Evaluation of image matching search engines

UNIVERSITY OF TURKU
Department of Future Technologies
Computer Science
Master of Science Thesis
Mikko Kokkonen
2019

UNIVERSITY OF TURKU
Department of Future Technologies

MIKKO KOKKONEN Evaluation of image matching search engines

Master of Science Thesis, 58 pages
Computer Science
Dec 2019

Corner detection and feature detecting are essential parts of image matching. This master of science thesis goes through theory of image matching and how it can be used in different applications. The thesis introduces different feature detectors and convolutional neural network which are all generally used within image matching.

Data for this thesis was collected with literature review by examining image matching technologies such as speeded-up robust features feature detector. The information from image matching technologies was used in later parts of the thesis by creating an experimental image search engine and testing it with different feature detectors.

Testing the experimental search engine shows differences on how KAZE and SURF feature detectors behave with different kind of images. It also shows how different configurations like using different Hessian thresholds affect the results of image matching.

The thesis also goes through testing public image search engines with test image called Lena. Lena is a standard test image which is widely used in computer vision. This Lena image is used with different variations to see how well public image search engines perform when the image is altered.

Keywords: image matching, similarity search, image registration

TURUN YLIOPISTO
Tulevaisuuden teknologioiden laitos

MIKKO KOKKONEN

Kuvien vastaavuuden määrittävien hakukoneiden arviointi

Pro gradu -tutkielma, 58 sivua
Tietojenkäsittelytieteet
Joulukuu 2019

Kulmien ja ominaisuuksien tunnistaminen ovat olennainen osa kuvien vastaavuuden etsimistä. Tämä pro gradu -tutkielma käsittelee kuvan vastaavuuksien etsimisen teoriaa, sekä sen erilaisia sovelluskohteita. Tutkielma esittelee erilaisia ominaisuusilmaisimia sekä perusteet konvoluutioneuroverkoista, joita käytetään yleisesti kuvaparien vastaavuuksien tunnistamisessa.

Tietoa tähän tutkielmaan on kerätty kirjallisuuskatsauksella, joka koskee kuvan vastaavuuden etsimiseen käytettävien teknologioiden, kuten speeded-up robust features -tekniikan ominaisuusilmaisinta. Kuvien vastaavuuksien etsimiseen käytettävistä teknologioista saatua tietoa hyödynnettiin tutkielman myöhemmissä osissa luomalla kokeellinen kuvanhakukone ja kokeilemalla sitä käyttäen erilaistia ominaisuusilmaisimia.

Kokeellisella hakukoneella suoritetut testit osoittavat miten KAZE- ja SURF-ominaisuusilmaisimet eroavat toiminallisuudeltaan erilaisia kuvia tarkasteltaessa. Ne myös osoittavat miten erilaiset asetukset, kuten muutokset Hessian kynnysarvossa vaikuttavat kuvan vastaavuuden etsimisen tuloksiin.

Tutkielmassa tarkastellaan myös julkisten kuvanhakukoneiden testausta testikuva Lenan kanssa. Lena on standardi testikuva, jota käytetään yleisesti tietokonenäön kanssa. Lena-kuvan erilaiset variaatiot mahdollistavat näkemään, miten kuvanhaku käyttäytyy, kun kuvaa on muokattu.

