}) = \mathbf{I}$, $\mathbf{H}(\mathbf{u},\mathbf{v})\mathbf{u} = \mathbf{v}$, $\mathbf{w}\cdot\mathbf{u} = \mathbf{w}\cdot\mathbf{v} = 0 \rightarrow \mathbf{R}(\mathbf{u},\mathbf{v})\mathbf{w} = \mathbf{w}, \mathbf{R}(\mathbf{u},(\mathbf{u}+\mathbf{v})^0)^2 = \mathbf{R}(\mathbf{u},\mathbf{v})$.

A coordinate transformation matrix $\mathbf{N}_c$ can now be based on the definitions of Eq. 9 and two consecutive Euler rotations $\mathbf{R}_\phi$ and $\mathbf{R}_\theta$ depicted in Fig. 4:

$$\mathbf{R}_\phi = \mathbf{R}_{\mathbf{e}_{c3}\,\mathbf{b}_c} \tag{11}$$

$$\mathbf{R}_\theta = \mathbf{R}_{\mathbf{b}_c\,\mathbf{a}_c} \tag{12}$$

$$\mathbf{N}_c = (\mathbf{R}_\theta\, \mathbf{R}_\phi)^{-1}. \tag{13}$$

The final transformation from coordinates specific to a camera c to WCS equals:

$$\mathbf{n} = \mathbf{N}_c\, \mathbf{n}_c \tag{14}$$

where $\mathbf{n}_c = Q_1^{-1}(p, \boldsymbol{\alpha}), p \in P$ is the camera-specific orientation of a pixel p from the previously addressed pinhole model.

Projection: The final map $Q_2 : \mathbf{n} \rightarrow \mathbf{g} \in G$ is the projection to the tracking plane G:

$$l = (z_G - g_c \cdot \mathbf{e}_3)/(\mathbf{n}\cdot\mathbf{e}_3) \;\; \text{(projection length)} \tag{15}$$

$$g = g_c + l\mathbf{n} \in G \;\; \text{(on-line condition)} \tag{16}$$

The projection length l has been solved from the in-plane condition: $g \in G \rightarrow g \cdot \mathbf{e}_3 - z_G = 0$.

The total mapping $F_{sM} = Q_2 \circ \mathbf{N}_c \circ Q_1^{-1} : P \rightarrow G$ for mono-camera has been now constructed. By assuming normal distributed measurements of Table 1 and normal distributed pinhole model parameters from Matlab Toolbox [24], one gets the pixel placement accuracies listed in Table 2. Due to nonlinearities in Eqs. 9... 16, the resulting distribution of tracking plane position $g \in G$ is not normal distributed. Std. values are used in Table 2 to make comparisons possible. Computations were done by python Sympy package [27] by coding Eqs. 9... 16 with their corresponding variance terms (first five values) of Table 1. $\varDelta\mathbf{N}$ at the last row of Table 1 stands for the maximum angular orientation variance at $\mathbf{e}_3$ direction and it is a result, not an input for the computation.

The ideal camera placement is exact, whereas the current accuracy is within 10 mm (std.). The ideal camera model is impossible but it can be modelled with Eqs. 9 by assuming the pixel specific local direction vector $\mathbf{a}_c$ be given. All existing camera models are between the ideal one and the Matlab Toolbox, which is based on [7] and [16].

Table 1. Error terms (1 std.) of the mono-camera calibration.

Term	Explanation	value
Δa	direction pixel error	2.0 pix.
Δp	chessboard pixel error	0.9 pix.
Δz_G	tracking plane position	10 mm
Δg_c	real global camera position (*)	(10,10,15) mm
Δg_c	ideal global camera position (**)	(0,0,0) mm
$\Delta \mathbf{N}$	global camera orientation	0.4×10^{-3} rad

The accuracy is clearly limited by the camera placement if direct calibration is not utilized. Also, the limiting factor in direct calibration is horizontal position measurement, which can be improved in the future.

Table 2. The effect of camera placement inaccuracy. The projection calibration, an ideal and a real mono-camera model accuracy Δg compared. Values are std. in vertical and horizontal direction.

Method	no pos. error (*) (mm)	real pos. error (**) (mm)
direct calibration	2.0 ... 7.0	2.0 ... 7.0
ideal camera model	3.2 ... 5.2	13 ... 15
mono-calibration	3.4 ... 10	15 ... 21

5 Silhouette Tracking

The real-time swimmer tracking method is based on finding the horizontal pixel translation which minimizes the intensity difference of two sequential images. There are three caveats in this simplistic approach though:

1. The background does not move and causes a strong slow-down bias. In our chosen approach, we use the difference of two consecutive images as a basis of further computations, thus eliminating this effect.
2. Hands are constantly propelling the swimmer forwards with a backwards stroke aside the body. This causes similar slow-down bias, but only locally. This error source can be eliminated by dividing the observed area to e.g. 3×6 subparts as depicted in Fig. 5 and taking the average of the majority translational shift observed.
3. The resulting speed or horizontal position as a function of time cannot be directly associated to any particular body part. This problem can be solved only by a secondary tuning at the post-processing phase.

The swimmer tracking succeeds in keeping the swimmer at focus, thus enabling the reduction of the video record size. Bio-mechanical indicators e.g. accurate speed of head, pelvis etc. have to be measured at the post-processing phase. The inexact preliminary speed measurement does not prevent later additive corrections!

The algorithm to find the horizontal shift $u^* \in \mathbb{R}$ (in pixels) between two sequential images $\mathcal{I}_1$ and $\mathcal{I}_2$ is presented. It is based on minimizing the following simple and fast dissimilarity measure:

$$d_A(\mathcal{I}_1, \mathcal{I}_2) = \sum_{p \in P_1 \cap P_2} |I_1(p) - I_2(p)| / |P_1 \cap P_2|, \tag{17}$$

where the summation is applied to the overlapping part of two images or sub-images after a possible horizontal translation explained later in the text. Since the athlete silhouette has a slight shape change between two frames, the dissimilarity minimization is made by sub-images, and only the most similar sub-image pairs will contribute to the decision about the effective horizontal translation happening between frames. This limited set of sub-images is detected by their deep local minima of $d_A(.,.)$ measured by the second order derivative (or its approximant) at the minimum. The algorithm is designed to be of $O(|P_t|)$ (constant time for a pixel), and it allows simple task-parallelization per each sub-window, although it is intended to a single processor. It requires one parameter, the a priori value u_0 for the vertical speed.

Following conventions have been used in the algorithm: $\mathcal{I}(\Delta i)$ is an image $\mathcal{I}$ shifted by Δi pixels horizontally along the current swimming direction. $\mathcal{I}_{i,k}$ is a k'th sub-image of an image $\mathcal{I}_i$. $g(u)$ is a low-order polynomial fit to a data set $\{(i, f(i)) |\ i \in \text{a short interval} \subset \mathbb{N}_+\}$, e.g. of degree 2 or 3.

```
Data: Sequential images $\mathcal{I}_1$ and $\mathcal{I}_2$, and the previous horizontal
      translation $u_0 \in \mathbb{R}_+$
Result: Translation $u^* \in \mathbb{R}_+$ which best fits sub-images $\mathcal{I}_{.k}$
forall $\mathcal{I}_{1,k} \subset \mathcal{I}_1$ do
    $f(i) \equiv_{df} d_A(\mathcal{I}_{1,k}(i), \mathcal{I}_{2,k}(0))$ see Eq. 17;
    $i_0 \leftarrow \lfloor u_0 \rfloor$ truncated value;
    $i_k \leftarrow \operatorname{argmin}_{i \in [i_0 - \Delta u, i_0 + \Delta u]} f(i)$ optim. evaluations based on local convexity;
    $g(u) \equiv_{df} f(i)$ interpolated to continuous $u \in \mathbb{R}_+$;
    $u_k \leftarrow \operatorname{argmin}_{u \in [i_k - \Delta u, i_k + \Delta u]} g(u)$;
    $w_k \leftarrow \frac{d^2 g}{du^2}(u_k)$ approximative or exact second derivative;
end
$u^* \leftarrow \text{mean}_{k \in [1,4]} u_k$ when ordered by descending $w_k$;
```

Algorithm 1: Finding the optimal horizontal pixel shift between two images.

There is a faster, non-interpolating and more inexact version of this algorithm using discrete value i_k directly and approximating the importance w_k of a sub-image k by the second order difference $D^2(.)$ with a unit step size:

$$\begin{aligned} u_k &\leftarrow i_k \\ w_k &\leftarrow f(i_0 - 1) - 2f(i_0) + f(i_0 + 1) = D^2(i_0). \end{aligned} \tag{18}$$

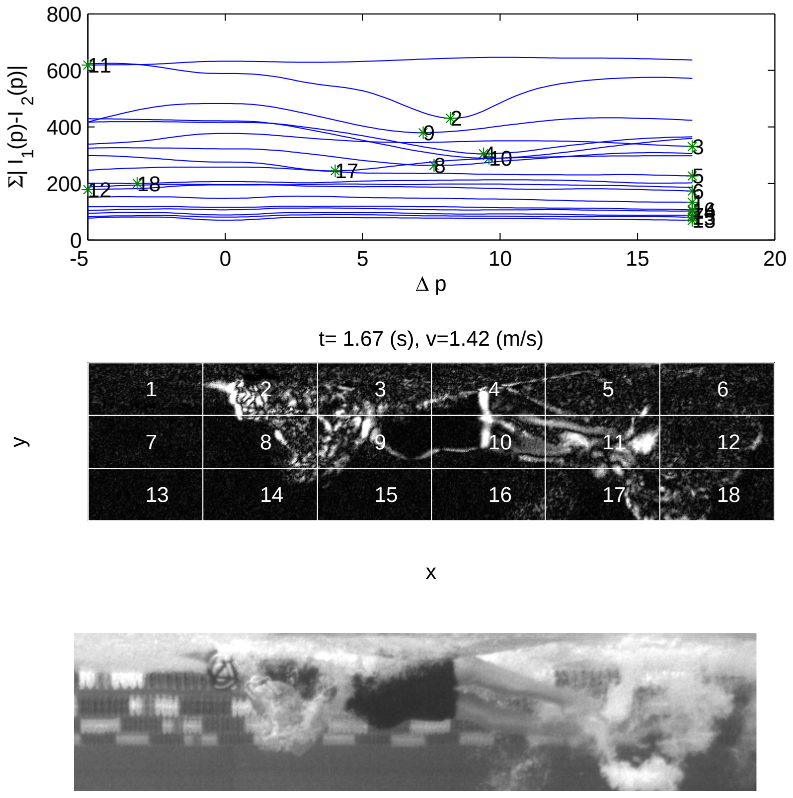


Fig. 5. Above: Horizontal pixel translation Δp in for the optimal match at each sub-window. Middle: Division of the camera view to sub-images $\mathcal{I}_{i,k}$, $k \in [1, 18]$. The background is neutral since the difference of two sequential images has been used for the swimmer tracking. Below: Original video frame (G channel of the RGB signal).

A typical speed of a female backstroke swimmer is $u_0 = 8...9$ pixels/frame ($\approx$ 1.5...1.7 m/s). This speed is specific to abilities of the athlete, swimming style, sex and age. It serves as an initial guess for the algorithm 1 when first two images are being processed. The later image pairs will use the latest observed u^* as the initial guess. Two sequential images are then divided to sub-image pairs $\mathcal{I}_{\cdot k}$, which have in some cases unique optimal horizontal shift u_k (in pixels). The shifts are made in exact pixels i, and usually only few trials are needed to find the best pixel fit $i_k = \underset{i}{\operatorname{argmin}} f(i)$ when starting the search from the truncated initial guess $i_0 = \lfloor u_0 \rfloor$. The continuous counterpart $g(u)$ of the image fit function $f(i)$ is evaluated only locally at a short interval, and a low-order 2nd or 3rd order polynomial is enough to estimate the quality of the fit. Fig. 5 depicts more generic splines in order to illustrate the regularity of the local fit.

If two sub-images resemble each other, they provide more useful information about the translation than other sub-image pairs, and should be preferred. The quality of the fit is reflected by the depth of the local minimum. E.g. sub-images 2,4,9 and 10 in Fig. 5 indicate the horizontal speed best and are chosen to form the averaged horizontal shift u^*.

The momentary horizontal velocity of the swimmer is now $v(t) = u^*(t)\,\gamma/\Delta t$ where $\gamma = 4\,mm$ is the geometric pixel size from Eq. 1 and time $t = i\Delta t$ for an image $\mathcal{I}_i$. Fig. 6 depicts the resulting tracking speed. Individual strokes can be registered but the plot would require some smoothing. The current system has a real-time velocity plot based on marker-tracking, but at the moment it seems to have poorer quality. There are further improvements possible to Alg. 2 in order to produce better tracking speed plot, but similar improvements can be achieved in the post-processing phase, as well. E.g. a Kalman filtering employed in [2] can be used.

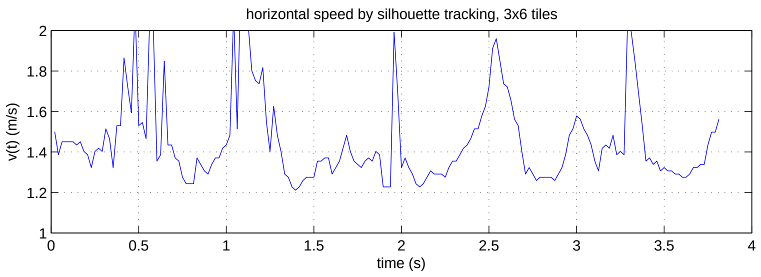


Fig. 6. Silhouette tracking. Above: Horizontal translation for the optimal match at each subpart. Below: Division of the camera view to subparts I_k, $k \in [1, 18]$. The background is black since the difference image is being used for the swimmer tracking.

6 Swimming Cycle Registration

There are several alternatives for swimming cycle registration. One can detect pikes from the velocity curve of Fig. 6 and produce a visual dissimilarity graph based on velocity histories of each swimming stroke as in [2].

Another alternative is silhouette vectorization, which would provide access to the state space of the performance. This approach would open the door to fully nonlinear dynamics analysis.

Third alternative uses the translational sub-image dissimilarity demonstrated in Alg. 2 to find the closest match from the frames which exist approximately one estimated cycle duration away from the current frame. This gives a continuous cycle duration measure. The next matching frames form a detectable path in the image dissimilarity matrix D depicted in Fig. 7. The definition of the dissimilarity matrix D follows:

$$D = \{d(\mathcal{I}_i, \mathcal{I}_j)\}_{i,j \in 1..n}, \tag{19}$$

where n is the number of video images over one length of the pool. There are

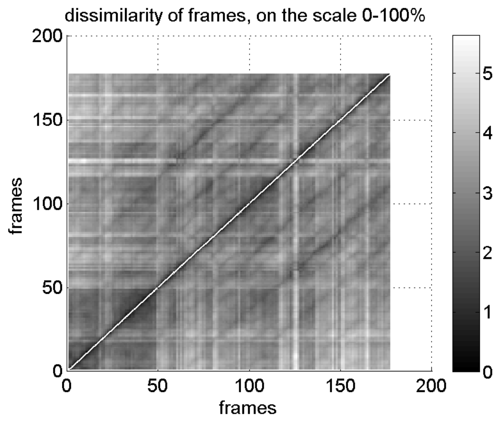


Fig. 7. Translational dissimilarity of full frames in a swimming performance with a gliding phase (frames 1-50) and backstroke phase (frames 51...). Exceptional events are clearly visible. 100 % dissimilarity means difference between completely black and completely white images.

several possibilities for the dissimilarity $d(.,.)$ which will be explored next. Fig. 5 has been produced by matching whole images with only horizontal translation allowed, while left part of Fig. 6 has been produced using sub-images and fully free translation. An algorithm implementing a dissimilarity measure $d_B(\mathcal{I}_i, \mathcal{I}_j)$ based on extent of minimizing free translational shift is presented next. The previous notion of horizontally shifted image $\mathcal{I}(i)$, $i \in \mathbb{N}_+$ is now extended to completely free translation $\mathcal{I}(\mathbf{i})$, $\mathbf{i} \in \mathbb{N}^2$. Also, previous interval search for minimization is substituted by square area search over $\text{box}((u_x, u_y), \Delta u) = [u_x - \Delta u, u_x + \Delta u] \times [u_y - \Delta u, u_y + \Delta u] \subset \mathbb{R}^2$, $u_x, u_y \in \mathbb{R}$. A similar $\text{box}(\mathbf{i}, \Delta u) \subset \mathbb{N}^2, \mathbf{i} \in \mathbb{N}^2$ has a corresponding discrete domain.

```
**Data**: Two images or sub-images $\mathcal{I}_i$ and $\mathcal{I}_j$ and the search limit $\Delta u =$ 2.
**Result**: the image dissimilarity $d_{ij} = d_B(\mathcal{I}_i, \mathcal{I}_j)$ based on the norm of
            minimizing 2D pixel translation.
$f(\mathbf{i}) \leftarrow d_A(\mathcal{I}_1(\mathbf{i}), \mathcal{I}_2(0))\,,\ \mathbf{i} \in \mathbb{N}^2$ ;
$\mathbf{i}^* \leftarrow \operatorname*{argmin}_{\mathbf{i} \in \text{box}(\mathbf{0}, \Delta u)} f(\mathbf{i})$ optimizing evaluations based on local convexity;
$g(\mathbf{u}) \equiv_{def} f(\mathbf{i})$ interpolated to continuous $\mathbf{u} \in \mathbb{R}^2$;
$\mathbf{u}^* \leftarrow \operatorname*{argmin}_{\mathbf{u} \in \text{box}(\mathbf{i}^*, \Delta u)} g(\mathbf{u})$;
$d_{ij} \leftarrow \|\mathbf{u}^*\|_2$
```
Algorithm 2: Finding the extent of an optimal pixel shift between two images.

Another variant of Alg. 2 defines dissimilarity $d_C(.,.)$ with translations $\mathbf{u}$ restricted to horizontal shift only. Yet another dissimilarity applies to whole images only:

$$d_D(\mathcal{I}_i, \mathcal{I}_j) = \underset{\mathcal{I}_{1k} \in \mathcal{I}_1}{\text{mean}}\, d_C(\mathcal{I}_{1k}, \mathcal{I}_{2k}), \tag{20}$$

which has a possible variant restricting the value set used for averaging by a similar interpolation and weight strategy as used in Alg. 1.

The following cycle duration algorithm assumes the dissimilarity matrix D partially pre-computed using the dissimilarity variant $d_D(.,.)$ of Eq. 20 so that it contains the path of best matching future frames. This can be ensured by computing a diagonal stripe of D using some practical margin width, say 15 frames. The algorithm then traces the path to produce the cycle duration plot on right side of Fig. 8. First, the best matching future frame $\mathcal{I}_{i+j_i}$ is searched for a frame $\mathcal{I}_i$. Then, an interpolation function $g(.,.)$ is based on a modest sample set of neighboring dissimilarity values of D to find a best matching cycle duration u_i. This algorithm produces interpolated frames $\mathcal{I}_{i+u_i}$, $u_i \in \mathbb{R}_+$, from where the cycle duration $T(t) = u_i \Delta t$, $t = i\Delta t$ can be derived:

```
Data: Dissimilarity matrix $D$, image index $i$ and the previous cycle
      duration $u_{i-1} \in \mathbb{R}_+$ or a priori $u_0$
Result: Duration $u_i \in \mathbb{R}_+$ which best fits frame pairs $\mathcal{I}_i, \mathcal{I}_{i+u_i}$.
forall $i \in 1...n$ do
    $j_0 \leftarrow \lfloor u_{i-1} \rfloor$;
    $j_i \leftarrow \operatorname{argmin}_j D_{ij}$ starting the search from $j_0$;
    $g(v,u) \leftarrow D_{ij}$ interpolated to continuous $u, v \in \mathbb{R}$ using earlier
    computed values of $D$ in a neighborhood box$((i, j_k), \Delta u)$;
    $u_i \leftarrow \operatorname{argmin}_{j_k - \Delta u \le u \le j_k + \Delta u} g(i,u)$ a local search with $\Delta u = 3$;
end
```

Algorithm 3: Finding the optimal match over the swimming cycle duration.

As before, a faster and less accurate version can be provided by bypassing the interpolation step: $u_i \leftarrow j_i$. It seems, that a practical choice is to use the latter less accurate model and use e.g. Kalman smoothing (see e.g. [8]) with a physically feasible point mass and damping coefficients to produce the final velocity plot of Fig. 6 and the cycle period graphs depicted in Fig. 8.