Avainsanat: kuvan vastaavuus, yhtäläisyyshaku, kuvan rekisteröinti

TABLE OF CONTENTS

Abbreviations

OpenCV	Open Computer Vision Library
SURF	Speeded-Up Robust Features
QBIC	Query By Image Content
SIFT	Scale-Invariant Feature Transform
DoG	Difference of Gaussian
BRIEF	Binary Robust Independent Elementary Features
XOR	Exclusive Or
POPCNT	Population count
FAST	Features from Accelerated Segment Test
SUSAN	Smallest Univalue Segment Assimilating Nucleus
ORB	Oriented FAST and Rotated BRIEF
BRISK	Binary Robust Invariant Scalable Keypoints
FREAK	Fast Retina Keypoint
CBIR	Content-based image retrieval
ROI	Region of interest
CNN	Convolutional Neural Network
GPU	Graphical Processing Init
ASCII	American Standard Code for Information Interchange
REST	Representational State Transfer
API	Application Interface
JSON	Javascript Object Notation
OCR	Optical Character Recognition
WPF	Windows Presentation Framework
IPP	Integrated Performance Primitives
MVVM	Model-View-Viewmodel
CPU	Central Processing Unit
RAM	Random Access Memory

1 INTRODUCTION

The topic of this master's thesis is to research image matching. Research is done by literature review, testing online search engines and by developing an experimental search engine. Literature review will be used in the evaluation of the search engine to find reasons for differences on the behavior of the search engines in different situations.

Image matching stands essentially for browsing and searching images from large datasets based on the visual content of the images. These images can vary very much between themselves for example due to their quality, format and in what kind and amount of interesting data they have. Datasets holding the images can be huge which means that scanning the images can consume much computing resources. The datasets can also change dynamically for example, in situations where the image being matched originates from a camera stream.

Preprocessing of the data helps to remove unimportant images, however this usually requires specialized knowledge and tools. For example, some of the features in the image might be important for the object detection, and selecting such features requires knowledge of image processing. Also, the feature detection and image matching might require different approaches for different image datasets. One approach might be more accurate than the other but it can be much slower also. On the other hand, a faster approach might be too inaccurate to be able to provide proper results for image matching. These varying effects of the image matching methods mean that there are always tradeoffs between different methods. For example, one method can be faster but less accurate and still very complex to use. Another method can be easy to use and very accurate but it takes long running time.

Different image matching approaches are investigated in this thesis's literature review since some of the approaches might be more suitable than other for a given dataset. The thesis also contains an evaluation of different commercial engines for image matching. Some of the commercial engines are suited for special purposes. For example, already discontinued search engine tool called Ditto was focused on finding brand images.

In the first chapter the world of image matching is introduced to the reader. It describes on high-level what image matching is and where it could be used. It also describes the main problems in the image matching. The second chapter is deeper dive to image

matching. introducing different methods and techniques. The third chapter describes what are the current applications for image matching and where image matching is currently being used. It also tells a little bit about where image matching could be used in future when techniques and methods develop even further. The fourth chapter is about the open computer vision (OpenCV) library which was used in the present work to develop an experimental image matching search engine. The fifth chapter summarizes the results of this research. The research questions in this thesis are:

- How good is the open computer vision library for image matching?
- How good are the public search engines for image matching?
- Are public search engines using different techniques?
- If they are is it possible to see it in their performance?

Answers to these questions are done by researching the current online search engines and also by developing an experimental search engine using the OpenCV library. The sixth chapter contains conclusions and findings about image matching.

2 IMAGE MATCHING

Image matching which is also known as image registration, image fusion or image warping is a process of aligning two or more images. The image matching process is used to find the image which matches best with the input image or has some common features of it. Image matching itself is used in applications where valuable information is searched using one or more images (Oliveira, Tavares, 2014).

Earlier image matching techniques had two different methods for doing it. Two methods were *direct* and *feature-based* methods. Direct methods try to align images by iteratively searching for the best pixel alignment. Feature-based methods have different approaches for aligning images by finding and matching significant features from the image. These features are for example, edges and corners. Another approach for image recognizing and matching is to use invariant features which is at the intersection of direct and feature-based methods. Invariant features use large amount of local image data around significant features from the image. The collected data are then used to form invariant descriptors for indexing and matching (M. Brown, R. Szeliski & S. Winder, 2005).

Later in the early 1980s a new technique came for image matching. This new technique is called content-based image matching. The content-based image matching technique made it possible to index images by their visual content. Visual contents could be for example, color and texture (Müller et al., 2004). Recently, the research in image matching has transferred more to finding objects and understanding the image as a whole. This means finding images with objects like cars, cats, boats or houses. Several different methods have been proposed to help with the "description of images". These methods use the features extracted from the image to describe what is in it (Halawani et al., 2006).

Image matching systems have been under a very active research since the 1990s. During this time systems have been built for commercial and research purposes. Most image matching systems have been built to support following options:

- Random browsing
- Search by example
- Search by sketch
- Search by text
- Navigation with customized image categories

Some systems are built to use more than one options simultaneously to get better results (Rui, Huang & Chang, 1999).

Some image matching systems require that the shape representation is invariant to translation, rotation and scaling. However, there are also systems which work without the shape representation being invariant. Features which can be searched from the image are for example: color, shape and texture. Color is one of the most widely used visual feature in image matching. Shape representations can be generally divided into two categories which are boundary-based and region based (Rui, Huang & Chang, 1999).

Finding two corresponding points between two images is the basis for finding features from images. The search of corresponding points can be divided into three main steps. Search begins by selecting interest points from distinctive locations in the image. These interest points can be for example: corners, blobs and T-junctions. The most valuable property of an interest point is its repeatability. The repeatability property measures the reliability of a detector. Reliability is important as finding the same physical interest points under different viewing conditions is an important task. After finding the interest points, the neighborhood of every interest point is represented by a feature vector. The last step is to match the descriptors between different images. Distance in between the feature vectors then defines the goodness of the matching (Bay et al., 2008).

Development of image matching has already been going on for decades. Development work has branched to develop image matching methods for various different kind of needs. One of the routes for image matching development has been the use of a set of local interest points. Initially the way how local interest points were used did pave its way for Lowe's work with SURF (Lowe, 2004).

Many different applications which use computer vision rely on matching keypoints across images. Nowadays the use of computer vision and vision algorithms has spread from the traditional computers with lots of computation power to smaller devices like smart phones and even embedded devices with low computation power and low memory. This change in the computation power has meant that also the computer vision algorithms must adapt to the new requirements. Algorithm developers are currently also concentrating on making the algorithms more computationally effective but still making sure the descriptors remain robust to scale, rotation and noise (Alahi, Ortiz & Vandergheynst, 2012).

This chapter continues by discussing how and which features are used in image matching. It also reviews different feature detection techniques and gives insight of their pros and cons. Feature detectors can be combination of different techniques and these techniques such as Harris corner detector are also reviewed in this chapter.

2.1 Content-based image matching using features

Content-based image matching system, which is also known as query by image content, was introduced commercially in the IBMs system called Query by Image Content (QBIC) at 1995 (Flickner et al., 1995). IBMs system was that time the most well-known system for image matching. Many other content-based image matching systems have been developed since then (Shrivastava, Tyagi, 2014). However, despite commercial product like IBMs QBIC are the most well-known, most of the available systems originated from academic backgrounds. These early academic products included systems like Candid, Photobook and Netra, and they used simple characteristics to describe the image content. Characteristics were color and texture characteristics (Müller et al., 2004, Chen, Wang, 2002).