7 Conclusions

A principal objective of the system is to provide immediate trainer feedback. To achieve that in the early phase of the project, the implementation has been based on the following principles:

- The analysis is limited to a monotonic stage of the athletic performance. Detection of stage changes like from gliding to stroking remains a future problem. At the moment the analysis is triggered manually. Several swimming styles are supported, though.
- A real-time marker tracking has been implemented. It will be substituted by the markerless tracking in the future.

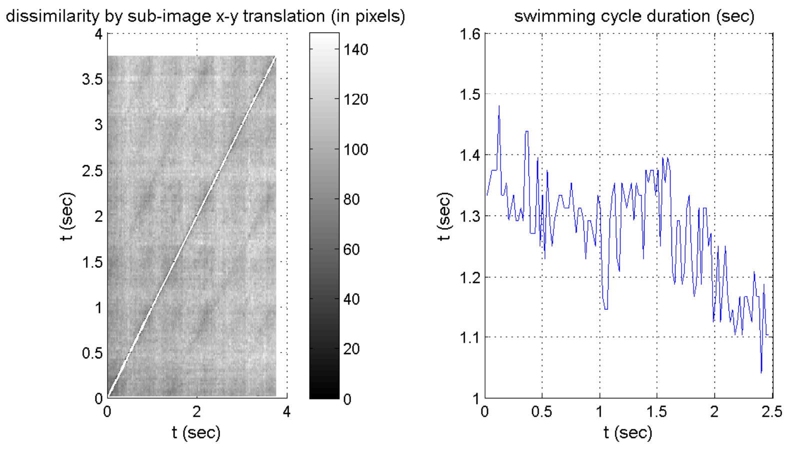


Fig. 8. Left: The frame dissimilarity matrix D based on the dissimilarity measure $d_D(.,.)$. Both horizontal and vertical translations are allowed. See Alg. 3. The cycle length shows as diagonal stripes. Right: The cycle duration based on the dissimilarity matrix at left. The graph shows how the athlete shifts from slide mode to stroke mode. The graph contains noise, since the interpolation routine of Alg. 3 was by-passed for quality assurance.

– The geometric mapping of images, markerless silhouette tracking, seamless combination of 3 camera signals, tracked velocity visualization and cycle regularity visualization are real-time processes, which will quite likely perform at 25 fps (for every second frame).
– silhouette capture and vectorization, separate legs-vs-hands stroke analysis, biomechanical analysis and various problems about clustering athletes by their swimming technique are all left as a post-processing step.

As the the execution order of the Alg. 1 is $O(1)$ per pixel, and there are c. 100 processor instructions per pixel, it is estimated that the execution time will be very small and that the intended computing equippement [1] can handle the tracking of 25 fps video input. This means two processors will be capable of real-time geometric transform [2] and tracking task with c. 25 fps i.e. every second frame are subjected to computation and every second frame is interpolated with a simple instruction set. The computation tasks are inherently parallel, facilitating upgrade to system speed 50 fps.

A simple pipeline of real-time swimmer tracking, which requires relatively modest computational arrangements and is able to provide immediate coaching feedback, has been presented. The analysis is limited to projective 2D plane, and it is our hope that it will be capable of silhouette vectorization and further bio-mechanical analysis. The camera calibration process adapts to the economically feasible sparse camera array arrangement and to poor camera placement accuracy. A procedure to decide between conventional camera calibration and direct projection plane calibration has been presented.

At the moment the system performance has not been validated against contemporary techniques and technologies. A new approach avoiding many pitfalls of computationally heavy operations has been proposed, and further improvements depend on the feedback of the research community with long traditions and experience on this field. The concrete micro-array chip implementation of algorithms will proceed later, and we remain optimistic that the result will have practical value.

The current system will be upgraded by a fourth camera at the location indicated by a grey circle in Fig. 1. The video monitoring would then span whole the pool length.

There are several alternatives to algorithms used to track the swimming speed. One can preprocess or vectorize the images by FFT etc. methods. These alternatives were not of interest now, since we aim at a video database of large number of athletic sessions with correctly focused video frames in order to cumulate samples for further application of Machine Learning methodologies, e.g. for swimming gait evaluation.

The geometric mapping of the video image is fast while it maintains enough signal quality for markerless tracking algorithms. An interesting alternative interpolant for the mapping F_s of Eq. 4 is the best available water-plexiglass-air camera model as described in [9]. The same model has been applied e.g. in [15]. The model requires more tuning parameters than e.g. [24]; camera position and orientation, plexiglass thickness and refraction coefficient, and the air length between camera and plexiglass must be fit by mean square error minimization of Eq. 7. This approach will be attempted in the future.

We hope that proposals and demonstrations of this paper stir up some interest in the research community. We aim at a combination of simple and feasible methods, and the current set of algorithms lacks only a high-quality silhouette registration with a simple real-time version to detect swimming phase changes and events.

Acknowledgements. The project is a joint venture of University of Turku IT department and Sports Academy of Turku region and it has been funded by city of Turku, National Olympic Committee, Finnish Swimming Federation, Urheiluopistosäätiö and University of Turku. Machine Technology Center Turku Ltd. participated to instrumentation.

References

1. ARM Cortex-A7 Processor. http://www.arm.com/products/processors/cortex-a/cortex-a7.php, (2016)
2. Nevalainen, P., Kauhanen, A., Raduly-Baka, C., Heikkonen, J.: Video based Swimming Analysis for Fast Feedback, In: Proc. of Intern. Conf. on Pattern Recognition Applications and Methods (ICPRAM2016), x–xx, (2016)
3. Palshikar, G., K.: Simple Algorithms for Peak Detection in Time-Series. In Proc. 1st Int. Conf. Advanced Data Analysis, Business Analytics and Intelligence (2009)
4. Christodoulidis, A., Delibasis, K. K., Maglogiannis, I.: Near Real-time Human Silhouette and Movement Detection in Indoor Environments Using Fixed Cameras. In:

Proceedings of the 5th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA ’12), (2012)

5. Choudhury, S.,D., Tjahjadi, T.: Silhouette-based gait recognition using Procrustes shape analysis and elliptic Fourier descriptors, In: Pattern Recognition, 45, 3414-3426, (2012)
6. Dadashi, F., Millet, G., Aminian, K.: Inertial measurement unit and biomechanical analysis of swimming: an update. Sportmedizin, Vol. 61, 21–26, (2013)
7. Zhang, Z.: A Flexible New Technique for Camera Calibration, In: IEEE Transactions on Pattern Analysis and Machine Intelligence, 22, 13301334, (2000)
8. Hartikainen,J., Seppänen, M., Särkkä, S.: State-Space Inference for Non-Linear Latent Force Models with Application to Satellite Orbit Prediction, In: CoRR, (2012)
9. Sedlazeck, A., Koch, R.: Perspective and Non-Perspective Camera Models in Underwater Imaging - Overview and Error Analysis. In: Theoretical Foundations of Computer Vision. LNCS, vol. 7474, 212-242, (2011).
10. Siirtola, P., Laurinen, P., Roning, J., Kinnunen, H.: Efficient accelerometer-based swimming exercise tracking, In: IEEE Symp. on Computational Intelligence and Data Mining (CIDM), 156-161, (2011)
11. Makoto, H., S., and Kimura, M., Yaguchi, S., Inamoto, N.: View Interpolation of Multiple Cameras Based on Projective Geometry, In: International Workshop on Pattern Recognition and Understanding for Visual Information, (2002)
12. Ceseracciu, E.: New frontiers of markerless motion capture: application to swim biomechanics and gait analysis, Padova University, (2011)
13. James, D., A., Burkett, B. Thiel, D., V., An unobtrusive swimming monitoring system for recreational and elite performance monitoring, In: Procedia Engineering, 5th Asia-Pacific Congress on Sports Technology (APCST), vol. 13, 113–119, (2011)
14. Pansiot, J., Lo, B., Guang-Zhong, Y.: Swimming Stroke Kinematic Analysis with BSN, In: Body Sensor Networks (BSN), 2010 International Conference on, (2010)
15. Haner, S., Svärm, L., Ask, E., Heyden, A.: Joint Under and Over Water Calibration of a Swimmer Tracking System, In: Proceedings of the International Conference on Pattern Recognition Applications and Methods, 142–149, (2015)
16. Heikkilä, J., Silven, O.: A four-step camera calibration procedure with implicit image correction, In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 1106-1112, (1997)
17. Sportsmotion: A motion analysis system, http://www.sportsmotion.com/, (2011-2015)
18. Dartfish: Dartfish video analysis tool, http://www.sportmanitoba.ca/page.php?id=116, (2011-2015)
19. Kirmizibayrak, J., Honorio, J., Xiaolong, J., Russell, M., Hahn, J., K.: Digital Analysis and Visualization of Swimming Motion. The International J. of Virtual Reality, Vol. 10, 9–16, (2011)
20. Bideau, B.: Biomechanics and Medicine in Swimming IX. In: Chatard J-C. (ed.), IXth International World Symposium on Biomechanics and Medicine in Swimming. 52-53, (2003)
21. Luo, H.-G., Zhu, L.-M., Ding, H.: Camera calibration with coplanar calibration board near parallel to the imaging plane. Sensors and Actuators A: Physical, 132, 480–486, (2006)
22. Kannala, J., Heikkilä, J., Brandt, S., S.: Geometric Camera Calibration. In: Wiley Encyclopedia of Computer Science and Engineering, (2008)
23. Bottoni, A., et al. Technical skill differences in stroke propulsion between high level athletes in triathlon and top level swimmers. In: Journal of Human Sport and Exercise, Vol. 6, No. 2, 351–362, (2011)

24. Bouguet, J. Y., Camera calibration toolbox for Matlab, (2008)
25. Mullane, S., L., Justham, L., M., West, A., A., Conway, P., P.: Design of an end-user centric information interface from data-rich. In: Procedia Engineering, Vol. 2, 2713–2719, (2010)
26. Martin, N., Roy, S.: Fast view interpolation from stereo: Simpler can be better. In: Proceedings of 3DPTV'08. The Fourth Intern. Symp. on 3-D Data Processing, Visualization and Transmission, Georgia Institute of Technology, Atlanta, GA, USA, (2008)
27. Rocklin, M., Terrel, A.,R.: Symbolic Statistics with SymPy. In: Computing in Science & Engineering, Vol. 14, No. 3, 88-93, (2012)
28. , Dorst, L., Fontijne, D., Mann, S.: Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry, (2007)

This is an expansion of the work presented in [2]. The following novelties have been added:

- horizontal speed error analysis in Sec. 4.2,
- comparison of the direct planar calibration and conventional calibration methods in the presence of the camera position and orientation uncertainty in Sec. 4.3 and Table 2,
- a real-time method for markerless swimmer tracking over multi-camera array in Section 5. This is the main result of this paper. Earlier version in [2] used marker tracking, which is considerably easier task.
- Swimming cycle registration based on markerless tracking on Section 6. Method is based now on comparison of image pair differences instead of mere tracker speed curve analysis.

Publication P5

Triangular Curvature Approximation of Surfaces - Filtering the Spurious Mode

P. Nevalainen, I. Jambor, J. Pohjankukka, J. Heikkonen, T. Pahikkala, Triangular Curvature Approximation of Surfaces - Filtering the Spurious Mode. In: Maria De Marsico, Gabriella Sanniti di Baja, Ana L. N. Fred (Eds.), Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2017, 457–466, ScitePress, 2017.

Triangular Curvature Approximation of Surfaces
Filtering the Spurious Mode

Paavo Nevalainen[1], Ivan Jambor[2], Jonne Pohjankukka[1], Jukka Heikkonen[1] and Tapio Pahikkala[1]
[1]*Dept. of Information Tech., Univ. of Turku, FI-20014 Turku, Finland*
[2]*Dept. of Diagnostic Radiology, Univ. of Turku, FI-20014 Turku, Finland*
{ptneva, ivjamb, jonne.pohjankukka, jukhei,tapio.pahikkala}@utu.fi

Keywords: Curvature Spectrum, Parameterless Filtering, Irregular Triangulated Networks, Discrete Geometry.

Abstract: Curvature spectrum is a useful feature in surface classification but is difficult to apply to cases with high noise typical e.g. to natural resource point clouds. We propose two methods to estimate the mean and the Gaussian curvature with filtering properties specific to triangulated surfaces. Methods completely filter a highest shape mode away but leave single vertical pikes only partially dampened. Also an elaborate computation of nodal dual areas used by the Laplace-Beltrami mean curvature can be avoided. All computation is based on triangular setting, and a weighted summation procedure using projected tip angles sums up the vertex values. A simplified principal curvature direction definition is given to avoid computation of the full second fundamental form. Qualitative evaluation is based on numerical experiments over two synthetical examples and a prostata tumor example. Results indicate the proposed methods are more robust to presence of noise than other four reference formulations.

1 INTRODUCTION

Wide-scale point clouds have become accessible to analysis everywhere. The point cloud surface registration typically has an approximate or accurate Delaunay triangular or tetrahedral mesh generation as a preliminary step. The surface models are called irregular triangularized networks (TIN) for historical reasons. The application domains can be roughly divided to three categories by the target environment: built environment, natural resource data and medical 3D imaging.

The ratio $0 \leq \sigma_h/r \leq 0.3$ of the perpendicular noise component σ_h and the nominal surface radius r describe the difficulty of curvature registration. The built environment data has usually high point density and small noise ratio compared to the natural resource data (Mitra and Nguyen, 2003). Built surfaces are usually solid, curvature values change slowly over distance, and it is desirable to be able to detect the local curvature accurately. A typical mean curvature method for such data is based on the Laplace-Beltrami (L-B) operator (Meyer et al., 2003; Mesmoudi et al., 2012).

Other two application domains have the noise ratio much higher, approximately $\sigma_h/r = 10^{-2}...10^{-1}$ (Schaer et al., 2007). Surfaces, e.g. the terrain surface, are porous, covered with vegetation or mathematically undefined. Natural resource data is gatherer by aerial light detection and ranging (LiDAR) or by spatial photogrammetry. Natural resource data has shape recognition tasks where the point samples per target ratio reaches one (Nevalainen et al., 2016), i.e. one single elevated hit is a possible target (e.g. a surface stone), see Fig. 1. Detection of an individual target is naturally uncertain in the presence of noise, but one can cluster larger areas e.g. to stony or non-stony ones using e.g. the curvature spectrum (Nevalainen et al., 2015). On the other hand, there is a natural frequency limit defined by the nominal mesh length. Excitation of this frequency over a large area (see Fig. 1 left side) is

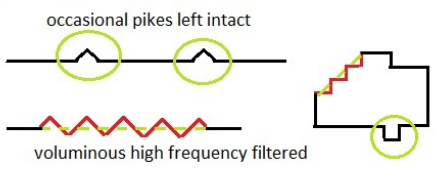

occasional pikes left intact
voluminous high frequency filtered

Figure 1: The voluminous highest noise component arises either from the scanning process (LiDAR point clouds, left), or from the voxel granularity (right). Neither case should require any parameters to filter. Occasional anomalies (circles) should be transferred intact to pattern recognition phase.

usually a numerical artifact which should be filtered at some point of the processing.

Medical 3D applications, especially magnetic resonance imaging (MRI), often have non-isotropic voxels causing excitation at the frequency limit, see right part of the Fig. 1. Numerical methods should be resilient to effect of noise, low sampling and discretization patterns.

Surface registration is reminiscent of interpolation, whereas noise reduction is filtering. Typically, these two operations can be performed in any order, or combined together. Spatial filtering requires several parameters, and it is worthwhile to seek curvature registration methods, which would handle the highest frequency as depicted in Fig. 1:

1. leaving single pikes to be handled by later pattern recognition and filtering steps.
2. eliminating automatically large excitations of the highest frequency.

Naturally, if such a behaviour is squarely against the needs of a specific application, there is an abundant supply of existing curvature registration methods, which should be employed instead. Alternative methods have several opposing properties for discrete differential operators (Wardetzky et al., 2007) used as building blocks for curvature analysis. If a new application field arises, one has to be aware of the trade-offs between different properties.

The Gaussian and mean curvature completely define the local curvature of any continuous surface. It is the consensus of the current research that the local Gaussian curvature is best estimated on TIN models by so called angle deficit (see e.g. (Crane et al., 2013), and the result is robust to noise. This reference method is named as vertex Gaussian in this paper.

This paper uses the classical differential geometric definition of the average Gaussian curvature (Pressley, 2010): it is the ratio of the total orientation change over a surface area, a TIN triangle in this case. It is pointed out in Sec. 3.1 that this simple definition leads to a triangular Gaussian curvature estimation on vertices, which fills the requirements 1 and 2 mentioned before.

The mean curvature is numerically more difficult target than Gaussian curvature. One starting point for computation has been neglected in the literature thus far. It is possible to define the mean curvature by the local rate of change of the surface area when the surface is mapped towards the direction of its unit normal (Pressley, 2010). This definition is related to the theory of minimal surfaces and it can be applied directly to the triangulated surface with defined vertex normals. Also this novel mean curvature formulation has the earlier mentioned properties 1 and 2, as ruminated in Sec. 3.1.

The rest of the paper has the following structure: Section 2 introduces the triangular Gaussian and mean curvatures, and a collection of competing definitions. Also the problem of finding the principal curvature direction has been addressed there. Section 3 has a practical example (prostata tumor), and two synthetical test cases to verify the properties 1 and 2 of the proposed method. Section 4 brings in the conclusions.

2 TRIANGULAR CURVATURE

The following notation will be used throughout the presentation. The set of cloud points $\mathcal{P} \subset \mathbb{R}^3$ is given. A triangle $t = (a,b,c), a,b,c \in \mathcal{T} \subset \mathcal{P}^3$ is defined by three vertex points which can be referred in cyclic fashion in counterclockwise order (with three possible combinations considered identical). To shorten the notation, the vertex membership $a \in t$ and the geometric insidence $q \in t$ have the same notation, when the intended meaning is clear from the context. A vertex p has a set of surrounding triangles $T_p = \{t \in \mathcal{T} | p \in t\} \subset \mathcal{T}$. The edge neighborhood $N_p = \cup_{t \in T_p} t \setminus \{p\}$ is a counterclockwise cyclically ordered set of points connected to p by a triangle edge.

The triangle $t = \{a,b,c\}$ has a unique face normal n_t (oriented outwards) and an area A_t:

$$N_t = \quad (b-a) \times (c-a)$$
$$A_t = \quad \|N_t\|/2 \tag{1}$$
$$n_t = \quad N_t^0, \tag{2}$$

where N_t is a temporary cross product term and vector power $v^0 = v/\|v\|$ of a vector v denotes the vector normalization.

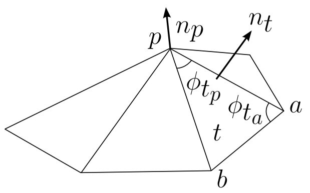


Figure 2: Triangle concepts: tip angles ϕ_{tp} are indexed by vertices p of triangles t. Also the vertex normal n_p and face normal n_t depicted.

The local curvature state of the surface is completely defined after finding out both mean curvature H and the Gaussian curvature G. Sections 2.1- 2.7 present the curvature quantities both in a triangle t and at a vertex p.

2.1 Triangular Gaussian Curvature

The tangential orientation change $\Delta\alpha$ over a length l defines κ, the average of the curvature of a 2D curve over the same length: $\kappa = \Delta\alpha/l$. Analogous to this, the average of the Gaussian curvature of a smooth surface $S \subset \mathbb{R}^3$ can be defined (Pressley, 2010, p.166-168) as the ratio $G_S = \omega_S/A_S$, where A_S is the surface area of S and ω_S is the solid angle of which the surface normal $n(q), q \in S$ traces. This definition can be applied to a triangle $t = \{a,b,c\}$ with vertex normals n_a, n_b, n_c with the exception that the accurate surface S is not known and the triangle area A_t is a lower bound approximation of the hypothetical smooth area $meas(S_q)$. Ramifications of this fact will be addressed in Sec. 3.1.