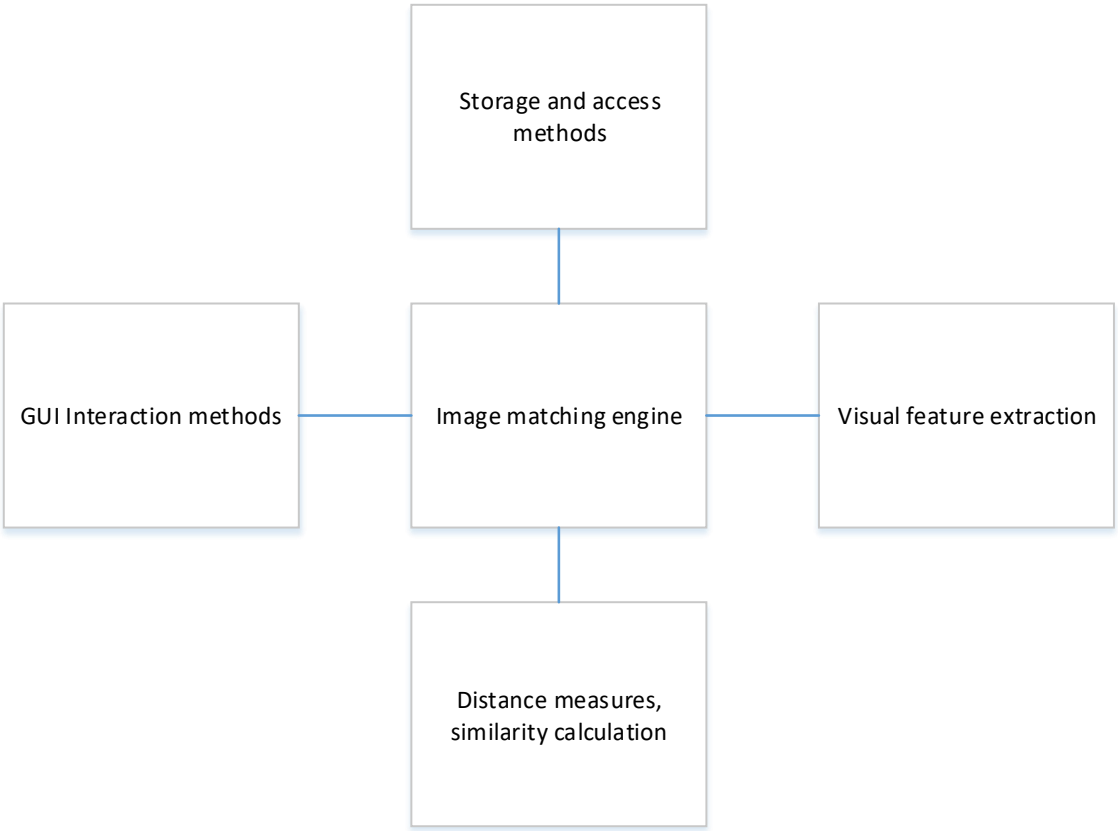

Figure 1. Main components for content-based image matching systems (Müller et al., 2004).

Most of content-based image matching systems have similar architecture as described in Figure 1. Their architecture includes tools for extracting the visual features of the images but also tools for browsing and archiving the images. The tools include efficient mechanisms to store and retrieve the features. In addition, content-based image matching systems usually have tools for distance measurements or similarity calculation and graphical user interface. Distance measurements are used to measure the distance between queried image and possible results. Similarity calculation is used to filter out images which might have some similarities but do not have enough to be one of the searched images (Müller et al., 2004). All these tools are used together to browse, search and navigate images from larger databases (Shrivastava, Tyagi, 2014).

2.1.1 Visual features used in content-based image matching

Content-based image matching methods use several different visual features which are classified into separate groups. The groups are primitive features, logical features and abstract features. The primitive features group consists of features like color or shape, while the logical features group consists of features like identity of objects shown. Lastly abstract features consist of features which are more abstract, like the scene's significance. From these groups, the most used group is the primitive features group which was the most used group in the beginning of 2000s (Shrivastava, Tyagi, 2014, Müller et al., 2004). During recent years the spatial location of objects has come into usage when querying and indexing images from databases (Shrivastava, Tyagi, 2014). Research on content-based image matching has earned massive interest due to its application in different multimedia systems. Content-based image retrieval could be used to bring large amounts of new information because of its capability of dealing with different kind of multimedia types such as text, video, images and audio (Montazer, Giveki, 2015).