The solid angle ω_t in Eq. 3 is the total trace of normal $n(q)$, $q \in t$ and, assuming a barycentric interpolation scheme, it equals the solid angle of a vector tri-blade n_a, n_b, n_c (van Oosterom and Strackee, 1983):

$$\tan(\omega_t/2) = \frac{n_a \cdot n_b \times n_c}{1 + n_a \cdot n_b + n_b \cdot n_c + n_c \cdot n_a} \qquad (3)$$

$$G_t = \omega_t/A_t. \qquad (4)$$

The numerator in Eq. 3 equals zero when at least two vertex normals are parallel, which results in requirements 1 and 2 of Sec. 1 to be fulfilled as far as triangular Gaussian G_t of Eq. 4 is concerned. This will be elaborated further in Sec. 3.1.

2.2 Triangular Mean Curvature

Considering a triangle $t = (a,b,c)$ and the associated surface normal approximants n_a, n_b, n_c at vertices a, b, c, and a barycentric dependency of normals $n(q), q \in t$ in the triangle t, one can define a normal mapped parallel triangle $t^u = \{q + un(q) | q \in t\}$. Using a definition of (an averaged) mean curvature in (Pressley, 2010, p. 207), one gets:

$$\begin{aligned} H_t &= \frac{1}{2A_t}\left(\frac{d}{du} A_{t^u}|_{u=0}\right) \\ &= \frac{(n_b - n_a) \times (c-a) + (b-a) \times (n_c - n_a)}{4A_t} \cdot n_t. \end{aligned} \qquad (5)$$

Note that triangular mean curvature $H_t \equiv 0$ when all the vertex unit normals are parallel i.e. $n_a = n_b = n_c$. This leads to requirements 1 and 2 of Sec. 1 to be fulfilled.

2.3 Projective Tip Angles as Weights

The vector angle function acos() and the projected vector angle function $\text{acos}_n()$ simplify the upcoming presentation. The projection angle ϕ'_{12} is the angle ϕ'_{12} between vectors v_1 and v_2 when seen from direction n. See the right part of the Fig. 3. A projection matrix $P(n) = I - n^0 n^{0T}$ is used to define $\text{acos}_n(.)$:

$$\text{acos}(v_1, v_2) = \cos^{-1}(v_1^0 \cdot v_2^0) \qquad (6)$$

$$v'_i = P(n)v_i,\ i = 1,2$$

$$\text{acos}_n(v_1, v_2) = \text{acos}(v'_1, v'_2). \qquad (7)$$

Note that $\text{acos}(v_1, v_2) \equiv \text{acos}_{v_1 \times v_2}(v_1, v_2)$.

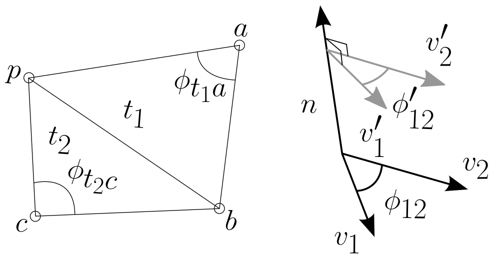


Figure 3: Left: the angle ϕ_{12} between two vectors v_1, v_2 and the projected angle ϕ'_{12} between projected vectors v'_1, v'_2. Right: Definition of the edge angles $\phi_{t'q}$ and $\phi_{t'q'}$ of an edge (p,q).

The projective tip angles ϕ'_{tp} are used systematically to average all triangular quantities $X_t, t \in \mathcal{T}$ to corresponding vertex quantities X_p, $p \in \mathcal{P}$:

$$\phi'_{t_1 a} = \text{acos}_{n_p}(p-a, b-a) \text{(See Fig. 3)} \qquad (8)$$

$$\phi'_p = \textstyle\sum_{t \in T_p} \phi'_{tp} \qquad (9)$$

$$X_p = \textstyle\sum_{t \in T_p} \phi'_{tp} X_t / \phi'_p. \qquad (10)$$

This weighting procedure of a quantity X will be denoted as: $X_t \to X_p$ in the rest of the text.

Good numerical properties of tip angle weighting pointed us to amortize the computational costs by applying it to produce the following vertex properties: normals n_p, triangular mean curvature H_p, triangular Gaussian G_p and principal curvature direction v_p. Another benefit was the unified handling of the boundary points, since the angle sums $\phi'_p \leq \phi_p \leq 2\pi$ give an excellent weighting at the boundary. This is important because e.g. the natural resource data is prune to have missing values and holes in the point cloud, and the boundary points are thus common. When a point p is not in the border, the sum of projected angles equals: $\phi'_p \equiv 2\pi$. There are other weighting schemes in the literature, these are being discussed in Sections 2.5 and 2.6.

2.4 Vertex Gaussian

Since the projected tip angles have been introduced, it is possible to define an alternative vertex Gaussian using the spherical excess (Crane et al., 2013) formulation. The vertex Gaussian of Eq. 12 serves as a reference method:

$$\phi_p = \sum_{t\in T_p} \phi_{tp} \quad (11)$$

$$G_p = (\phi'_p - \phi_p)/(A_p/3). \quad (12)$$

2.5 Vertex Normals

Vertex normals n_p are weighted from triangle normals n_t using the generic scheme of projected tip weighting defined in Eq. 10: $n_t \to n_p$. There are several other possible definitions. Vertex normals can be considered pointing towards the new altered vertices after the local surface is varied, or they somehow represent a continuous but unknown reference positions. The alternatives satisfying DDG convergence requirements listed in (Crane et al., 2013) are reproduced here for discussion. The vertex normal can be:

1. The vector area: $n_p = \left(\sum_{t\in T_p} A_t n_t\right)^0$
2. The area (or volume) gradient $n_p = (dA_p/dp)^0$, when one vertex p is varied in $\mathbb{R}^3$.
3. The normal of a sphere which inscribes vertex p and its edge-neighborhood points N_p. See (Max, 1999; Crane et al., 2013).

Such a sphere fitting required by the alternative 3 is impossible with usual point clouds, but just applying the definition from a case of a perfect sphere fit to any general triangle neighborhood, the resulting normal vector n_p behaves smoothly:

$$n_p = \left(\sum_{t=(a,p,b)\in T_p} \frac{n_t}{\|b-p\|\|a-p\|}\right)^0.$$

According to (Jin et al., 2005), versions 1 and 2 are simple but prune to noise, projected tip angle weighting (our choice) is reliable and simple, and version 3 is rather good but also somewhat expensive.

2.6 Other Mean Curvature Definitions

This short survey omits all methods based on a local fit of a smooth interpolant, see e.g. (Yang and Qian, 2007). These methods show resilience to noise, but tend to have an uncontrollable loss of high frequencies and are usually computationally more expensive than the methods presented in the following.

The mean curvature through the discrete Laplace-Beltrami (also known as the cotan-Laplace) operator has been documented in (Mesmoudi et al., 2012). It is one of the best methods according to (Mesmoudi et al., 2012). The mean curvature H_p at a vertex p becomes:

$$H_p = \frac{1}{4A'_p}\|\sum_{b\in N_p}(\frac{1}{\tan\phi_{t_1a}} + \frac{1}{\tan\phi_{t_2c}})(b-p)\|, \quad (13)$$

where triangles $t_1, t_2 \in T_p$ have a common edge (p,b) with opposite vertex angles ϕ_{t_1a}, ϕ_{t_2b}. See Fig. 3. The vertex specific area $A'_p \approx A_P/3$ is the area of so called mixed Voronoi cell. Using A'_p instead of $A_p/3$ reduces the area contribution of possible obstuse angles ϕ_{tp} in a way, which is detailed in (Mesmoudi et al., 2012). The exact value of A'_p depends on the geometry of the triangle set T_p but is always rather close to the above given expected average. The variance in A'_p adds numerical stability of the estimates of the vertex mean curvature H_p but is rather costly to calculate.

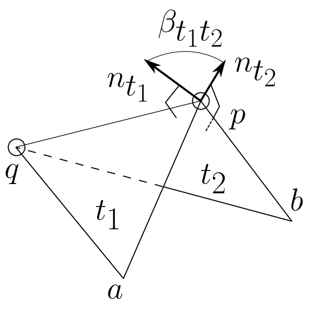


Figure 4: The edge angle $\beta_{t_1t_2}$ is positive when the edge folds downwards (or inwards in case of tumors).

The concentrated Gaussian curvature by (Mesmoudi et al., 2012) equals Eq. 12. The concentrated mean curvature by (Mesmoudi et al., 2012) is re-cast to the notation in this paper as:

$$sgn(t_1,t_2) = -sgn((b-p)\cdot n_{t_1}) \quad (14)$$

$$\beta_{t_1t_2} = \mathrm{acos}(n_{t_1}, n_{t_2}) sgn(t_1,t_2) \quad (15)$$

$$\omega_p = 2\pi - \sum_{t_1\in T_p} \beta_{t_1t_2} \quad (16)$$

$$H_p = \frac{1}{4A'_p}(2\pi - \omega_p), \quad (17)$$

where the edge angles $\beta_{t_1t_2}$ are depicted in Fig. 4, the angle sum ω_p is the inwards opening solid angle at vertex p, and the summation is done over edges $(p,q), q \in N_p$.

The sign of the edge angle $\beta_{t_1t_2}$ is determined by a vector blade handedness sign (a determinant sign) of an edge $(p,q) = t_1 \cap t_2$ between triangles $t_1 = (a,p,q)$ and $t_2 = (q,p,b)$, see Eq. 14. Note that the edge sign is positive for pikes (the situation depicted in Fig. 4) and symmetric: $sgn(t_1,t_2) = sgn(t_2,t_1)$. The normals n_t are a result of earlier stages of the computational process.

Also the solid angle ω_p in Eq. 17 is already available from the preceding solid angle filtering (SAF), which can be done to reduce the noise level of the point cloud or for filtering out the foliage signal (Nevalainen et al., 2016), or before any shape classification via curvature spectrum. Availability of spatial angles ω_p makes this method computationally the cheapest one.

In some applications like tumor detection in electron magnetic resonance (EMR) imaging, the orientation of the surface normal n_p is completely free (but outwards from the tumor). That is why the edge signum refers only to two adjoined triangles t_1 and t_2 which are both oriented outwards. The signum in Eq. 15 requires one vector operation (saxpy, see (Golub and Van Loan, 1996)) of $\mathbb{R}^3$ vectors.

Barycentric interpolation (Theisel et al., 2004) is based on normalized linear change of the normal n over the triangle t from where a generic expression for Gaussian and mean curvature can be deduced. For our purposes only the mean curvature H_{tp} of triangle $t = (a,b,c)$ at a vertex point $p \in \{a,b,c\}$ needs to be considered. The Eq. 18 is adapted to our notation from (Theisel et al., 2004; Nevalainen et al., 2015):

$$\begin{aligned} h &= n_a \times (c-b) + n_b \times (a-c) + n_c \times (b-a) \\ H_{tp} &= (n_p \cdot h)/(2n_p \cdot N_t) \qquad (18) \\ H_{tp} &\to H_p, \qquad (19) \end{aligned}$$

where h is a temporary vector multiplicant.

There is also a triangular approximation of the second fundamental form (Crane et al., 2013; Rusinkiewicz, 2004), which is used in (Rusinkiewicz, 2004) to derive the principal curvatures, mean and Gaussian curvature principal and directions directly. This method requires iteration of a least squares problem, and it seems to be computationally more expensive than the methods covered here.

There are other possible interpolation schemes over a triangle, e.g. using radial basis or by applying the well-known Rodriguez rotation formula (Dorst et al., 2007) twice (first over one edge, then between edge and a vertex of interest). Preliminary tests indicate that these options seem to lead to more complex formulas yet the numerical results stay very close to the schemes included to this study. This holds to both the triangular and the vertex values.

2.7 Principal Curvature Orientation

The curvature eigenvalues κ_{t1} and κ_{t2} of a triangle t are the curvature extremals when tracing a continuous surface S through point p by a perpendicular plane:

$$\kappa_{tl} = H_t \pm \sqrt{H_t^2 - G_t}, l = 1,2 \qquad (20)$$

Object and shape recognition may use any subset of the four curvature characteristics G, H, κ_1, κ_2.

The barycentric surface normal map $t \to t^u$ was used to derive Eq. 5. By applying it again, but this time to find a trajectory with most drastic curvature effect per traversed arc length on a triangle t, one gets the principal curvature direction v_t of a triangle t. This is a direction with the largest curvature (eigenvector of κ_1). The second eigenvector is not of interest, since it will be dictated by the first eigenvector. Another

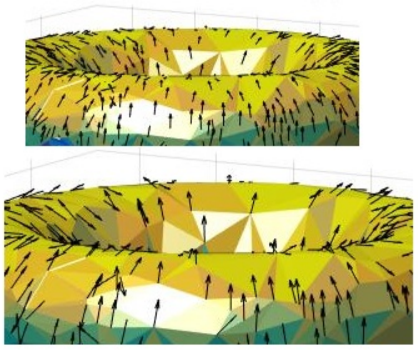


Figure 5: Averaging principal curvature direction v_t from triangles (above) to vertices v_p (below).

way to express v_t is based on constraining the second fundamental form to be diagonal and solving the principal direction from this constraint at Eq. 21. This is different from (Rusinkiewicz, 2004), where whole the second fundamental form is solved by least squares fitting a set of linear constraints. Below are the equations leading to the eigenvalue problem:

$$\begin{aligned} {}'m_a &= P(b-c)(a-c) \text{ (before scaling)} \\ {}'m_b &= P(a-c)(b-c) \text{ (before scaling)} \\ m_a &= \frac{{}'m_a}{{}'m_a \cdot (a-c)} \\ m_b &= \frac{{}'m_b}{{}'m_b \cdot (b-c)} \\ D_t &= (n_a - n_c)m_a^T + (n_b - n_c)m_b^T \\ P(n_t)D_t v_t &= \lambda v_t, \qquad (21) \end{aligned}$$

where ${}'m_a, {}'m_b, m_a' m_b$ are constituents of a constant matrix $D_t = dn_t/dq$, the rate of change of the normal at triangle t. Note that eigenvalue λ is not proportional to principal curvature, since the barymetric mapping does not preserve the unity of the normals.

The weighted summation scheme $v_t \to v_p$ of Eq. 10 is not directly applicable, since the principal directions $\pm v_t$ are defined by the orientation only, without a coherent sign. The summation must take this into account. The following heuristics relies on the monotonic nature of the vector summation of the non-unit cumulative vector $\hat{v}_p$:

$$\begin{aligned} \hat{v}_p(S) &= P(n_p)\textstyle\sum_{t \in T_p} \phi'_{tp} sgn_{tp} v_t \\ S^* &= \operatorname{argmax}_S \|\hat{v}_p(S)\| \\ v_p &= (\hat{v}_p(S^*))^0, \qquad (22) \end{aligned}$$

where S is the set of signums $S = \{sgn_{tp}\}_{t \in T_p}$, which can be found performing $O(|T_p|)$ scalar products

$\hat{v}_{p\,current} \cdot v_t$ by a single enumeration and reversing a subset of signums if necessary.

The weighting scheme in Eq. 22 relies on the projected tip angle weights ϕ'_{tp}, which have multiple applications and thus can be amortized from computational cost. The weighting scheme in (Rusinkiewicz, 2004) uses triangular contributions of the vertex specific area A'_p. This weighting scheme has not been tested by us. Overall, avoiding the least squares fit and area weighting makes our method less expensive computationally.

3 NUMERICAL EXPERIMENTS

Two synthetical models and a visual inspection of a practical problem have been covered, see Sections 3.1- 3.3. The following mean curvature methods have been compared:

1. triangular average mean curvature (our method)
2. L-B (Meyer et al., 2003)
3. concentrated mean curvature (Mesmoudi et al., 2012)
4. barycentric interpolation of the normal (Theisel et al., 2004)

Two Gaussian curvatures have been compared, triangular average Gaussian (our method, Eq. 4), and vertex Gaussian (Crane et al., 2013). Since there are 3 vertex normal definitions, 2 weighted summation policies, 4 mean curvature and 2 Gaussian curvature definitions, results of only the most interesting combinations have been provided.

3.1 A Local Pike

This model demonstrates the different character of each methods with respect to noise in the surface normal direction. Especially the two noise modes presented in Fig. 1 are modelled. The case 1 in Fig. 6 is an apex of a larger regular formation. The case 2 is a single pike which can be either noise or a useful feature. The case 3 demonstrates a large noise field at the highest possible frequency dictated by point cloud density. The geometrical mean $\sqrt{|G|} = \kappa_G$ derived from the Gaussian curvature G is used for the comparisons, since it has the same physical dimension (inverse of radius) as the mean curvature.

The abscissa value h is the height of point p. When $h \to 0$, it is the planar special case with nominal radius $r \to \infty$ and $\kappa r \to 1$.

The barycentric method exaggerates curvature at large h values, which are likely to be noise. The barycentric mean curvature and the triangular curvatures

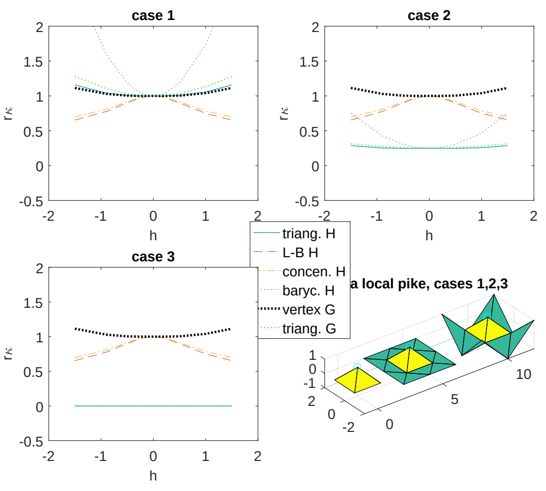


Figure 6: Four mean curvature methods and two Gaussian curvature methods compared in various settings with one point protruding out. The square root of the Gaussian curvature is used for comparison. The analysis point at height h has been circled.

(our methods, both G and H) tend to dampen a singular pike (case 2). The barycentric method is losing its dampening tendency at high values of h, which are more likely to be noise.

The output value of the vertex mean and Gaussian curvatures (and barycentric mean curvature) is scaled downwards (dampened) by a ratio w, value of which depends on the case. The cases $1, 2, 3$ have dampening ratios $w = 1,\ 1/3,\ 0$. A singular pike (case 2) has dampening factor $1/3$ which is still adequate to contribute in the curvature spectrum or to be detected by later pattern recognition phase.

The egg cell pattern of case 3 gets completely dampened by triangular curvatures G and H, and by the barycentric method. The vertex normals defined by Eqs. 2 and 10 become parallel, which then causes the triangular curvatures of the involved triangles t to be zero, see Eqs. 5 and 4. This can be a useful property in some applications, e.g. in stone detection (Nevalainen et al., 2016), or in reducing the granularity effect produced by voxels.

The concentrated curvature, vertex Gaussian, and Laplace-Beltrami are closely related in all cases. The behaviour of all six methods is rather similar to each other in the hyperbolical case (a saddle point) and this case has not been included in this presentation.

3.2 A Torus

A torus of radii $r = 1, R = 2.5$ has been used. This is a classical test case, since the curvature aspects of the ideal shape are analytical, yet both elliptic and hyperbolic local surface metrics occurs.

Two torii, a dense one with $|\mathcal{P}| = 820$ and a sparse one with $|\mathcal{P}| = 220$ were used. Fig. 7 shows the sparse torus with uniform local height distribution. A uniform distribution is used also on the tangential manifold metrics. The height noise concerns point locations and the tangential noise concerns triangulation irregularity. The height noise std. was varied between $0 \leq \sigma_h \leq 0.3r$. The upper end of the noise is typical to many LiDAR applications. The height noise distributions from real LiDAR data are not uniform nor Gaussian. The main curvature spectrum seems to depend mainly on the std of the uniform or natural height distribution, not from the choice between the two.

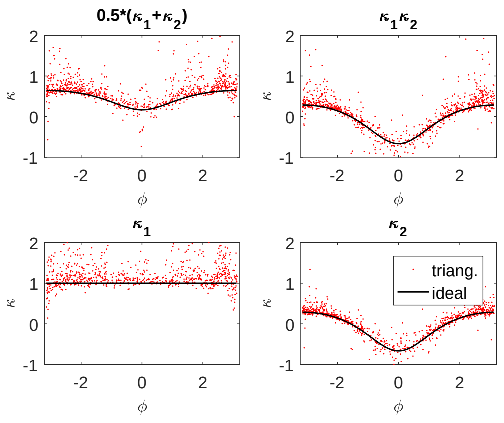


Figure 7: The triangular mean curvature and triangular Gaussian curvature and two curvature eigenvalues on a torus as a function of the angle ϕ associated with the smaller radius. The height noise is at the maximum $\sigma_h/r = 0.3$.

3.3 A Prostate Tumor

The main difficulty with MRI point clouds arises from the anisotropy of the point cloud. The voxels are elongated $2.75 \times 0.48 \times 0.48\ mm^3$ and this demands a lot from the curvature analysis methods. Information about the curvature spectrum of the tumor has been applied to e.g. breast cancer classification (Lee et al., 2015). It is possible that the curvature spectrum will be an important feature alongside spatial texture patterns, 3D Fourier transform, overall size and location of the tumor for clustering algorithms. The Gaussian curvature and principal curvature direction can help in e.g. descriptor based vectorization (Vranic and Saupe, 2001). Fig. 10 depicts a prostate lesion, which shows a typical developable shape: the lesion could be spread back to planar (its Gaussian curvature is approximately zero).

3.4 Results

Our method, when referred, means triangular mean and Gaussian (Eqs. 5, 4, 10) and the principal curvatures derived from them. Tests reach the high noise amplitude range $\sigma_h/r = 0.3$ typical to the natural resource data, see Fig. 7. Effects of noise filtering of L-B and our method have been depicted in Fig. 8. L-B is bound by its fidelity to local geometry. Difference at smooth surface (the left part of the abscissa) is due to the irregularity of the triangles, which brings some advantage to an averaging method like ours.

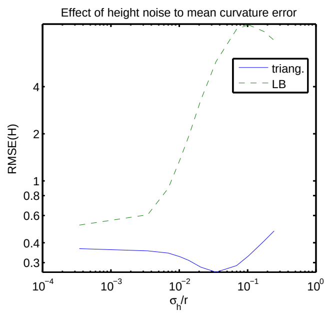


Figure 8: The root mean square error of the mean curvature H estimation error under different perpendicular noise levels σ_h (std.) on a torus with radii r and R.

Fig. 9 has curvature spectrae based on L-B and our method. Other methods were inferior at the noisy end and had to be excluded. The presence of noise spreads the detected spectrum from the ideal smooth case. L-B manages the task only if the triangularization is rather regular and the height noise almost zero. Our method captures two-thirds of the mean curvature distribution, yet suffers from the spectrum spread caused by the noise, which is inevitable. Both Gaussian approximations perform as well enabling e.g. the curvature spectrum classification to be possible under wide range of noise levels. Other two methods (concentrated and barycentric) perform worse than L-B.

Fig. 10 depicts the prostate lesion with the Gaussian curvature close to zero everywhere meaning its surface is mostly developable (a so called ruler surface). This is an artifact caused by a combination of the elongated voxels and the method used for triangularization. The surface has high energy noise component caused by the voxel granularity. Our methods dampen this highest geometric noise component automatically. Also concentrated and barycentric mean curvatures performs surprisingly well. L-B suffers from its fidelity to the highest shape frequency component.

The computational cost of the barycentric method is too high when compared to its performance, see

Table 1: Evaluation of the mean curvature methods. Proposed methods in boldface.

Method	Vector opers/t	Spectrum quality	Singular noise w	Massive noise w
vertex G	15	good	1	1
triangular G	15	good	1/3	0
triangular H	15	good	1/3	0
LB	18	average	1	1
concentrated	2	poor	1	1
barycentric	32	poor	1/3	0

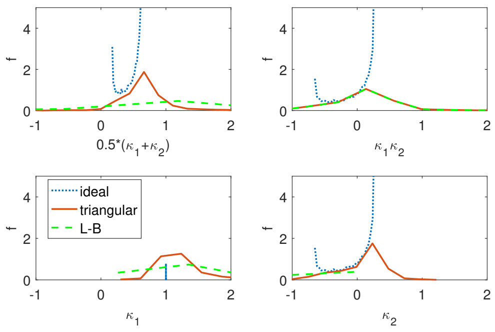


Figure 9: The ideal curvature spectra of a torus with $r = 1$, and Laplace-Beltrami and triangular approximations. The effect of height noise $\sigma_h/r = 0.1$ spreads out the approximated spectrae.

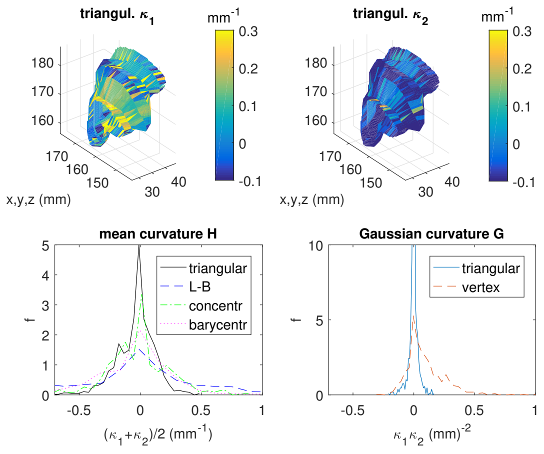


Figure 10: Upper row: the principal curvature components. Lower row: Distributions of the mean and Gaussian curvatures by different methods.

Table 1. L-B has the best accuracy when the perpendicular noise is small and the triangulation is rather regular, but fails when the perpendicular noise is high. The spectrum quality is given a qualitative judgement. See the definition of the dampening ratio w at Sec. 3.1.

4 CONCLUSIONS

The proposed method (triangular mean and Gaussian curvature) has about the same computational demand as the reference method (LB mean and vertex Gaussian curvature) in case where the vertex specific area A'_p of the mixed Voronoi cell is computed exactly as recommended in (Mesmoudi et al., 2012). Based on the good performance under height noise, it seems that the triangular method should be used in such natural resource data applications, where the curvature spectrum is required, and the spectral range should reach near the highest shape frequency, but excluding the large excitations of the mentioned frequency.

The above definition may seem contrived, but e.g. a typical rasterization process is lossy and tuning the filtering process requires a lot of parameters, which concern the highest shape frequency naturally contained with the methods proposed here. Further validation is necessary with e.g. track analysis of forestry harvesters (Pierzchala et al., 2016).

Principal orientation computation presented in Sec. 2.7 is closely related to other two methods presented, e.g. it uses the same projected tip angle weights. One has to inspect in the future how useful the principal orientations are in micro-topographic analysis. It may be that a multi-scale approach for producing several TIN models with coherent curvature and principal orientation information is needed.

There is a huge bulk of raster analysis methods and a lot of experience in applying these methods for e.g. height raster data analysis. Emerging triangular analysis tools based on DDG will not outdate these methods, but in some cases there seems to be potential to improve the curvature spectrum range closer to the theoretical limit dictated by the point cloud sample density and the known sample accuracy.

ACKNOWLEDGEMENTS

This work was supported by the funding from the Academy of Finland (Grant 295336). Kevin R. Vixie

and Otis Chodosh brought up the intuitive view to mean curvature on Math Overflow site on 2012.

REFERENCES

Crane, K., de Goes, F., Desbrun, M., and Schröder, P. (2013). Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, pages 7:1–7:126, New York, NY, USA. ACM.

Dorst, L., Fontijne, D., and Mann, S. (2007). *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA.

Jin, S., Lewis, R., and West, D. (2005). A comparison of algorithms for vertex normal computation. *The Visual Computer*, 21:71–82.

Lee, J., Nishikawa, R. M., Reiser, I., Boone, J. M., and Lindfors, K. K. (2015). Local curvature analysis for classifying breast tumors: Preliminary analysis in dedicated breast ct. *Medical Physics*, 42(9).

Max, N. (1999). Weights for computing vertex normals from facet normals. *Journal of Graphics Tools*, 4(2).

Mesmoudi, M. M., De Floriani, L., and Magillo, P. (2012). Discrete curvature estimation methods for triangulated surfaces. In *Applications of Discrete Geometry and Mathematical Morphology*, pages 28–42. Springer.

Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H. (2003). *Visualization and Mathematics III*, chapter Discrete Differential-Geometry Operators for Triangulated 2-Manifolds, pages 35–57. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mitra, N. J. and Nguyen, A. (2003). Estimating surface normals in noisy point cloud data. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG03, pages 322–328, New York, NY, USA. ACM.

Nevalainen, P., Middleton, M., Kaate, I., Pahikkala, T., Sutinen, R., and Heikkonen, J. (2015). Detecting stony areas based on ground surface curvature distribution. In *2015 International Conference on Image Processing Theory, Tools and Applications, IPTA 2015, Orleans, France, November 10-13, 2015*, pages 581–587.

Nevalainen, P., Middleton, M., Sutinen, R., Heikkonen, J., and Pahikkala, T. (2016). Detecting terrain stoniness from airborne laser scanning data . *Remote Sensing*, 8(9):720.

Pierzchala, M., Talbot, B., and Astrup, R. (2016). Measuring wheel ruts with close-range photogrammetry. *Forestry*, 89(4):383–391.

Pressley, A. (2010). *Elementary Differential Geometry*. Springer Undergraduate Mathematics Series. Springer London.

Rusinkiewicz, S. (2004). Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission*.

Schaer, P., Skaloud, J., Landtwing, S., and Legat, K. (2007). Accuracy Estimation for Laser Point Cloud Including Scanning Geometry. In *Mobile Mapping Symposium 2007, Padova*.

Theisel, H., Rössl, C., Zayer, R., and Seidel, H. P. (2004). Normal based estimation of the curvature tensor for triangular meshes. In *In PG04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG2004)*, pages 288–297. IEEE Computer Society.

van Oosterom, A. and Strackee, J. (1983). A solid angle of a plane triangle. *IEEE Trans. Biomed. Eng.*, 30(2):125–126.

Vranic, D. V. and Saupe, D. (2001). 3d shape descriptor based on 3d fourier transform. In Fazekas, K., editor, *3D Shape Descriptor Based on 3D Fourier Transform In Proceedings of the EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services (ECMCS 2001)*, pages 271–274.

Wardetzky, M., Mathur, S., Kaelberer, F., and Grinspun, E. (2007). Discrete laplace operators: No free lunch. In Belyaev, A. and Garland, M., editors, *Geometry Processing*. The Eurographics Association.

Yang, P. and Qian, X. (2007). Direct computing of surface curvatures for point-set surfaces. In *SPBG'07*, pages 29–36.

Publication P6

Estimating the rut depth by UAV photogrammetry

P. Nevalainen, A. Salmivaara, J. Ala-Ilomäki, S. Launiainen, J. Hiedanpää, L. Finér, T. Pahikkala, J. Heikkonen. Estimating the rut depth by UAV photogrammetry. Remote Sensing. 2017, 9(12), 1279

remote sensing

MDPI

Article

Estimating the Rut Depth by UAV Photogrammetry

Paavo Nevalainen [1,*], Aura Salmivaara [2], Jari Ala-Ilomäki [2], Samuli Launiainen [2], Juuso Hiedanpää [3], Leena Finér [2], Tapio Pahikkala [1] and Jukka Heikkonen [1]

1 Department of Information Technology, University of Turku, FI-20014 Turku, Finland; tapio.pahikkala@utu.fi (T.P.); jukhei@utu.fi (J.H.)

2 Natural Resources Institute Finland (Luke), Latokartanonkaari 9, FI-00790 Helsinki, Finland; aura.salmivaara@luke.fi (A.S.); jari.ala-ilomaki@luke.fi (J.A.-I.); samuli.launiainen@luke.fi (S.L.); leena.finer@luke.fi (L.F.)

3 Metsälinkki Ltd., Kylämetsäntie 2 B, FI-28760 Pori, Finland; juuso.hiedanpaa@metsalinkki.fi

* Correspondence: ptneva@utu.fi; Tel.: +358-40-3518236

Received: 24 September 2017; Accepted: 6 December 2017; Published: 9 December 2017

Abstract: The rut formation during forest operations is an undesirable phenomenon. A methodology is being proposed to measure the rut depth distribution of a logging site by photogrammetric point clouds produced by unmanned aerial vehicles (UAV). The methodology includes five processing steps that aim at reducing the noise from the surrounding trees and undergrowth for identifying the trails. A canopy height model is produced to focus the point cloud on the open pathway around the forest machine trail. A triangularized ground model is formed by a point cloud filtering method. The ground model is vectorized using the histogram of directed curvatures (HOC) method to produce an overall ground visualization. Finally, a manual selection of the trails leads to an automated rut depth profile analysis. The bivariate correlation (Pearson's r) between rut depths measured manually and by UAV photogrammetry is $r = 0.67$. The two-class accuracy a of detecting the rut depth exceeding 20 cm is $a = 0.65$. There is potential for enabling automated large-scale evaluation of the forestry areas by using autonomous drones and the process described.

Keywords: micro-topography; forest harvesting; UAV; photogrammetry; micro-topography; point cloud; TIN; curvature; rut formation

1. Introduction

Mechanized wood harvesting operations can cause rut formation, which deteriorates soil quality, decreases forest productivity and affects hydrological balance and water quality through changed sediment discharge [1–8]. Thus, the rut depth distribution is one of the central measures of the environmental and economic impact of harvesting operations.

Indeed, international forest certification standards issued by the Forest Stewardship Council (FSC) and by the Programme for the Endorsement of Forest Certification (PEFC) along with national legislations have recommendations and regulations concerning the rut formation [9,10]. Monitoring the obedience of standards and laws requires data of ruts, and these data are traditionally collected by manual measurements from randomly-selected samples of logging sites [11]. Due to the costs of time-consuming manual measurements, data collected cover unfortunately only a very small part of the operation sites.

Ruts are formed when the loading exerted by the forest vehicle exceeds the strength of the soil [12–14]. The bearing capacity depends on various static and dynamic factors: e.g., soil type, root density, slope and other micro-topographic water dynamics and frost state. As a result, both the spatial and temporal variation in trafficability is extremely large [14,15]. An easily collected and extensive dataset on rut depths that covers field measurements over several test sites and various weather conditions is thus needed for any attempts to model forest terrain trafficability [16].

The development of techniques in the collection, processing and storage of data offers a great possibility to collect extensive datasets on rut formation. Close-range aerial imagery captured by unmanned aerial vehicles (UAV) provides one of several methods under development. This method for retrieving data on changing terrain by photogrammetry with efficient digital workflows has been used by [4,13,17,18].

Other potential methods include light detection and ranging (LiDAR) scanning [16,19] and ultrasonic distance ranging accompanied by proxy measurements such as controller area network (CAN-bus) data [20]. Compared with other methods, UAV photogrammetry provides a cost-effective option for collecting high-resolution 3D point clouds documenting the operation site in full extent [4].

The photogrammetric UAV data have some deficiencies [20] when compared with aerial LiDAR (ALS) [21]. ALS is a true 3D mapping method in the sense that it allows multiple Z returns from a single laser beam, whereas photogrammetric data have only one Z value for each point. This results in hits concentrated at the top layer of the dense canopy or undergrowth, leaving the ground model often rather inadequate or narrow.

Usually, the logging trails have a corresponding canopy opening (a canopy pathway) around them, and the UAV data points sample the trail surface if the flight altitude is low enough and the flight pattern dense enough. Occasional obstructions and discontinuities occur, though, and the numerical methods must adapt to this difficulty. While the point cloud is limited compared to the one produced by a LiDAR scanner, the potential for rut depth detection is worth examining due to lesser costs, smaller storage demands and abundance of details provided by the imagery (RGB) information.

A large dataset on ruts would aid not only extensive monitoring of forest standards, but also the modeling and forecasting purposes as rut formation depends quite directly on soil bearing capacity and, thus, relates to forest terrain trafficability [22]. A post-harvest quality assurance pipeline using the UAV technology could be an essential part in cumulating such dataset.

The rut width depends on the tire dimensions, and the distance between the ruts depends on the forest machine dimensions; these can be approximately known a priori. The rut features have generally a rather fixed scale: a rut is about 0.6–1.2 m wide; the depth varies between 0 and 1.0 m; and the distance between ruts is a machine-specific constant usually known beforehand (approximately 1.8–2.5 m).

The most relevant study in trail detection is [23], where the usage of the digital elevation map (DEM) has been avoided, making the point cloud recording possible without specific geo-referencing markers. This makes the field procedure simpler and faster and gives more freedom to choose the UAV flight pattern. A multi-scale approach similar to [24] could be useful, since the point cloud density changes especially at the canopy border. Now, a mono-scale analysis has been used. The triangularization produced by methods described in Section 2.5 has approximately a 20-cm average triangle edge length.

Several restrictions have been made in [23]: the locations have to be relatively smooth and without underbrush. An interesting study of [4] uses photogrammetric data from a ground-based recording device. A very good matching of rut depth has been achieved using different point cloud processing methods. The test site is open canopy, but measurements require a field team traversing the trail.

We propose a sequence of processing steps for more realistic conditions (varying terrain, some underbrush and trees allowed, trail pathways only partially exposed from the canopy cover). We have subtracted DEM height provided by the National Land Survey of Finland (NLS) for some visualizations, but actual methods are independent of the local DEM.

We propose a procedure for large-scale field measurement of rut depth. The procedure is based on the point cloud collected by the UAV photogrammetry. The following data models are constructed for autonomous logging trail detection and finally for rut depth data extraction:

1. a canopy height model to focus the point cloud on the logging trail. The model is produced by a solid angle filtering (SAF) of point clouds [25];
2. a surface model as a triangularized irregular network (TIN) for trail detection; the model is produced by applying SAF with a different parameter setting, followed by mean curvature flow smoothing (MCF) [26];
3. ground model vectorization (orientation of possible ruts and likelihood of having a trail through any given point;
4. the height raster of a straightened harvesting trail; this phase uses a histogram of curvatures (HOC) method;
5. a collection of 2D rut profile curves.

2. Materials and Methods

2.1. Study Area

The field study was carried out in Vihti, Southern Finland (60°24.48′N, 24°23.23′E), in mid-May 2016. Two routes within approximately 1 km from each other were driven first by an 8-wheeled Ponsse Scorpion harvester with the official operating weight of 22,500 kg, and the rear wheels of each bogie were equipped with chains. Consecutively, the route was driven 2–4 times by a loaded 8-wheeled Ponsse Elk forwarder. The mass of the forwarder was 30,000 kg (operating weight + the scale weighted full load as seen in Figure 1, second detail below). The rear wheels of the front bogie were equipped with chains, and the rear bogie was equipped with Olofsfors Eco Tracks. The tire width was 710 mm for both the harvester and the forwarder. The general setting of the forest operation, the machinery and their chose weights and the routes used closely resemble a standard forest harvesting operation. As mentioned in several publications [4,16,22], the rut depth has high variability dependent on a multitude of factors, and currently, only case studies are practical.

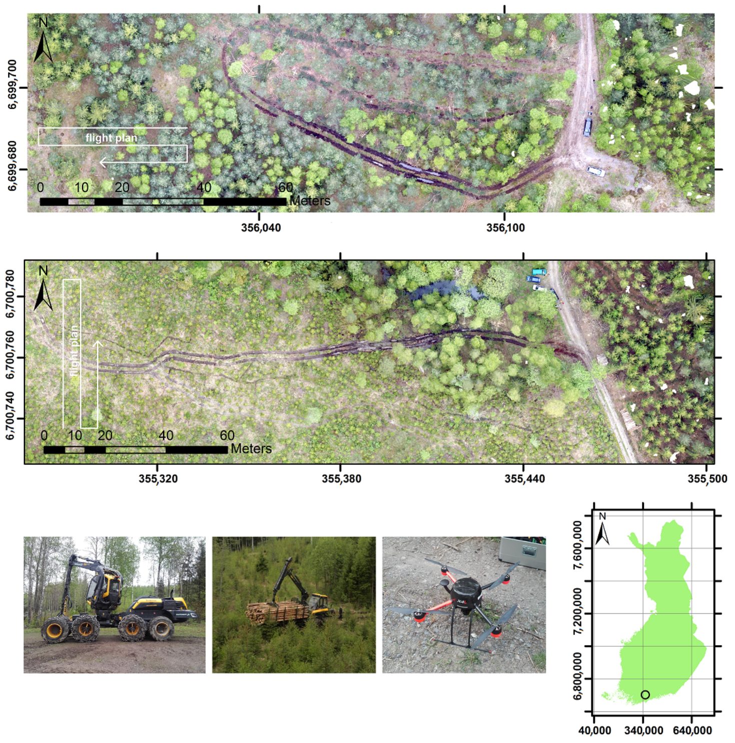


Figure 1. Top and **middle**: Trail 1 and Trail 2 areas as orthophotos and the drone flight plan schematics; **bottom**: the harvester, forwarder and drone used in the field campaign are shown with the map location (with the ETRS-TM35FIN coordinate system, see Abbreviations section) of the site.