2.1.2 Image matching using scale-invariant feature transforms

Around the early 2000s new insights and methods triggered a new research trend in content-based image matching. The scale-invariant feature transform (SIFT) technique has been one of the techniques that has managed to pave its way to image matching. The name comes from how the technique uses image data. SIFT transforms image data into scale-invariant coordinates relative to local features (Lowe, 2004). One of its advantage is its effectiveness on large-scale multimedia databases (Wengang Zhou, Houqiang Li & Qi Tian, 2017). The SIFT is a feature detector technique developed by Lowe in 2004. It was originally designed for the needs of computer vision and robotics where fast and stable image matching is very important. Image matching and feature detection have been important problems in machine vision and robotics, especially because machine vision and robotics applications are growing in various fields. The problem with machine vision is especially arduous, since the feature detection should be ideally able to detect features even in situations where the image is transformed. The transformations could be for example: rotation, scaling, illumination, noise addition and skewing. Also, another

problem lies with the features that are under detection. Ideally these features should be substantially distinctive in order for the feature detection to work with high accuracy (Karami, Prasad & Shehata, 2017).

The SIFT technique is very efficient in object recognition but for robust object detection it also requires large computational complexity. This is a major drawback, because the results cannot be calculated real-time making it unsuitable for applications that need real-time results. There has been several variants and extensions developed for SIFT which have speeded up its computation. However, these variants have exchanged the complexity for less accuracy (Karami, Prasad & Shehata, 2017).

The SIFT technique is capable on finding matching features even in situations where the image has been rotated or the viewpoint has been changed. It is also capable of solving affine transformations in matching features meaning that the technique can find matching lines which are oriented same way. The SIFT algorithm has four basic steps for finding matching features as described originally by David Lowe (Montazer, Giveki, 2015):

- Estimate scale space extrema
- Localize key points
- Orientate localized key points
- Compute local image descriptors

The first phase for the SIFT algorithm is to identify potential interest points. This is done by searching over all scales and image locations. Computation is accomplished by estimating scale space extrema using the *difference of Gaussian* (DoG) method (Karami, Prasad & Shehata, 2017, Montazer, Giveki, 2015). DoG is a filter that makes the original image in spatial domain smoother. This is by doing convolution operations on the original grey image (Shoujia et al., 2012). Gaussian transfer functions are used to map data points directly to optical properties typically making a lookup table (Kniss et al., 2003). DoG filter uses two Gaussian differences with different widths as its transfer functions (Shoujia et al., 2012). In the first phase, interest key points are checked and all that have been found are used in the second phase (Montazer, Giveki, 2015). Keypoints are localized by first localizing candidates for the keypoint and by eliminating the low contrast points. After eliminating the low contrast points the leftover points are orientated. One or more orientations are done by assigning them to each keypoint location based on the local image gradient and its directions. Lastly, local image descriptors are computed for each

keypoint. It is done by measuring local image gradients at the selected scale in the region around each keypoint. Local image gradients are transformed to representation that allows significant levels of local shape distortion and change in illumination (Lowe, 2004, Karami, Prasad & Shehata, 2017). Keypoint descriptor of each keypoint is a 128-dimensional vector. Since each point contains lots of information about the meaning of the point, the keypoint descriptor has to encode this information (Montazer, Giveki, 2015).

One of the more important ways for SIFT technique to be efficient, is the way how it works. The SIFT technique extracts a large number of features from the image. Features cover the whole range of scales and locations of the image. The SIFT technique can find typically around 2000 stable features from a small image with 500x500 pixels. The number of features depends naturally on the image content and quality. Bigger images might contain a large number of features for the technique to detect. For the feature detection, the choice of parameters affects the number of features found. For object detection the number of features found is especially important. This amount becomes important because detecting smaller images from the cluttered background requires that at least three features are correctly matched from each object for reliable identification (Lowe, 2004).

With the SIFT technique the image matching and recognition are done by first extracting SIFT features from a set of reference images, and then storing them in a database. The new images features are first extracted and compared individually to the features in the database. Concurrently the method tries to find candidate matching features based on the Euclidean distance of their feature vectors. For each keypoint, the best candidate match can be found by comparing the keypoint and its nearest neighbor from the database (Lowe, 2004).

The reason for the time-consuming extraction is that the features are highly distinctive. The database of the found features is typically huge, so to find a high probability match for a single feature the feature needs to be highly distinctive. Some of the images might be very noisy which makes some of the found features being incorrectly matched. This incorrectness is because the noise in background could match with the searched features. The incorrect matches can be filtered out from the full set of, matches by improving the accuracy of identifying correct matches. Correct matches can be identified by comparing

location, orientation and scale of the features with the new image. Raising the accuracy of feature matching using this method has been found to be effective enough, because the probability that multiple features are matched with these parameters is lower than the probability that the features are matched. Finding incorrect features from the set of features with this method is not too time consuming, because the SIFT technique uses efficient hash table implementation of the generalized Hough transform to determine invalid features (Lowe, 2004).

Features found from the image are then combined into clusters for further use. The feature clusters are used to recognize objects from the image. This is done by first finding collections with three or more features. The identified feature collections are associated with objects. Inspection is done by first doing an estimation of a least-squared affine approximation of the object's presentation to find different shapes (Lowe, 2004). Least-squares regression is used to find the best matching fit for the affine transformation. Least-squares method works by minimizing the sum of the squares of the offsets of the points from the curve (Hutcheson, 2011). Affine transformation is transformation between two vectors that preserves collinearity (Weisstein, 2004). When the presentation is found, the images features are looked through for consistencies with the object's presentation, and at the same time feature deviations are removed. Finally, a detailed computation is used to create the probability that the set of features indicates the presence of an object (Lowe, 2004).

Images are usually acquired from a variety of sources and these sources may hold large amounts of images. The variety of images and the scale of features have brought out another technical challenge to computing systems. The challenge is to index and manage all of the data effectively and to make it easily available. Therefore, there has been attempts to make the SIFT technique even more effective and robust. One of the methods is based on the SIFT technique as it has been already used successfully in many computer vision related tasks. There are two important drawbacks in the SIFT technique, time complexity and memory usage. A solution for these problems is to decrease the dimensionally of the features extracted by the SIFT but at the same time retaining the most descriptive image features. The dimensionally reduction can be done by a new kind of clustering method (Montazer, Giveki, 2015).

Montazer and Giveki, have found out that when using SIFT for content-based image retrieval, the amount of keypoints extracted from the image can be very big. The huge number of keypoints results in large database. Matching images with large databases can be very slow and technically challenging, which means that this kind of use of the database is impractical. An idea for decreasing the size of the SIFT descriptor is to do data clustering. The size of SIFT descriptor is decreased by dividing the keypoint descriptor to smaller feature vectors. (Montazer, Giveki, 2015).

2.1.3 Image matching using speeded up robust features

Using a set of local interest points can be traced back to the year of 1981 and work around visual obstacle avoidance in the development of a space rover. Space rover used several kinds of vision systems to locate objects and used a corner detector to identify features in those objects (Moravec, 1981). This work has been further developed by making it more repeatable under small image variations and near edges (Lowe, 2004). Later on, the use of local interest points was further developed and it was shown to be efficient in motion tracking. Using local interest points made also recovering 3D structures from motion more effective. (Harris, 1993). Harris's work with the corner detection has become widely accepted and applied for other image matching tasks (Lowe, 2004).