The ruts were measured manually using a horizontal hurdle and a measuring rod. Manual measurements were taken at one-meter intervals. Accurate GPS coordinates of manual measurements were recorded at 20-m intervals.

The route covered various soil types: Bedrock covered by a 5–15-cm layer of fine-grained mineral soil, clay and sandy till covered by an organic layer of 10–60 cm. The terrain profile varied from flat to slightly undulating. Soil moisture conditions varied along the route. Part of the trail in Trail 1 was covered with logging residue.

2.2. Point Cloud Data

Photogrammetric data were collected using a single-lens reflex (consumer-grade compact) camera (Sony a6000) with a 24 MPix 20-mm objective. The camera was attached to a GeoDrone-X4L-FI drone, which has a normal flight velocity of 6 m/s and an onboard miniature global navigation satellite + inertial system (GNSS/INS) with a typical accuracy within a one-meter range. The flight heights and the corresponding ground sample distance (GSD) are reported in Table 1. The forward overlap was 80% and the side overlap 70%.

Table 1. Details of the photogrammetric data. GSD is provided by the GeoDrone manual. Vertical noise is an approximation of the vertical accuracy of the point cloud points.

Trail	Flight Height (m)	GSD (cm)	# Photos	Cloud size	Route Length (m)	Vert.Noise (std.cm)
1	100	2.0	42	8.1×10^6	208	4.0
2	150	3.0	34	7.1×10^6	280	3.5

Since there were two different locations, it was possible to test two different flight heights. The optimal flight height depends on the tree species, canopy density and tree density. Trail 2 had less tree cover, so the vertical accuracy was about the same. The aim of this study, however, was not to optimize the flight plan, but to develop an efficient method for point cloud post-processing. The vertical noise level was measured on the smooth surfaces of the geo-markers from the difference of the initial point cloud and the final ground TIN.

Two sets of four and five ground control points (depicted as circles in Figure 2) were positioned to minimize terrain surface height error and to allow reconstruction of the unified point cloud from aerial images. GPS coordinates of the control points were recorded with a setting of a 5-cm std.position error target with the RTK GeoMax Zenith25 PRO Series. The above mentioned error target can be reached almost immediately in open fields, but the forest canopy imposes difficulties; sometimes, a prolonged period of waiting time was required for a reliable measurement. All measurements were accomplished within a period of 30 min of cumulated field time per site, however.

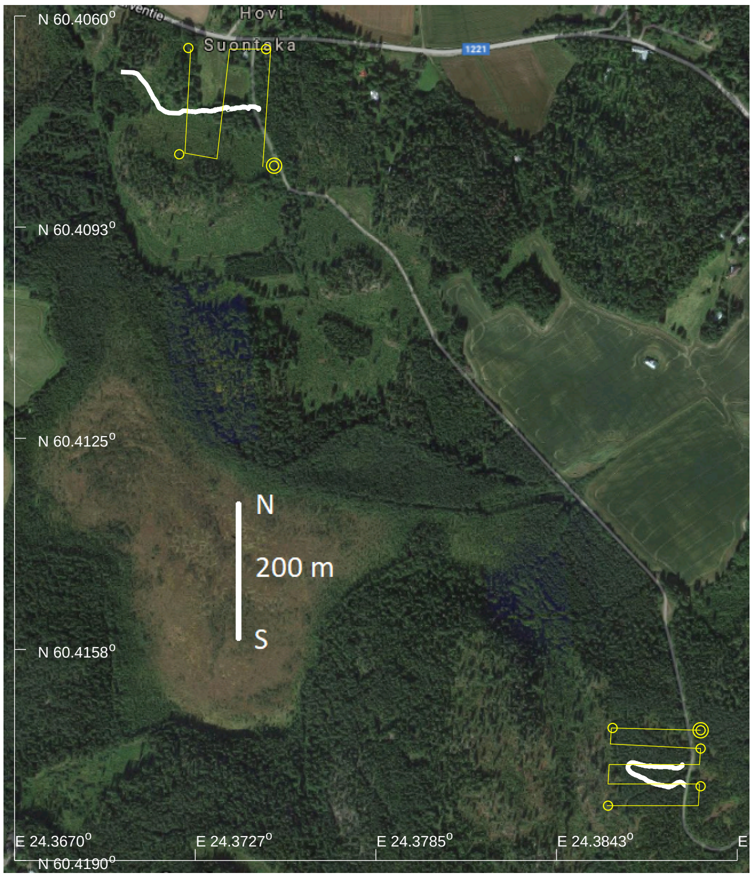


Figure 2. Trail 1 (below) and Trail 2 (above) are separated by approximately 1 km. The geo-reference markers are depicted by circles, the flight path is shown with the yellow cross-line and the trails by thick white curves. Trail 2 was not fully covered by the UAV photogrammetry. The starting points of the flight paths are indicated by the double circle.

2.3. Methodology

An overview of the methodology is given here. The subsequent Sections 2.5–2.8 go through different stages in detail, while introducing various parameters. Section 2.9 summarizes these parameters.

The TIN models for both the canopy height and the ground model are developed first. The canopy height model is used to improve the quality of the ground model at the fragmented boundaries. The ground model visualization is used for comparing with the manually-measured rut depths. A height convolution iteration is performed at this phase to form a smooth center line. The convolution filter uses the information about the assumed wheel distance of the forest vehicle. The final step is about recording the rut depth profiles.

The essential steps of the workflow are: (1) UAV photogrammetry; (2) canopy elimination, forming and visualizing the (4) curvature state of the (3) ground model; (5) finding the central line of the trail and (6) forming the profile model of the ruts; see Figure 3. Many processing steps used have several alternatives, and a rather intense search for various techniques has already been done in the preliminary phase of this study.

Several parameters were used to produce the point cloud, the ground model TIN and the final rut depth profiles. A summary of the parameters can be found in Section 2.9, where all the numerical values are stated.

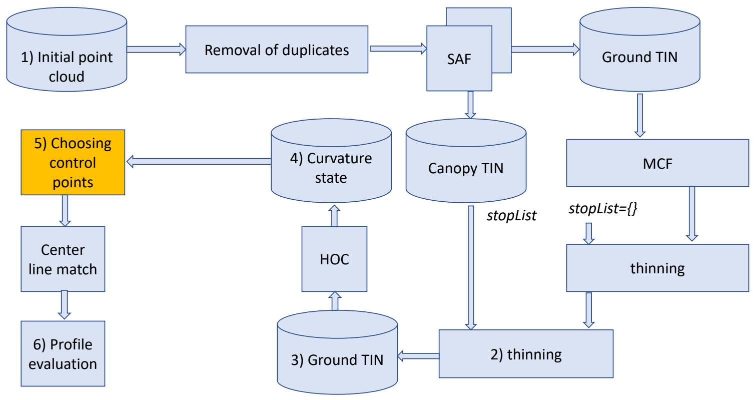


Figure 3. The process chart of the computational steps (1)–(6), which are detailed later in the text. SAF is used twice to produce two TIN models. The thinning is done first to all ground points with no limitation, then to that part of the remaining canopy front points, which are overlapping the canopy points. HOC produces a curvature state, which is visualized in order to choose manual control points. The control points are inserted manually, and this phase has potential for further automation (orange box). The end product is the rut depth profile distribution.

2.4. Step 0: Point Cloud Generation

The photos taken were processed with AgiSoft PhotoScan Professional software into 3D high-resolution point-clouds. The software applies a structure-from-motion (SFM) process that matches feature-based images and estimates from those the camera pose and intrinsic parameters and retrieves full 3D scene reconstruction [27], which is geo-referenced by ground control points [28]. The geo-reference markers were identified manually in the set of images in which they appeared, and the AgiSoft software performed the necessary geo-referencing using the images, the UAV GNSS/ISS records and the geo-marker pixel and positioning information as the input. Point clouds were re-projected to the ETRS-TM35FIN projection.

2.5. Steps 1, 2: Point Cloud Preprocessing

Proper preprocessing proved to be essential for detecting the rut profiles accurately. Eliminating the direct and secondary effects of the proximity of the canopy was important. Canopy narrows the stripe of the exposed ground and causes a curvature signal. The vegetation and the harvesting residue cause noise and force one to filter and smoothen the TIN in a controlled way.

Two point clouds of Figure 2 and Table 1 have initially an average distance of 6 cm to the horizontally nearest natural neighbor; see Figure 4.

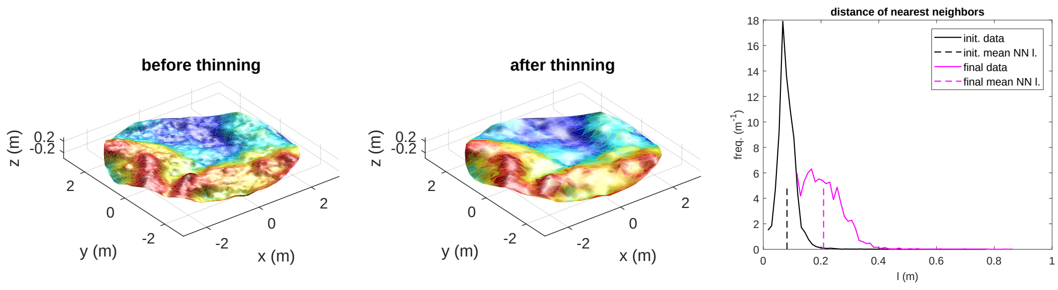


Figure 4. Left: The TIN model after SAF. **Middle**: The TIN model after the thinning process where the average triangle edge length has been forced to 20 cm. **Right**: Initial and final NN distance distributions. The mean point distance $\bar{l}$ has been shifted from initial 0.07 m to 0.20 m. **Bottom right**: Definition of the NN distance as an average of the TIN triangle side lengths.

Natural neighbors (NN) are defined by the well-known 2D Delaunay triangulation [29], and those are needed when the ground TIN model is created. The rut detection requires a detailed control over the TIN model production, so we do not use the standard methods available from the GIS and point cloud software. Instead, we apply SAF [25], which utilizes the solid angle ω_p of the "ground cone" associated with each point p. Filtering is based on having all accepted points within the limits $\omega_1 \leq \omega_p \leq \omega_2$ in the resulting TIN.

An important descriptor of the point cloud before and after the thinning is the distribution of the triangle edge lengths of the Delaunay triangularization [30]. See Figure 4, left, where two distance distributions (before and after thinning) are depicted.

The preprocessing of the point cloud data consists of a sequence of operations to make the rut depth extraction computations feasible. The operations are listed below:

(a) Eliminating point pairs with horizontal distance less than 5 mm. This makes the further processing code simpler. The point above/below gets removed when computing the ground/canopy model, respectively. The minimum distance was decided by the 0.5% data loss criterion. See Figure 4. This process can be done in linear $O(|P|)$ time, where $|P|$ is the size of the point cloud.

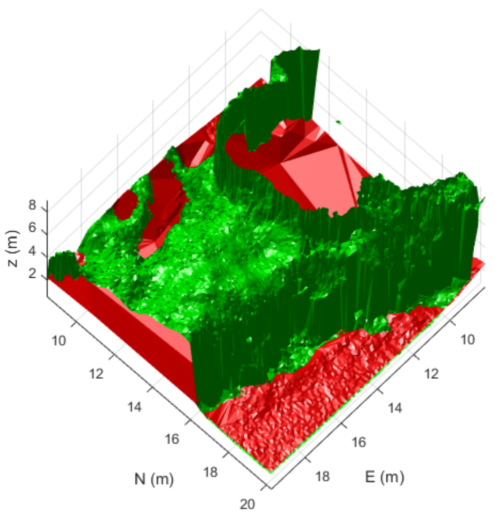
(b) Building the ground TIN model by applying SAF two times in order to have a structured reduction of the canopy noise at the narrow stripes at the further steps. The first SAF run builds (a) the canopy TIN (green in the middle detail of Figure 5). The model is based on controlled triangularization unlike usual canopy models based on local windows; see e.g., [31]. The second run builds (b) the ground TIN. The spatial angle limit parameters used can be found in Section 2.9. Two models built are used in Steps (c) and (d).

(c) The mean curvature flow (MCF) [26] was applied to the ground TIN model to smoothen it. The MCF procedure contains one method parameter $\lambda \in [0, 1]$, which controls the aggressiveness of the smoothing effect. MCF is a TIN smoothing method, which resembles the mean filtering of the raster data. It has a typical trade-off between smoothing the noise and possibly deteriorating a useful signal.

(d) Thinning of the point cloud. Thinning is dictated by the mean natural neighbor distance $\bar{l}$ (see Figure 4), which increases during the process from the preliminary average $\bar{l} = 0.07$ m to the target value $l_b = 0.20$ m in the red zone of Figure 5. The limit value still enables a rather good imprint of the rut shape on the resulting surface triangularization. The green canopy area was subjected to thinning to $l_c = 5.0$ m. This was to eliminate the noise at the border areas of the ground model. Parameters l_b and l_c were deemed the most suitable for the further process. See the listing of the actual Algorithm 1.

Figure 6 shows some large TIN triangles produced by the thinning step (d). The large triangles do not contribute to the curvature analysis, since they are eliminated by a triangle size limit $l_{max} = 1.2$ m. This is to make it possible to analyze the trails with a very fragmented boundary and the canopy front.

There is the intermediate zone called the canopy front (see the blue horizontal bar at the right side of Figure 5), which is being defined by the height difference of two TIN models. Taking the height difference of two TIN models with independent triangularization is a standard GIS procedure with some subtle practical considerations and speed-ups, which have not been documented here. The end result consists of a mixture of red and blue zones with the canopy part (green) practically disappearing; see Figure 5. The sparse areas have very large triangles (see Figure 6), but the height differences at the canopy front are moderate, i.e., the artificial curvature spike visible in the right detail of Figure 5 becomes partially eliminated. The advantage is that the canopy removal can be achieved without referencing any DEM model.



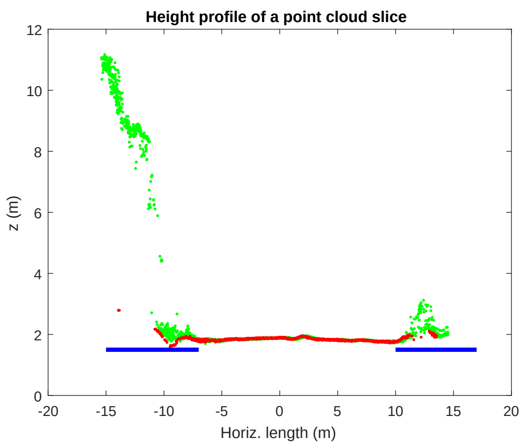
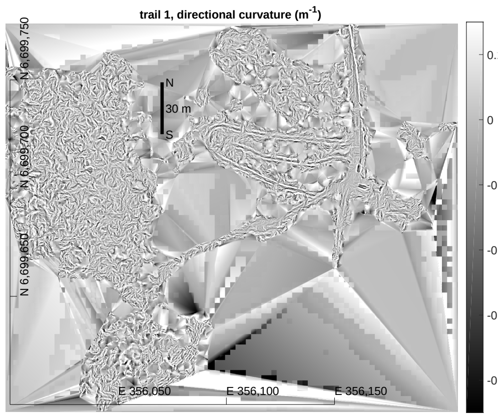


Figure 5. Left: The canopy detection is based on two separate SAF runs with different parameters to detect the canopy top surface (green) and the normal ground model (red). **Right**: A vertical slice of the point cloud depicted at the left. Blue vertical bars indicate where the extra thinning of the ground model is being applied. Note the ground model profile arising at the canopy front due to the lack of photogrammetric ray penetration.



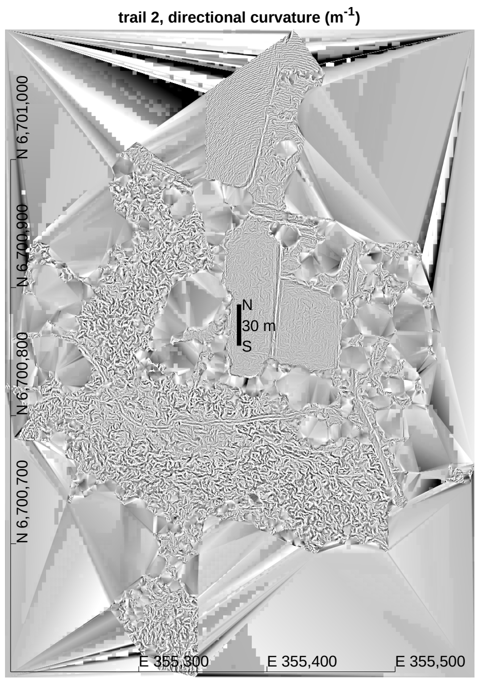


Figure 6. Left: Trail 1; Right: Trail 2. The directional curvature at the locally-dominant direction. The visualization is formed from 20 different directional curvature images by tiling them using 3.8 m image tiles. **Left**: Trail 1 is in the center. **Right**: Trail 2 traverses horizontally across the lower part of the image. Co-ordinates are in ETRS-TM35FIN (m).

Since the thinning is a non-trivial procedure, it is outlined here. The algorithm runs initially as presented, and the *stopList* is a list of points already removed. Initially, it is set to be empty; but when the target length l_b has been reached, a new target value l_c will be set, and a second run ensues with the $stopList$ being initialized by a set of the points at the canopy front (a zone indicated by blue bars in

Figure 4). This makes it possible to thin the canopy front more heavily. The canopy front is the set of connected ground model triangles with any canopy point above each of the triangles:

Algorithm 1: Thinning the point iteratively making the mean neighborhood distance reach the target value. l_{target} is increased incrementally during the process. Note: The algorithm is applied twice: first, as it is presented, and the second time with the *stopList* being initiated with the canopy front.

```
Data: Point cloud $P$, mean distance limit $l_{target}$
Result: Thinned point cloud $P$
$\bar{l} \leftarrow \infty$ while $\bar{l} > l_{target}$ do
    $T \leftarrow Delaunay(P)$
    $edges \leftarrow$ enumerating all edges of $T$
    $ls \leftarrow$ lengths of all $edges$
    sort $ls$ and $edges$ to the ascending length order
    $\bar{l} \leftarrow mean(ls)$
    $stopList \leftarrow \{\}$ (canopy front points on the second run!)
    while $\bar{l} > l_{target}$ and $|stopList| < 0.1|P|$ and $\min_{l \in ls} l < 0.5\, l_{target}$ do
        Starting from the shortest edge $e = (a, b) \in edge$:
        if $\{a, b\} \cap stopList = \{\}$ then
            $P \leftarrow P \backslash \{a\}$ Remove one end point $a$ of $e$ (choose randomly)
            $stopList \leftarrow stopList \cup \{b\}$ Add the other point $b$ to the $stopList$ list
    end
end
```

A Delaunay triangulation package providing a batch input, and a dynamical delete is recommended for the algorithm. If no suitable algorithm is available, we recommend space partitioning and slight modification of the algorithm (available from the authors), which thins the TIN as much as is possible until a new TIN is produced in one batch run.

2.6. Steps 3, 4: Trail Detection

Trail detection has two steps: (a) extracting the rotation-invariant curvature state and visualizing the TIN; and (b) manual insertion of the trail control points to initialize the numerical trail center line match.

Visualizing the trails: The target area was covered by a grid of circular samples with the grid constant $\delta = 3.5$ m and the sample radius $r = 2.1$ m. A dominant direction of the curvature was detected from each of the samples using HOC, which is a closely related to the well-known histogram of gradients (HOG) method [32], except that it works with the curvature values. HOC produces histograms of dominant curvatures and the corresponding orientation. The directed curvature is a basic geometric entity along a vertically-oriented plane cutting the TIN surface. The orientation is chosen so that there is a maximum difference between two perpendicularly-oriented curvature histograms; see Figure 7. This arrangement has a rotational invariance.

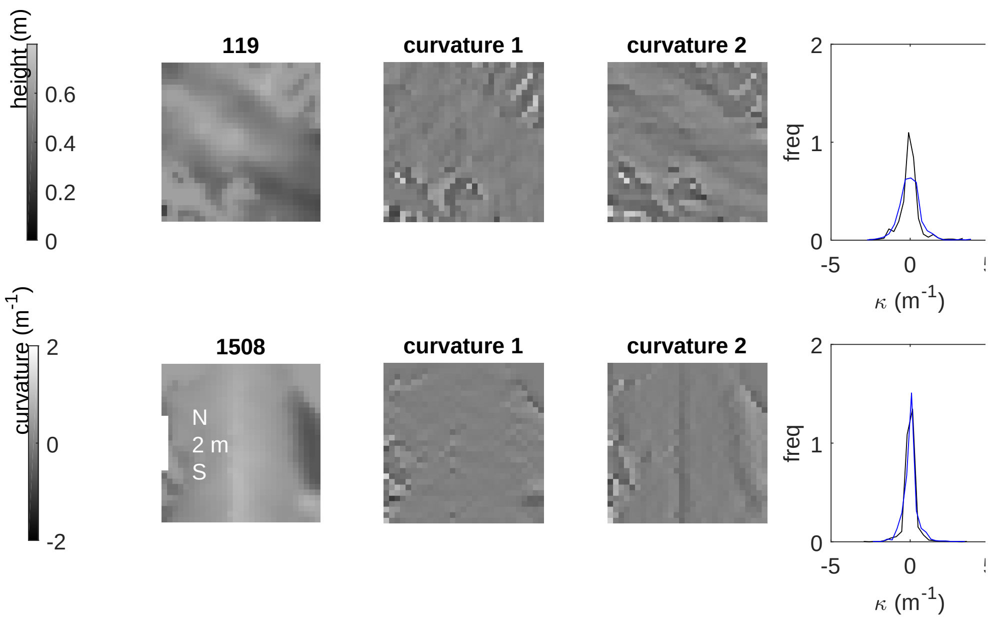


Figure 7. **Top**, from left to right: The height map of a 5.3-m square spot with two parallel ruts, the direction with the smoothest curvature, the direction with the most drastic curvature distribution and the corresponding histograms f_1 and f_2. **Bottom**: A spot with a ditch beside a road having a sharp V shape. beside a road having a sharp V shape. The curvature is rather isotropic. Actual samples are circular and centered on squares depicted with a radius $r = 2.1$ m.

The directional curvature can be specified for a TIN in a discrete differential geometric (DDG) way. The definition is based on the triangular average of the mean curvature introduced in [33], with the following modification: vertex normals of each triangle are projected to the directional plane before application of the mean curvature equation. The constructed histogram is then weighted by triangle surface areas. The histogram concerns a set of triangles among a sample circle with a radius r with circle centers forming a sample grid with a grid constant δ. The end result seen in Figure 6 was strongly affected by the choice of the sample radius r.

The details of producing the sample histogram $f(\kappa_\alpha)$ of the directional curvature κ_α in the orientation α are included in Appendix A. The directional curvatures were produced in $n_\alpha = 20$ directions. The number of directions was chosen by the exhibited noise level (difference in neighboring directions), which is larger the less directions are chosen and the smaller the samples are. $0 \leq \alpha_i \leq \pi, i = 1, ..., n_\alpha$.

Formally, the curvature state has three components, the orientation α^*, the smooth directional curvature histogram $f_1(\kappa) = f(\kappa_{\alpha^*})$ at the previously-mentioned orientation and the rough directional curvature histogram $f_2(\kappa) = f(\kappa_{\alpha^*+\pi/2})$, the last two perpendicular to each other. Histograms f_1 and f_2 are scaled to be distributions observed from Equation (A4). The orientation α^* is chosen by maximizing the directional distribution difference $e(\alpha)$:

$$\alpha^* = \arg\max_\alpha e(\alpha), \tag{1}$$

where $e(\alpha) = \frac{1}{2}\int |f_1(\kappa) - f_2(\kappa)| d\kappa$. Finally, one can define the curvature eccentricity $0 \leq e \leq 1$, extremes associated with isotropic and maximally anisotropic cases, respectively:

$$e = e(\alpha^*) \tag{2}$$

The value of the upper limit follows from the fact that both f_1 and f_2 are distributions. A vectorized representation $\mathbf{x}$ of the curvature state is formed by concatenating f_1 and f_2:

$$\mathbf{x} = (\{f_1(\kappa_i)\}_{i=1,...,n_\alpha}, \{f_2(\kappa_i)\}_{i=1,...,n_\alpha}), \tag{3}$$

where $i = 1, ..., n_\alpha$ is the bin index of the histograms.

The complete curvature state of each sample window is thus a triplet $(\mathbf{x}, e, \alpha^*)$, where the feature vector $\mathbf{x}$ identifies the rotationally-invariant local micro-topography, the eccentricity e of Equation (1) characterizes a degree of anisotropy and α^* of Equation (1) holds the orientation. The orientation is indefinite when isotropy is small: $e \approx 0$. Figure 6 shows the directional curvature over the TIN models of Trails 1 and 2. The image has been constructed from the sample grid using the rough directions $\alpha^* + \pi/2$ at each sample grid center.

Figure 7 depicts two spots, one with ruts and one without. The size of the depicted squares is 3.8 m. The boundary inference and noise from the surface vegetation tends to be isotropic, affecting both perpendicular histograms f_1 and f_2 about the same way, thus not contributing much to the eccentricity e, which is related to the difference of the histograms. A vegetation effect is seen at the upper pair of the curvature images and one boundary effect (a dark patch in the height image) in the lower row.

2.7. Manual Selection of Trails

It is typical to have the presence of older trails from previous operations, and it seems difficult to direct any automatic detection and analysis to the correct recent trails. Hence, manual insertion of proximate trail control points was used. The control points initialize a more accurate numerical trail detection phase described in the next section. The control points are depicted in Figure A4 of Appendix B.

2.8. Steps 4, 5: Profile Evaluation

The first part of the profile evaluation consists of an accurate detection of the trail center line. The second part is about forming the rut depth profile.

The TIN model was rasterized to a height image of a 0.2-m grid length, which was subjected to a convolution filter depicted in Figure 8 using 20 different orientations. The filter was specifically designed to match the trail created by a typical forest forwarder. The convolution computation in each direction α is fast, and thus, it was performed for the whole area at once. The actual formulation is somewhat involved, since the convolution of ruts is performed by two separate runs and then matched (by an ordinary raster multiplication) from the resulting two images using the rut separation D (the axial length of the forest machine). Convolution in two parts seems to produce a better signal than one single convolution consisting of the left and right ruts. The latter creates three responses, the extra ones with one half of the magnitude of the middle one.

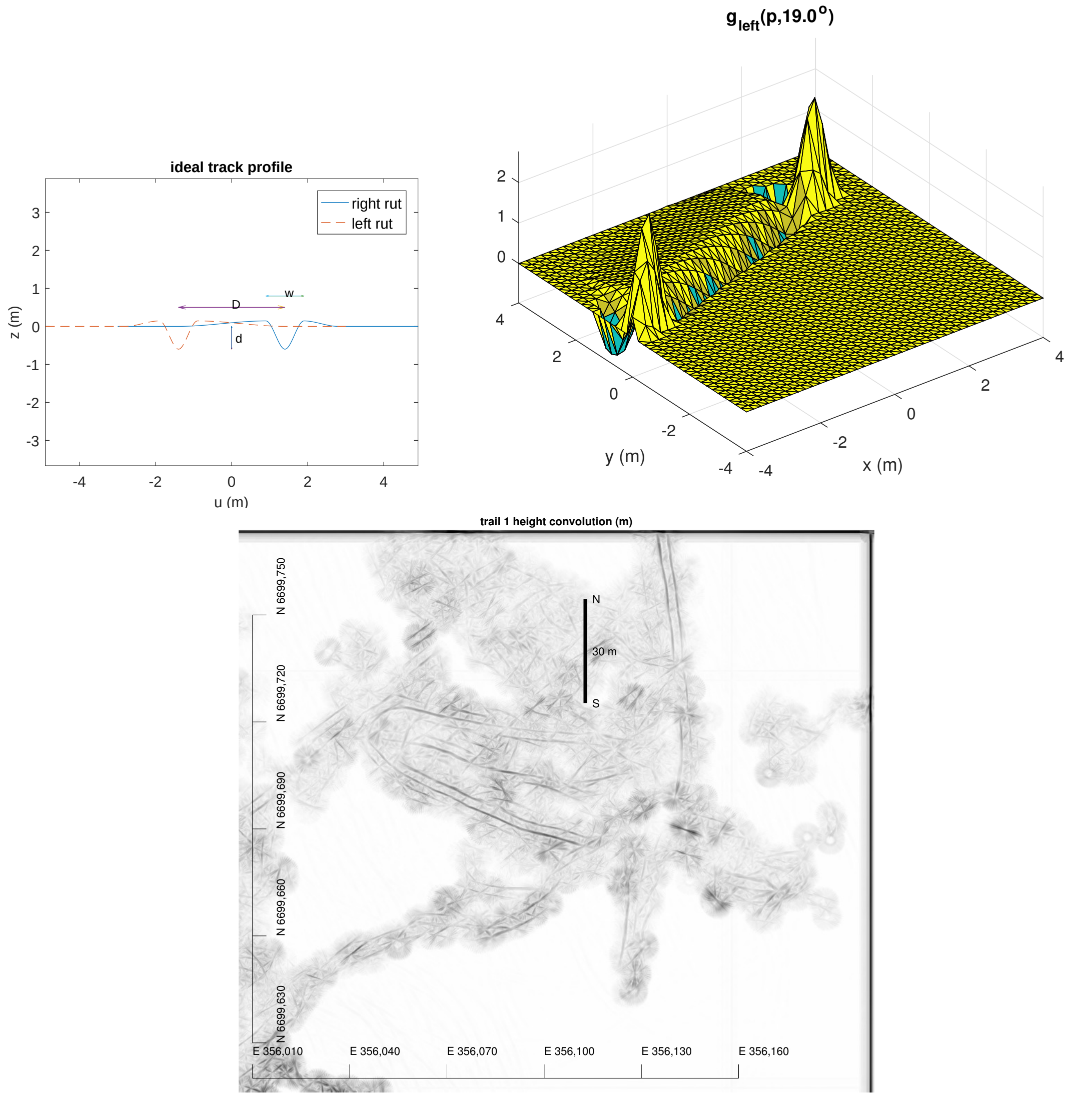


Figure 8. Left: The left and the right rut have mirror filters within a distance D, which can be set for each machine type. See Appendix B for the exact definitions. The ideal profile has an average nature, which is more pronounced at the average profile cumulated over the trail length. **Right**: The filter $g_{left}(p, \alpha)$ of the left rut in an orientation $\alpha = 19°$. The shape is formed by cubic splines and has a theoretically correct shape, e.g., to detect constant, gradient and unit impulses. **Bottom**: The result of the height convolution of Trail 1 in ETRS-TM35FIN co-ordinates. The image is a combination of the strongest responses of each orientation α.

Given the normal 2D convolution operator $\star$, the convolution signal c is:

$$c(p, \alpha) = [z(p) \star g_{left}(p, \alpha)][z(p) \star g_{right}(p, \alpha)], \tag{4}$$

where $z(p)$ is the rasterized TIN height at location p. The ideal trail profile as a combination of filter functions $g_{left}(.)$ and $g_{right}(.)$ of Equation (A5) with the local trail orientation α has been depicted as a projection along the trail in Figure 8 (top left). The top right detail is the convolution filter function g_{left} oriented in this depicted case in a direction $\alpha = 135°$. See the rut profile convolution parameters D, w, d (rut separation, rut width and depth, respectively) used for this particular case in Figure 8 and their values in Section 2.9. The profile convolution is one ingredient to be maximized along the smooth trail center line. The trail line 'finds' the response of two ruts depicted by the black end of the grey spectrum in Figure 8 (below). The multiplication arrangement of Equation (4) reduces extra stripes in the image, and the trail pattern triggers only once. Areas without a good signal are spanned by a strong continuity restriction and control points inserted manually. See the further details in Appendix B.1.

The manual points of Figure A4 outlining the initial trail were used to match a Cornu spline [34] with linear curvature change over the spline length. This spline allows the addition and removal of spline segments without altering the spline shape. The spline was then adjusted by maximizing the convolution match along the trail. The classic nonlinear regularization is based on the curvature squared integrated over the spline length (akin to the nonlinear elastic deformation energy of a thin beam).

A further adjustment was performed using the trail coordinate system with trail relative length t along the spline and the distance v from the spline as the new coordinates. The adjustment involved local shifts along the v axis to track the rut trajectory exactly. Section 3 presents the detected rut depths along the rut length and the average rut cross-profile. The algorithmic details are presented in Appendix B.

2.9. Parameterization

Table 2 lists all the parameters used. Actual cross-validation verification of the values shown has not been included. Some justifications of the values have been given in the text.

Table 2. The 25 parameters, their values and a short explanation.

Phase	Param. and Value	Explanation
thinning	$l_{target} = 0.2\,\text{m}$	thinning limit
thinning	$0.1\lvert P\rvert$	*stopList* size limit
thinning	$0.5 l_{target}$	triangle edge length limit
vectorization	$r = 2.1\,\text{m}$	sample radius
	$\delta = 3.5\,\text{m}$	sample grid interval
SAF canopy	$\omega_1 = \pi$	pike limit
	$\omega_2 = 3\pi$	hole limit
SAF ground	$\omega_1 = 4.4$	
	$\omega_2 = 8.1$	
MCF	$\lambda = 0.7$	smoothing degree
thinning	$l_b = 0.2\,\text{m}$	NN length at ground
	$l_c = 5.0\,\text{m}$	NN length at canopy front
trail detection	$n_\alpha = 20$	number of curvature directions
profile evaluation	$D = 2.8\,\text{m}$	rut distance
	$w = 1.2\,\text{m}$	rut width
	$d = 0.6\,\text{m}$	rut depth (no actual effect)
	$r_{rut} = 3.8\,\text{m}$ medium	convolution sample length
	$\Delta L = 0.2\,\text{m}$	rut profile sampling interval
	7 free shape params.	see Table A1
convolution fit	$\lambda_1 = 0.7\,\text{m}^2$	regularization weight
	$\Delta l = 0.6\,\text{m}$	convolution sampling frequency

3. Results

The initial filtering by SAF, MCF and thinning described in Algorithm 1 produces the point cloud, which is the basis for the further curvature analysis. An ideal regular triangular lattice with a 20-cm side length (the set value used) has the horizontal point cloud density $\rho_{ideal} = 14.4\,\mathrm{m}^{-2}$. The resulting ground model has a very high quality with density very close to the ideal one; see Table 3. The given values are for the canopy openings, since the earlier processing reduces the density of the canopy area to a much lower value ($\rho < 1\,\mathrm{m}^{-2}$; see Figure 6).

Table 3. The point cloud properties. Horizontal densities ρ are in m^{-2}. The abbreviations TP, TN, FP and FN correspond to true and false positives and negatives given as percentages.

Trail	Initial ρ	ρ TIN	ρ after Thinning	TP (%)	TN (%)	FP (%)	FN (%)
1	190	120	16	19	51	26	4
2	180	117	15	38	24	24	14

TIN models were generated from the photogrammetric point cloud at two target areas (Trails 1 and 2), in Figure 1; curvature vectors were recorded from a regular grid of sample locations (see Table 2) using the histograms of dominant directions, and the rut profiles were analyzed from the TIN model. Some results like the canopy and ground TIN models have been made available for an undetermined time period; see the Supplementary Materials on p. 18.

In Finland, the Government Decree on Sustainable Management and Use of Forests (1308/2013) based on the Forest Act (1093/1996) and related field control instructions by the Finnish Forest Centre, Helsinki, Finland (2013) regulate that rut depths of 10 cm (mineral soils) and 20 cm (peatlands) are classified as damage and can be detrimental to the forest growth. Table 3 gives a summary of detecting particularly the 20-cm depth: P (positive) stands for a depth more than 20 cm, N (negative) for a depth less than 20 cm and, e.g., TP and FN stand for the ratio of the sum of correctly-detected deep rut sections and deep sections not recognized, correspondingly. The overall accuracy (sum of TP and TN weighted by the trail lengths) is 65%. The performance is adequate for the purposes of the quality assurance of forest operations considering the amount of data that can be possibly collected compared to traditional methods.

After manual identification of the control points on trails, the center lines of trails were adjusted automatically by the convolution penalty, and the rut depth profiles were formed. Figure 9 shows the depth distributions along both trails on both manual and UAV measurements. Really deep depressions are rather rare, and they tend to become detected better. The UAV measurement detected much more of the depth 0.2–0.3 m range than the manual measurement on Trail 1. Trail 2 has a good correspondence between the manual and UAV measurements. The trail depth classification is in general worse in the presence of nearby trees.

Pearson's r correlation was taken between UAV and manual rut depth values at each manual measurement point; see Figure 10. The correlation $r = 0.67$ relates to the following inaccuracies: horizontal placement error of the point cloud and the reference level estimation in the manual measurement.

Figure 11 demonstrates how large differences can occur between two ruts. Some rocks and roots (each correctly identified) force the rut to have positive (above the reference level) heights. Multiple passes deteriorate the shape of the rut, but the average shape is relatively constant over large distances and similar terrain. The averaged trail profile shows typical displacement of the soil. The effect of slopes has been removed, but a long sideways slope causes one of the ruts to become deeper (the right rut of Trail 2 in Figure 11).

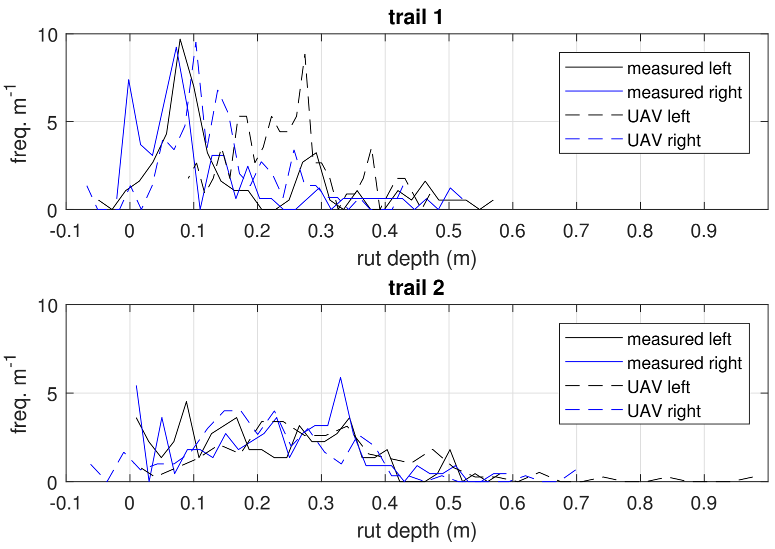


Figure 9. Trail 1 and 2 rut depth distributions over the rut length. The distributions of manually-measured rut depths and the rut depths extracted from the UAV photogrammetric point cloud. Each rut (left and right) of each trail has been plotted separately. Trail 1 has occasional deep depressions, whereas Trail 2 has a dominantly moderate rut depth.

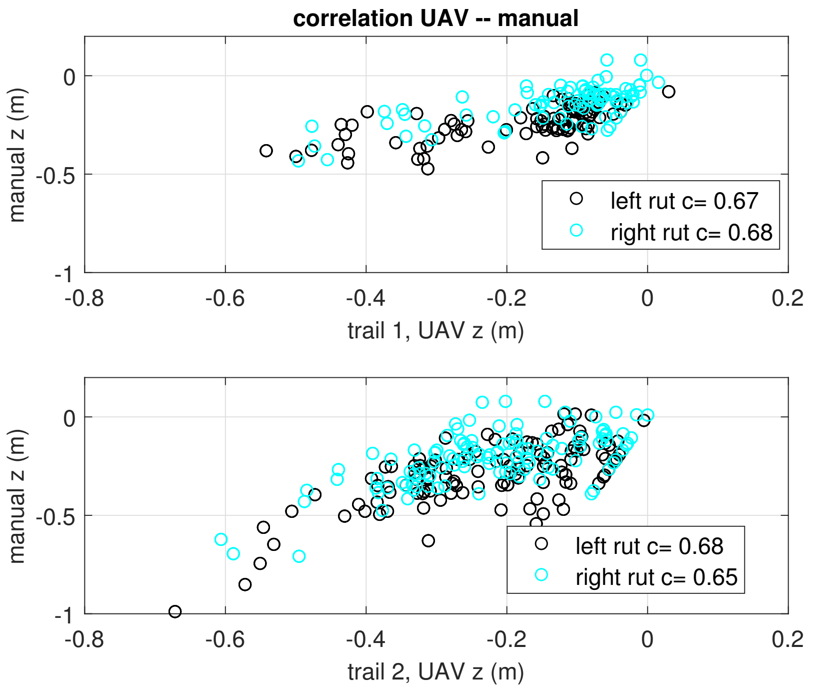


Figure 10. Correlation of the left and right rut depths measured manually and by the UAV method. Trail 1 at the top and Trail 2 at the bottom. The UAV values are extracted from the ground TIN model at the manual measurement points by the standard barycentric interpolation.