The speeded up robust features (SURF) technique approximates the SIFT technique. The SIFT and the SURF techniques both are based on the concepts of *descriptor* and the *detector*. Descriptor provides unique and robust description of each feature in the image and detector is used to detect these features. The SURF technique does not reduce the quality of the detected points, even though it performs faster than the SIFT technique. One of the reasons which makes it faster is the use of integral images for example, the Figure 2. These integral images make it possible to reduce the amount of operations for simple box convolutions. For example, box in Figure 2 takes only three additions and four memory accesses to calculate the sum of intensities inside a rectangular region. Operations can be reduced independently of the chosen scale which makes it good technique to be used for example, in video stabilization. SURF is a robust image interest point detector and descriptor scheme. (Pinto, Anurenjan, 2011, Bay et al., 2008).

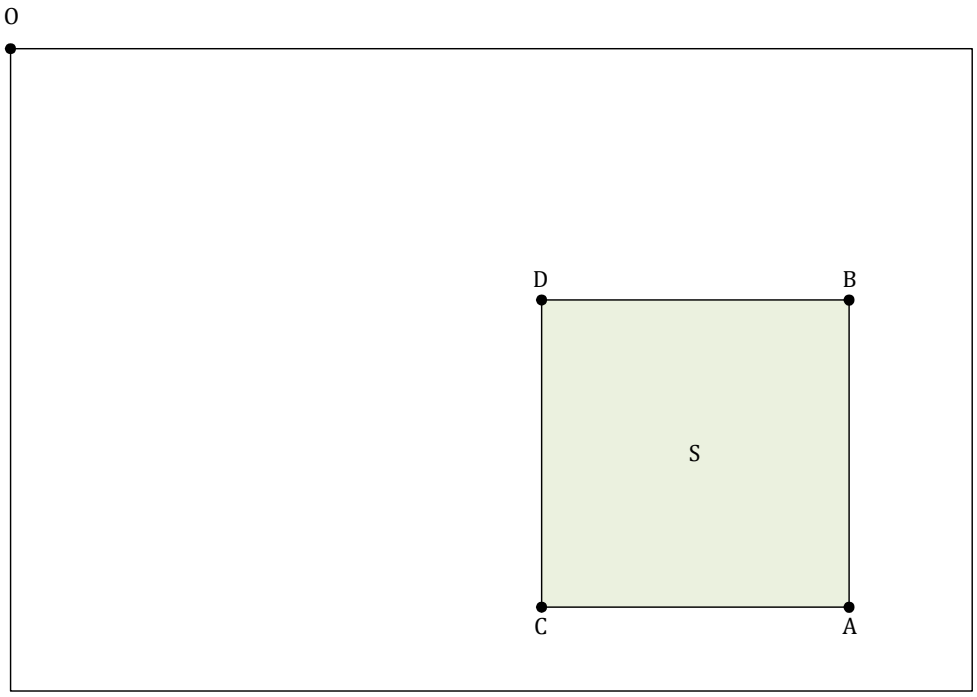

Figure 2. Integral image provides effective way of calculating the sum of all pixels in the input image (Bay et al., 2008)

The SURF detector is based on Hessian matrix which has been selected because of its good accuracy. The Hessian matrix relies on the determinant of Hessian for scale selection. Matrix is defined as

$$H(x,\sigma) = \begin{pmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{pmatrix}$$

where $L_{xx}(x,\sigma)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ with the image I at point x, and similarly for $L_{xy}(x,\sigma)$ and $L_{yy}(x,\sigma)$. The SURF technique obtains invariance to image rotation with a two-step approach. First, SURF uses the Haar wavelet to compute a reproducible orientation of the interest points (Pinto, Anurenjan, 2011). Haar wavelet is represented by a 2x2 matrix (Stanković, Falkowski, 2003).

$$H_s(1) = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

After computing orientation, a square region aligned to the selected orientation is constructed. Finally, the SURF descriptor is extracted from that square region (Pinto, Anurenjan, 2011). The methods used in SURF are very simple as they only include the

essential operations needed for the feature matching and the methods also rely on using the strengths of existing detectors and descriptors (Bay et al., 2008).