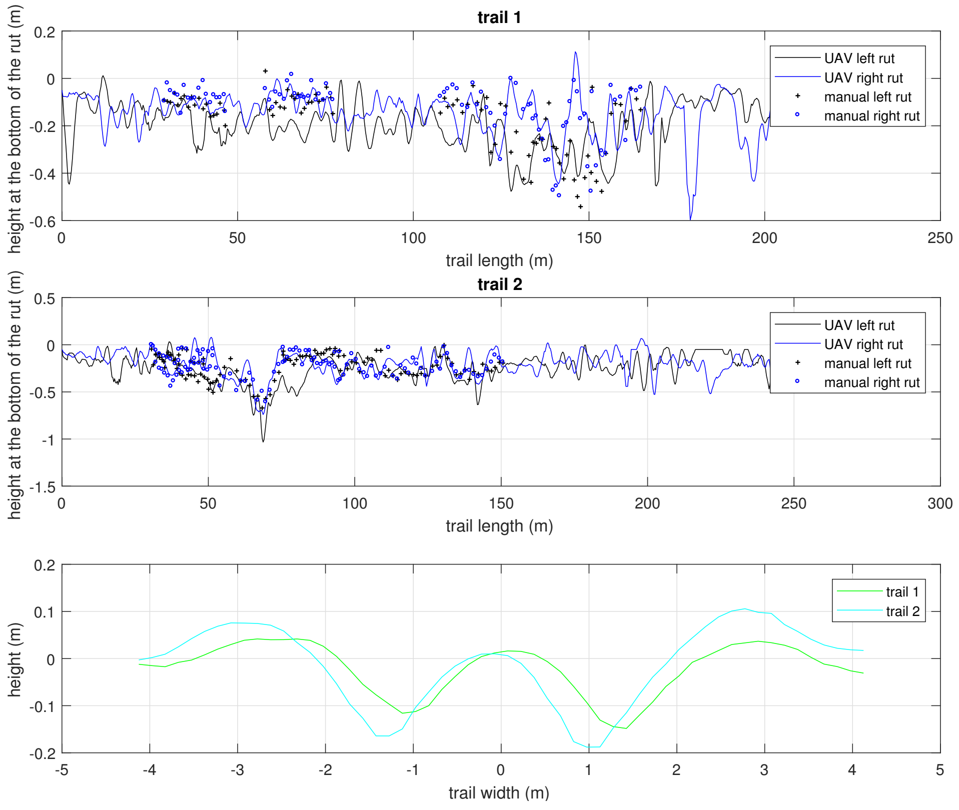


Figure 11. **Top** and **middle**: The depth profiles of the left and right rut of both trails extracted from the UAV photogrammetric point cloud. **Bottom**: The 3D trail cross-profile averaged over the length. The left and right ruts of each trail are visible. The reference height is set to $z = 0$ m. The narrow bottom of the ruts is an artificial result of the final rut profile rectification.

4. Discussion

The experiments of registering the rut depths by UAV photogrammetric point cloud are a part of a larger research project, e.g., [22], on traversability prediction. Thus, the site was emulating a typical forestry operation dictating payload amounts, machinery used and number of passes.

Figure 9 shows how the UAV measurement detected much more of the depth 0.1–0.3 m range than the manual measurement on Trail 1. This can be explained by difficulties in defining the control surface level in a compatible fashion by both methods (manual and UAV) on certain terrain conditions. The manual reference level tends to be assessed by humans from the proximity of the rut, whereas the UAV method fits the reference height origin (0-m level of the trail cross-profiles of Figure 11). Neither reference level is imperfect per se, but it seems that the calibration measurements should be done within an artificial environment with absolute measures, or with a ground-based LiDAR scanner, for example, which provides the chance to match the reference height of two point clouds reliably.

The point cloud need not to be geo-referenced nor matched to the DEM. This allows a UAV flight path that is reduced to cover only the trails with only a necessary amount of overlapping to enable the photogrammetric 3D point cloud generation. The RGB information would add to trail registration performance, but requires more research and more varied test materials with different weather, soil, terrain and light conditions.

The following is a short presentation of the observations of the experiments with alternative techniques (not including the proposed methodology presented in the Section 2.3):

Point cloud preprocessing: We extracted the curvature state of the ground model in order to expose the trails from the data. Subtracting the DEM height from the point cloud provides an easy way to eliminate the canopy points, but the resulting algorithm would not cope well in countries and sites with no ubiquitous DEM model. Furthermore, it requires careful planning and placement of the geo-referencing markers to reach the, e.g., $\pm 1\,\mathrm{m}/100\,\mathrm{m}$ height accuracy required with the two sites used in this study.

Ground height model: We produced TIN using a solid angle filtering [25] method. The local linear fit of [35] is a computation-intensive method, which does not adapt well to the noise from the canopy wall around the trail. Various heuristics such as taking the local mean of the lowest point cloud points after a space division to small compartments (approximately 20–30 cm) forces a repetitive inefficient computation and is not suitable for parameter optimization. Traditional local quadratics methods applied to the 3×3 raster window used to derive the DEM models [36] in geographic information systems (GIS), when scaled to the UAV point cloud density ($\approx 80\,\mathrm{m}^{-2}$ at the surface), smoothed the rut contour too much and did not provide adequate control for proper signal processing optimizations.

Vectorization: Vectorization is needed to detect a possible rut present locally and to assess the possible orientation of it. We used a novel histogram of curvatures (HOC) approach. There are several possibilities uncharted yet, including image difference methods based on entropy and information measures. Furthermore, all possible sampling radii and sample grid densities have not been fully covered yet. No adequate combination of machine learning methods, features and cloud preprocessing has been found yet to detect the ruts automatically with a reliable performance.

Neighborhood voting: It is possible to improve the track analysis by smoothing and strengthening the orientation information. There are several promising neighborhood voting methods, which need to be adapted only a little to this problem, but the basic signal from the vectorization phase has to be improved first.

Finding the trail center line: Trails can have a complicated structure, the proper handling of which has not been added yet. Early experiments with principal graphs using the software of [37] seem to be capable of handling loops, branches and junctions. It seems to be mathematically possible to utilize the local orientation information of [37], but at the moment, the noise level of the orientation is too high. The noise signal of the small trees, which do not completely get eliminated by solid angle filtering (SAF) [25], is particularly problematic. Multiple classification to ruts/young trees/open ground/canopy could be a solution.

Several experiments were made to automate the trail detection. Two major difficulties are: (1) finding methods generic enough to cope with most of the environmental conditions, especially with trails partially covered by the canopy; and (2) speeding up the initial processing steps. At the moment, the initial point cloud generation by photogrammetry is too slow for any online approach that would allow the automated flight control and autonomous flight path planning. At the moment, one is limited to a pre-defined flight pattern and batch processing after the field campaign.

5. Conclusions

We aim at a streamlined and economical workflow, which could be used after the harvesting operations both for collecting extensive ground truth data for trafficability prediction models and as a generic post-harvest quality assessment tool. A novel method of cumulating histograms of directed curvatures (HOC) is proposed, which reduces the computational burden of curvature analysis of the TIN samples. We demonstrate the procedure comparing the results with the field data collected from a test site. The procedure contributes towards automated trail detection and rut depth evaluation.

The proposed procedure can classify rut depths into two categories (insignificant depression/harmful rut depth) with practical precision using approximately 10 UAV images per 100 m of trail length. It is relatively inexpensive, since it is independent of the following conditions:

- before-after type of data collection
- GPS data of harvesting routes
- geo-referencing for utilizing the digital elevation maps (DEM)

The UAV requires the visual contact of the operator. This is more a legislation restriction, which could be removed in the future.

A proper point cloud pre-processing seems to be essential when using UAV point clouds for rut depth analysis and similar micro-topography tasks. The profile analysis based on manual control points is already ready for a practical tool implementation, while the pre-processing requires some practical improvements to be used with cross-validation or with neural network methods. Especially the segmentation into several classes, e.g., young trees, dense undergrowth and ruts, could be useful. Furthermore, a combination of the slope and curvature histograms and other pattern recognition methods has to be tested in the future.

The flight height of 100 m seems adequate with the equipment used. Experiments have to be performed with atypical flight plans with no geo-referencing and focusing the flight path on the trails more closely. These experiments will require direct structure-from-motion methods that are based on numerically-estimated camera positions, and the resulting point cloud has slightly lower spatial accuracy [38].

The 65% accuracy in classifying deep ruts (depth of over 20 cm) is adequate for practical purposes, e.g., as a post-harvest quality assurance. More extensive calibration data have to be produced to evaluate the performance in order to contribute to the on-going research of trafficability prediction. The combination of the canopy height model produced by UAV photogrammetry and the ground TIN produced by a more deeply-penetrating aerial LiDAR scan has to be studied in the future. This combination could contribute to understanding the relation between the rut formation and tree roots.

An efficient pipeline for rut depth field measurements is an essential step in gaining understanding about the interplay between environmental factors described by the public open data and the scale and nature of the variations caused to, e.g., cross-terrain trafficability [22] by the micro-topography. The current size of the test site is not adequate for final assessment of the future rut depth classification performance, but the suggested methodology based on UAV photogrammetry, ground TIN based on the SAF method, directional curvature analysis and using either manual or automatic trail detection seems to have potential.

As [16] states, the reference ground level (namely the ground level before the trail was formed) remains difficult to define by any means, including the best possible ground-based laser scanning and human assessment. The rut depth evaluation has to have some categorical element, addressing mainly two or three rut depth classes.

In the long run, the cloud pre-processing + SAF + MCF + thinning should be implemented either using existing software or a separate application to be developed. There is an on-going effort to streamline this process so that it behaves monotonically under usual cross-validation procedures in order to allow efficient parameter optimization for various machine learning tasks. The HOC procedure (generation of directed curvatures directly from TIN) is a novel generic technique, which will be a part of future attempts at vectorizing the ground models of both LiDAR and UAV photogrammetric origin in order to classify ruts and other useful micro-topographic features automatically at a large scale.

Supplementary Materials: The following are available online: http://users.utu.fi/ptneva/ALS/additional.pdf: A semi-informal constructive proof of triangular mean curvature yielding the directional curvature by the projected vertex normals under the barycentric interpolation scheme. https://seafile.utu.fi/f/5e8b21efdd0448ae813e/?dl=1: A part of the original point cloud (trail 1 curve), the canopy and the ground TINs as a ZIP file.

Acknowledgments: This work was supported by funding from the Academy of Finland (Grant 295337). Field studies were carried out in collaboration with the project ‘Enhancing efficiency and quality of forest operations by benefiting information related working conditions (Metsäoperaatioiden tehostaminen ja laadun parantaminen olosuhdetiedon hyödyntämisen avulla)’ coordinated by the Natural Resources Institute Finland LUKE. The photogrammetric points clouds were collected by Metsälinkki Ltd./Juuso Hiedanpää.

Author Contributions: Paavo Nevalainen performed the point cloud analysis, method design, method implementation, writing the article and the content and organization of additional materials. Jari Ala-Ilomäki and Juuso Hiedanpää designed the field measurements. Juuso Hiedanpää conducted the data collection. Aura Salmivaara contributed to the analysis of the contemporary research. Aura Salmivaara, Samuli Lauriainen and Tapio Pahikkala contributed to the article content. Jari Ala-Ilomäki, Jukka Heikkonen and Leena Finér defined the goals and general setting of the research. Tapio Pahikkala, Samuli Launiainen and Jukka Heikkonen participated in the verification of the methods and the results.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ALS	aerial laser scan
CAN-bus	controller-area network bus
CV	cross-validation
DEM	digital elevation map
DDG	discrete-differential geometry
ETRS-TM35FIN	co-ordinates based on European Terrestrial Reference System 1989. The central meridian is 27° and false easting 500 km.
FSC	(national) Forest Stewardship Council
GIS	geographic information system
GPS	geo-positioning system
GNSS/ISS	global navigation satellite + inertial system
GSD	ground sample distance
HOC	histogram of oriented curvatures
HOG	histogram of gradients
LiDAR	light detection and ranging
MCF	mean curvature flow
NLS	National Land Survey of Finland
NN	natural neighbors
PEFC	Programme for the Endorsement of Forest Certification Schemes
SAF	solid angle filtering
SFM	structure-from-motion
TIN	triangular irregular network
UAV	unmanned aerial vehicle

Appendix A. Directed Curvature on TIN

This Appendix documents some computational definitions and details to produce the curvature histograms efficiently. There are three relevant theorems, the proofs of which are provided in the Supplementary Materials.

The TIN vertex normals are calculated from triangle face normals using the projective tip angles; see Section 2 of [33]. The mean curvature H_t of a triangle t with the area A_t, vertex points a, b, c and vertex normals $n_p,\ p \in \{a, b, c\}$ is defined as follows [33]:

$$H_t(n_a, n_b, n_c) = \frac{(n_b - n_a) \times (c - a) + (b - a) \times (n_c - n_a)}{4A_t} \tag{A1}$$

The relevant theorems in the Supplementary Materials are the following:

1. The averaged mean curvature H_t of Equation (A1) is constant over the triangle t.
2. The directed curvature $H - t(\alpha)$ of Equation (A3) $H_t(\alpha)$ with orientation α is constant over the triangle t.
3. The directed curvature can be calculated by Equation (A3) with substituting original vertex normals n_p of Equation (A1) by the projected vertex normals m_p of Equation (A2).

Theorems 1 and 2 hold when the barycentric interpolation scheme (see, e.g., ([35], Appendix II)) is applied to the normal map defined by vertex normals.

The orientation plane is defined by the orientation angle α, and the projection $n_p \rightarrow m_p$ is arranged by:

$$\begin{aligned} \bar{\alpha} &= (\cos(\alpha), \sin(\alpha), 0)^T \\ P(\alpha) &= e_3 e_3^T + \bar{\alpha}\bar{\alpha}^T \\ m_p &= (P\, n_p)^0,\ p \in \{a, b, c\}, \end{aligned} \tag{A2}$$

where $.^T$ is a vector transpose, e_3 is the vertical unit vector, $P(\alpha)$ is a projection matrix and $v^0 = v/\|v\|$ denotes scaling a vector v to be a unit vector. The oriented curvature is:

$$H_t(\alpha) = H_t(m_a, m_b, m_c)\,. \tag{A3}$$

When the histogram of oriented curvatures $hist(\{(H_t(\alpha), A_t)\}_{t \in \text{sample}}$ is being built, each oriented curvature value $H_t(\alpha)$ gets weighted by the triangle area A_t.

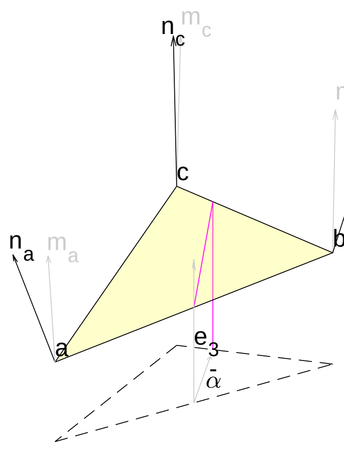


Figure A1. Left: Projection of vertex normals on the directional plane spanned by vectors $\bar{\alpha}$ and e_3. **Right**: Splitting of the raster rectangle (a, b, c, d) to four triangles.

Equations (A1)–(A3) have to be repeated n_α times. Numerical benefits can be gained by projecting all the vertex normals n_p of a set of triangles T of a sample area, $p \in t \in T$ at the same time in a matrix $\mathbf{N} = \{n_p\}_{p \in \text{point cloud}}$, and re-arranging the vertex points of Equation (A1) to an assembly matrix $\mathbf{W}$, so that:

$$\mathbf{H}(\alpha) = [P(\alpha)\mathbf{N}]^0\, \mathbf{W}, \tag{A4}$$

where $\mathbf{H}(\alpha) = \{H_t(\alpha)\}_{t \in T}$ and the unit scaling $[.]^0$ concerns each column of $P(\alpha)\mathbf{N}$. The sparse matrix $\mathbf{W}$ has only six non-zero elements on each column on average. This allows one to use a relatively high number of orientations α (16 in our application) allowing, e.g., the smoothing of the local window histograms $f(\kappa_{\alpha_j}), j = 1, ..., 16$ by referring to the two neighboring histograms $f(\kappa_{\alpha_{j-1}})$ and $f(\kappa_{\alpha_{j+1}})$. The window histogram $f(\kappa_{alpha_j})$ is cumulated from a subset of terms in $\mathbf{H}(\alpha_j)$ concerning triangles within a single circular sample spot with a radius r; see Table 2 and Figure 7. The histogram of curvatures (HOC) method consists of generating the histograms by Equation (A4) and choosing the vectorization by two dominant orientations of Equation (3). There are several other possible alternatives for Equation (3), however. A similar method exists for the DEM raster data, but it has been excluded from this paper.

Appendix B. Profile Analysis

Appendix B.1. The Convolution Filter

The convolution is for matching the trail center line to a pair of ruts (a trail). The convolution filter $g_{left}((x, y), \alpha)$ of Equation (4) for detecting a rut (left one in this case) was constructed from piecewise cubic functions. The following description omits the technical details of necessary rotation due to the orientation angle α. Each interval of Figure A2 is completely defined by specification of the nodal values $(v_i, z_i, z'_i), i = 1, ..., 5$ and $(u_i, z_i, z'_i), i = 6, ..., 8$, where v is the dimension across the trail, u runs along the length of the trail, z is height and z' is the derivative of height (either $\frac{dz}{dv}$ or $\frac{dz}{du}$, depending on the index i). Table A1 holds the necessary parameters, when given the rut width $w = 0.6$ m and the convolution sample length $r_{rut} = 3.8$ m. There could be a non-parametric convolution profile for most typical environments and machinery defined by cross-validation measures, but the current amount of data is not enough for that.

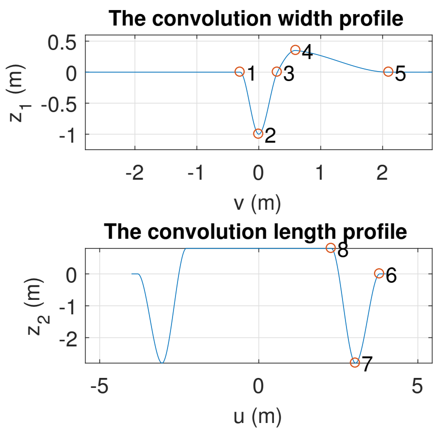


Figure A2. The shape of the convolution filter. **Top**: The left rut profile construction. **Bottom**: The length profile construction. See also Figure 8 for the general view.

Table A1. Convolution shape parameters. See Figure 8 and Table 2.

i	1	2	3	4	5	6	7	8
v_i or u_i	$-w/2$	0	$w/2$	w	$7w/2$	r_{rut}	$0.8r$	$0.6r$
z_i	0	-1	0	0.35	0	0	-2.8	0.8
z'_i	0	0	2	-0.1	0	0	0	0

The final convolution functions g_{left} and g_{right} are constructed from the two orthogonal components $z_1(v)$ and $z_2(u)$:

$$\begin{aligned} g_{left}(u, v) &= z_1(v - D/2) z_2(u) \\ g_{right}(u, v) &= z_1(-v + D/2) z_2(u), \end{aligned} \tag{A5}$$

to which necessary rotation by orientation α dictated by the trail trajectory and translation is applied to map between the real-world coordinates (x, y) and the trail-specific coordinates (u, v). The convolution raster $g(x, y)$ in the globally-rotated coordinates is depicted in the upper left part of Figure 8. See the definitions of the rut profile parameters D, w, d, r_{rut} from Figure 8 and their values from Table 2.

Appendix B.2. Forming the Height Raster

The height raster is used in the convolution phase. A rectangular height raster with a 0.2 m $\times$ 0.2 m grid was created from the TIN in order to apply the fast raster convolution of the previous section. The chosen raster size leads to 3.8 points per raster slot, which is about the same relative sample density as used in constructing the national DEM model from the aerial LiDAR data. Each slot was assigned the mean height of the ground point samples on the slot. This is a very crude rasterization policy, but serves its purpose as the domain of the numerical height convolution.

Appendix B.3. Convolution Fit of the Trail Center Line

Manual control points initialize an iteration to find out a smooth centerline of a trail consisting of two parallel ruts. An interpolation scheme with piecewise linear curvature [34] (Euler or Cornu spiral) was chosen. This spline has as few parameters (one curvature) per segment as a piecewise linear curve (one orientation), and it provides a high quality transform for the surrounding local point cloud. Figure A3 depicts the spline fitting arrangement. Formally, let $a = (q_1, \alpha_1, \kappa_1, \kappa_2, ..., \kappa_m)$ be a parameter vector, where q_1 is the first point (given manually), α_1 is the initial trajectory orientation at q_1 and κ_i are curvatures at each control point $q_i \in Q$, where the set of control points is initially $Q = \{q_i\}_{i=1,...,m}$. Then, $S(a) \in \mathbb{R}^2$ is the spline in question. Using a continuous index $t \in [1, m] = \mathcal{T} \subset \mathbb{R}$, one can define the sub-segments $(q_{t_j}, q_{t_{j+1}})$, $[t_j, t_{j+1}] \subset \mathcal{T}$. Sub-segments can be generated so that they have approximately the same length: $\Delta l = \|q_{t_{j+1}} - q_{t_j}\|$, where $\Delta l = 0.6$ m has been chosen to sample typical ruts. A sub-segment is depicted in Figure A3.

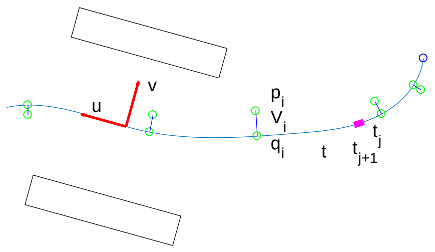


Figure A3. A schematic presentation of the trail center line matching. A convolution filter positioned at $q(t) \in S(a)$ has been outlined. A sub-segment $[t_j, t_{j+1}] \in \mathcal{T}$ is shown with a thick line. The local coordinate frame (u, v) can also be seen in Figure A2.

The curvature $\kappa(t_j)$ can be now defined at each sub-segment end q_{t_j}:

$$\kappa(t_j) = \frac{2\phi_j}{\|q_{t_{j+1}} - q_{t_j}\| + \|q_j - q_{j-1}\|},$$

where ϕ_i is the direction change between segments at q_{t_j}. The curve fitting uses the square of the curvature as the regularization term to find the optimal parameters α^*:

$$a^* = \arg\min_a \sum_{i=1}^{m} V_i^2 + \lambda_1 \sum_{t_j \in \mathcal{T}} \kappa(t_j)^2, \qquad \text{(A6)}$$

where $V_i = \min_{q \in S(a)} \|q_i - q\|$ is the distance between the control point q_i and the spline $S(a)$.

The convolution match should add a curve integral to the Equation (A6) to be maximized. That formulation has weak convergence properties, and an alternative with better convergence and the same end result has been adopted. Some of the control points $q_i \in Q$ are moved to new positions, where they have a maximal fit by the convolution response $c(., \alpha)$ of Equation (4). The movement is restricted to the perpendicular line from the control point q_i. The procedure uses the perpendicular local coordinate vector v depicted in Figure A3, and it has the following three steps:

$$V_i^* = \arg\max_V c(q_i + Vv, \alpha_i) \tag{A7}$$

$$q_i := q_i + V_i^* v, \tag{A8}$$

where α_i is the curve tangent orientation at q_i and := denotes the iterative updates of locations q_i. The update occurs every time when the orientation changes more than 15°. The orientations α_i and the possible updated distances V_i^*, $q_i \in Q$ can be recorded for further iterations. An addition of control points occurs progressively until a target density $L = 4$ m has been reached, and a deletion of a control point q_i occurs when it does not change the resulting curvature at the same position more than 10%. The two parameters $\Delta l, \Delta L$ presented here guarantee an acceptable convergence.

After the initial convergence, the fitting continues by upgrading segment points q_{t_j} to control points. The variable q_i has to be substituted by q_{t_j} in Equations (A7) and (A8), and a necessary set update $Q := Q \cup \{q_{t_j}\}$ has to be made. The addition of new points proceeds from one end of the trail towards the other, since the Cornu spline can have the last point undefined. The speed of the fit and the end quality are satisfactory. The end result is seen in Figure A4. Original manual points have drifted to new positions (circles), and new control points are not shown.

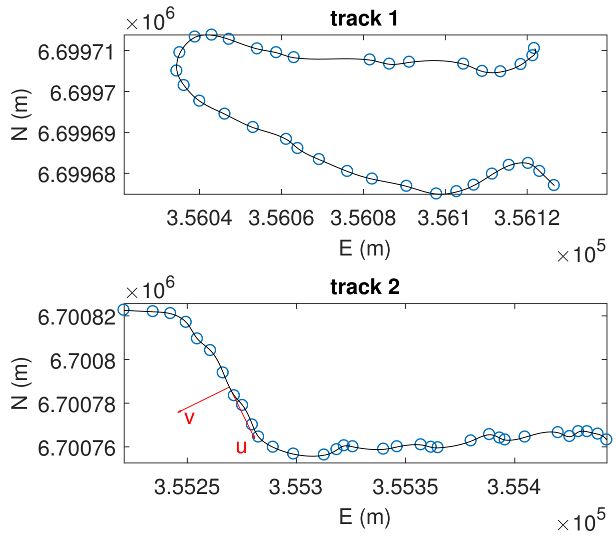


Figure A4. The trail central line detection by height convolution maximization. The local coordinates (u, v) are the same as in Figure A2.

Appendix B.4. Profile Adjustment

The previous step does not catch all the sharp turns of the trails. This can be seen from the first two trail profiles of Figure A5. An additional straightening is needed. This happens by shifting each (u, z) profile in such a way that the mean profile along the trail lengths keeps deepening.

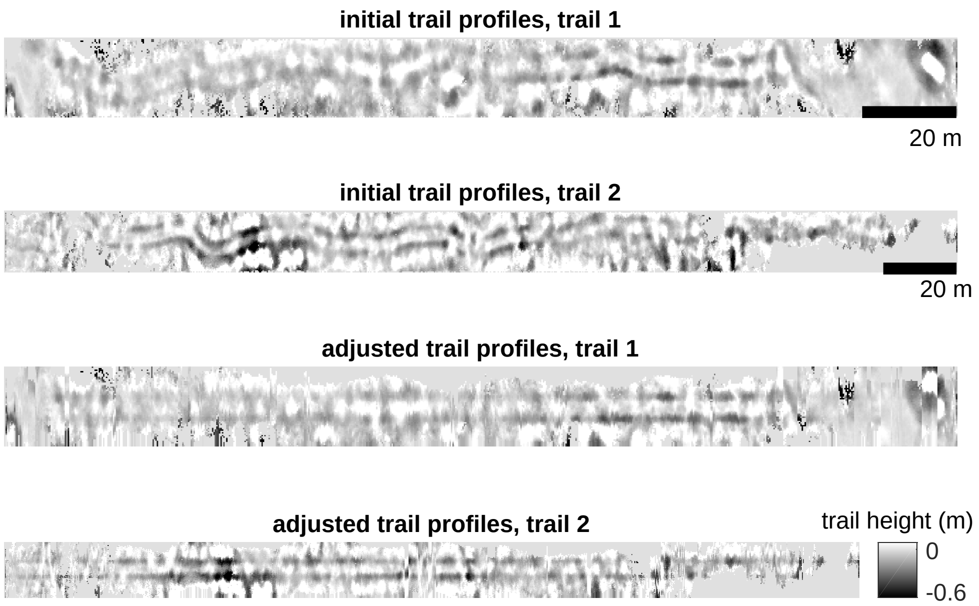
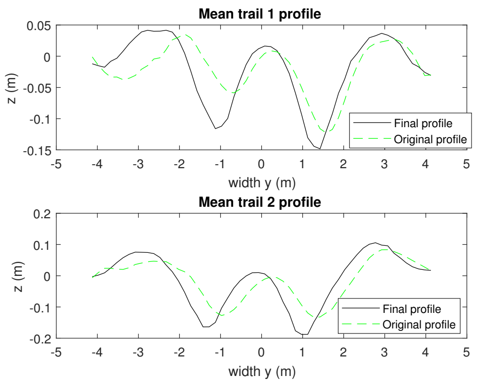

initial trail profiles, trail 1
20 m
initial trail profiles, trail 2
20 m
adjusted trail profiles, trail 1
adjusted trail profiles, trail 2
trail height (m)
0
-0.6

Figure A5. Top two: Trail profiles in the *u* direction after the trail center line has been fixed with the height convolution adaptation. Bottom: Trail profiles in the *u* direction after the local profile adjustment. Note: the height scale given concerns all height plots.

The computation is basically a minimization problem in 700–900 dimensional space (the number of translations of separate profiles each sampling 20 cm of the track length, e.g., with Trail 1: $180\,\text{m}/0.2\,\text{m} = 900$, where L is the trail length and ΔL is the profile slicing distance). The actual implementation is fast, though, since each individual translation is independent and needs to be computed only once. Figure A6 depicts the change of the mean rut profile from the initial (dashed line) to the adjusted (solid line). The final rut longitudinal depth profiles of Figure 11 have been produced from the center lines of the ruts of these adjusted profiles.



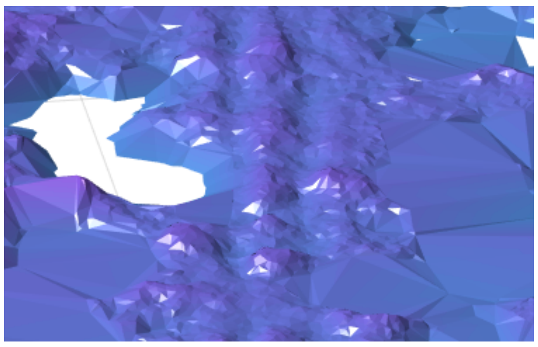


Figure A6. Left: Effect of the profile adjustment to the mean rut profile. **Right**: The TIN model before the final adjustment.

The original point cloud points $p = (x, y, z) \in P$ can be mapped to the projective coordinates defined by the spline $S(a)$ and the local coordinates (u, v): $(x, y, z) \to (t, v, z)$, where t can be scaled to the distance along the trail center line and v is the distance from the center line (including a local shift δv introduced in the previous profile adjustment step). The point cloud can be updated by a linear $O(|P|)$ computational cost during the iteration, thus allowing several other trail detection criteria than the height convolution. This is because a small change in the spline parameters a in each iteration step produces only a minor re-shuffling of co-ordinates t. All derived features (like curvature) naturally require a major re-computation, however. Figure A6 (right) shows a fragment of the rectified point cloud at Trail 1. A linear interpolation of interval $[t_i, t_{i+1}]$ end point normals was used for this detail.

References

1. Cambi, M.; Certini, G.; Neri, F.; Marchi, E. The impact of heavy traffic on forest soils: A review. *For. Ecol. Manag.* **2015**, *338*, 124–138.
2. Duncker, P.S.; Raulund-Rasmussen, K.; Gundersen, P.; Katzensteiner, K.; Jong, J.D.; Ravn, H.P.; Smith, M.; Eckmüllner, O.; Spiecker, H. How forest management affects ecosystem services, including timber production and economic return: Synergies and trade-offs. *Ecol. Soc.* **2012**, *17*, 50.
3. Owende, P.; Lyons, J.; Haarlaa, R.; Peltola, A.; Spinelli, R.; Molano, J.; Ward, S. Operations Protocol for eco-Efficient Wood Harvesting on Sensitive Sites. Developed by the ECOWOOD Partnership, 2002. Available online: http://www.ucd.ie/foresteng/html/ecowood/op.pdf (accessed on 8 December 2017).
4. Pierzchala, M.; Talbot, B.; Astrup, R. Measuring wheel ruts with close-range photogrammetry. *Forestry* **2016**, *89*, 383–391.
5. Quesnel, H.; Curran, M. Shelterwood harvesting in root-disease infected stands—Post-harvest soil disturbance and compaction. *For. Ecol. Manag.* **2000**, *133*, 89–113.
6. Sirén, M.; Ala-Ilomäki, J.; Mäkinen, H.; Lamminen, S.; Mikkola, T. Harvesting damage caused by thinning of Norway spruce in unfrozen soil. *Int. J. For. Eng.* **2013**, *24*, 60–75.
7. Vega-Nieva, D.J.; Murphy, P.N.; Castonguay, M.; Ogilvie, J.; Arp, P.A. A modular terrain model for daily variations in machine-specific forest soil trafficability. *Can. J. Soil Sci.* **2009**, *89*, 93–109.
8. Curzon, M.T.; D'Amato, A.W.; Palik, B.J. Harvest residue removal and soil compaction impact forest productivity and recovery: Potential implications for bioenergy harvests. *For. Ecol. Manag.* **2014**, *329*, 99–107.
9. PEFC. Programme for the Endorsement of Forest Certification: PEFC International Standard, 2010.
10. FSC. Forest Stewardship Council: FSC International Standard, FSC Principles and Criteria for Forest Stewardship, 2015.
11. Centre, F.F. Suomen Metsäkeskuksen Maastotarkastusohje (Field control instructions of Finnish Forest Centre). Technical Report, Finnish Forest Centre, 2016. Available online: http://www.metsakeskus.fi/sites/default/files/smk-maastotarkastuohje.2016.pdf (accessed on 8 December 2017).
12. Eliasson, L.; Wästerlund, I. Effects of slash reinforcement of strip roads on rutting and soil compaction on a moist fine-grained soil. *For. Ecol. Manag.* **2007**, *252*, 118–123.
13. Pierzchała, M.; Talbot, B.; Astrup, R. Estimating soil displacement from timber extraction trails in steep terrain: Application of an unmanned aircraft for 3D modelling. *Forests* **2014**, *5*, 1212–1223.
14. Uusitalo, J.; Ala-Ilomäki, J. The significance of above-ground biomass, moisture content and mechanical properties of peat layer on the bearing capacity of ditched pine bogs. *Silva Fennica* **2013**, *47*, 1–18.
15. Ala-Ilomäki, J. *Metsäisten Turvemaiden Kulkukelpoisuus (Trafficability of Forested Peatlands)*; Technical Report; Nature Resource Center of Finland (LUKE): Helsinki, Finland, 2005.
16. Salmivaara, A.; Miettinen, M.; Finér, L.; Launiainen, S.; Korpunen, H.; Tuominen, S.; Heikkonen, J.; Nevalainen, P.; Sirén, M.; Ala-Ilomäki, J.; et al. Wheel rut measurements by forest machine mounted LiDAR sensor. Accuracy and potential for operational applications? *Int. J. For. Eng.* **2018**, submitted.
17. Nouwakpo, S.K.; Huang, C.H. A simplified close-range photogrammetric technique for soil erosion assessment. *Soil Sci. Soc. Am. J.* **2012**, *76*, 70–84.
18. Haas, J.; Hagge Ellhöft, K.; Schack-Kirchner, H.; Lang, F. Using photogrammetry to assess rutting caused by a forwarder—A comparison of different tires and bogie tracks. *Soil Tillage Res.* **2016**, *163*, 14–20.
19. Koreň, M.; Slančik, M.; Suchomel, J.; Dubina, J. Use of terrestrial laser scanning to evaluate the spatial distribution of soil disturbance by skidding operations. *iForest Biogeosci. For.* **2015**, *8*, 386–393.

20. Lisein, J.; Pierrot-Deseilligny, M.; Bonnet, S.; Lejeune, P. A Photogrammetric Workflow for the Creation of a Forest Canopy Height Model from Small Unmanned Aerial System Imagery. *Forests* **2013**, *4*, 922–944.
21. Waldhauser, C.; Hochreiter, R.; Otepka, J.; Pfeifer, N.; Ghuffar, S.; Korzeniowska, K.; Wagner, G. Automated Classification of Airborne Laser Scanning Point Clouds. *Solving Comput. Expens. Eng. Probl. Math. Stat.* **2014**, *97*, 269–292.
22. Pohjankukka, J.; Riihimäki, H.; Nevalainen, P.; Pahikkala, T.; Ala-Ilomäki, J.; Hyvönen, E.; Varjo, J.; Heikkonen, J. Predictability of boreal forest soil bearing capacity by machine learning. *J. Terramech.* **2016**, *68*, 1–8.
23. Kim, A.M.; Olsen, R.C. Detecting trails in LiDAR point cloud data. In Proceedings of the SPIE, Baltimore, MD, USA, 14 May 2012; Volume 8379.
24. Kalbermatten, M.; Ville, D.V.D.; Turberg, P.; Tuia, D.; Joost, S. Multiscale analysis of geomorphological and geological features in high resolution digital elevation models using the wavelet transform. *Geomorphology* **2012**, *138*, 352–363.
25. Nevalainen, P.; Middleton, M.; Sutinen, R.; Heikkonen, J.; Pahikkala, T. Detecting Terrain Stoniness From Airborne Laser Scanning Data. *Remote Sens.* **2016**, *8*, 720.
26. Desbrun, M.; Meyer, M.; Schröder, P.; Barr, A.H. Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, Los Angeles, CA, USA, 8–13 August 1999; ACM Press/Addison-Wesley Publishing Co.: New York, NY, USA, 1999; pp. 317–324.
27. Wu, C. Visualsfm: A Visual Structure From Motion System. 2011. Available online: http://ccwu.me/vsfm/ (accessed on 8 December 2017).
28. Agisoft. *Agisoft PhotoScan User Manual: Professional Edition, Version 1.1.*; Agisoft: St. Petersburg, Russia, 2012.
29. De Berg, M.; Cheong, O.; van Kreveld, M.; Overmars, M. *Computational Geometry: Algorithms and Applications*, 3rd ed.; Springer: Santa Clara, CA, USA, 2008.
30. Devillers, O. On Deletion in Delaunay Triangulations. *Int. J. Comput. Geom. Appl.* **2002**, *12*, 193.
31. Mohan, M.M.; Silva, C.A.; Klauberg, C.; Dia, M. Individual Tree Detection from Unmanned Aerial Vehicle (UAV) Derived Canopy Height Model in an Open Canopy Mixed Conifer Forest. *Forests* **2017**, *8*, 340.
32. Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 1, pp. 886–893.
33. Nevalainen, P.; Jambor, I.; Pohjankukka, J.; Heikkonen, J.; Pahikkala, T. Triangular Curvature Approximation of Surfaces: Filtering the Spurious Mode. In Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2017, Porto, Portugal, 24–26 February 2017; Marsico, M.D., di Baja, G.S., Fred, A.L.N., Eds.; SciTePress: Setubal, Portugal, 2017; pp. 457–466.
34. Abramowitz, M. *Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables*; Dover Publications, Incorporated: Mineola, NY, USA, 1974.
35. Nevalainen, P.; Middleton, M.; Kaate, I.; Pahikkala, T.; Sutinen, R.; Heikkonen, J. Detecting stony areas based on ground surface curvature distribution. In Proceedings of the 2015 International Conference on Image Processing Theory, Tools and Applications, IPTA 2015, Orleans, France, 10–13 November 2015; pp. 581–587.
36. Hutchinson, M.F.; Gallant, J.C. *Terrain Analysis: Principles and Applications*; Wiley: Hoboken, NJ, USA, 2000; pp. 29–50.
37. Mao, Q.; Wang, L.; Tsang, I.; Sun, Y. Principal Graph and Structure Learning Based on Reversed Graph Embedding. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2227–2241.
38. Turner, D.; Lucieer, A.; Watson, C. An automated technique for generating georectified mosaics from ultra-high resolution unmanned aerial vehicle (UAV) imagery, based on structure from motion (SfM) point clouds. *Remote Sens.* **2012**, *4*, 1392–1410.

ISBN 978-952-12-3723-2
ISSN 1239-1883

Paavo Nevalainen Geometric Data Understanding

Paavo Nevalainen Geometric Data Understanding

Paavo Nevalainen Geometric Data Undestranding: Deriving Case-Specific Features