2.1.4 Image matching using binary robust independent elementary features

The technique of using binary robust independent elementary features (BRIEF) is another alternative for SIFT. Similarly, to the SURF technique also BRIEF requires less computation than SIFT. BRIEF determines a binary descriptor but it does not include own method for keypoint detection, which means that the method has to be implemented as a part of some other technique. Usually BRIEF descriptor is used with Features from Accelerated Segment Test (FAST) which implements keypoint detection (Figat, Kornuta & Kasprzak, 2014).

BRIEF works by having a smoothed area in image for which the technique runs simple binary tests between pixels. Its performance is quite close to SIFT in many areas including robustness, lighting and blur. One of its weak sides is that it is very sensitive to in-plane rotations (Rublee et al., 2011). The descriptor is also sensitive to noise. The sensitivity comes because the descriptor disregards the neighboring pixels in favor of point intensity. Typically, descriptor sensitivity is reduced by smoothing the image with a Gaussian filter. BRIEF also does not have a constant sampling pattern, which means that to build a descriptor, the pairs of pixels are randomly selected (Figat, Kornuta & Kasprzak, 2014).

The techniques using binary-valued features are becoming more popular due to several advantages over techniques using vectors like the SIFT and SURF techniques. Reasons for this are that binary-valued features can be faster to compute; they are more compact to store and are more efficient to compare. However, using binary-valued features is not a silver bullet as it does also have some drawbacks. The technique can become slow when using linear search on large datasets. Also, image matching algorithms used in techniques using vector-based features cannot be used in techniques like BRIEF which uses binary-valued features. For example, one of the algorithms which is typically used with SIFT and SURF is the approximate nearest-neighbor search but since it cannot be used with binary-valued features, another algorithm needs to be used (Muja, Lowe, 2012).

BRIEF uses the Hamming distance for comparing binary features. Hamming distance is calculated by performing a bitwise XOR operation followed by a bit count on the result.

Doing the calculation like this has its advantages as it involves only bit manipulation operations and these operations can be performed quickly. The x86_64 processor architecture has the POPCNT instruction which is used for the bit manipulation operation and modern computers have hardware support for this instruction (Muja, Lowe, 2012).

Even though BRIEF is faster to compute because of the use of Hamming distance it is only practical to use with smaller datasets when using linear search. BRIEF keypoint descriptor can get very big depending on the number of analyzed pairs of pixels of the analyzed patch. Usually, descriptor contains 128, 256 or 512 bits. Descriptor size grows depending on the pixels on the analyzed patch which means that the size affects the speed rate and discriminative power of the descriptor (Figat, Kornuta & Kasprzak, 2014). Calculating the Hamming distance between pair of binary features is fast which means that BRIEF is practical with smaller datasets but it can become slow when using linear search with large datasets. Usually this bottleneck is solved by replacing the linear search with an approximate matching algorithm. The approximate matching algorithm can find the approximated nearest neighbors with less computing costs and therefore can also be used for larger datasets. Using approximated neighbors means that they are not necessarily the exact neighbors but are still close enough to each other's (Muja, Lowe, 2012).

Algorithms created for vector-based features are not usually compatible with binary-valued features and since, the algorithms used with vector-based features usually perform a hierarchical decomposition of the search space. With binary-valued features the matching is usually done by assuming that the features exist in a vector space where each dimension of the feature can be continuously averaged. For example, with algorithms like the *kd-tree* algorithm, the *vocabulary* tree and the hierarchical *k-means* tree (Muja, Lowe, 2012).

For binary-valued features, the algorithms used for approximate nearest neighbor search are usually based on various hashing techniques. These include techniques like the locality sensitive hashing, semantic hashing or min-hash (Muja, Lowe, 2012). In locality sensitive hashing, multiple hash tables are used to store the pixels and the hash tables are then hashed in different buckets. The nearest neighbor is then retrieved from buckets by first fetching all buckets which match the query descriptor. The bucket elements are finally compared using a brute force matching. The idea behind local sensitive hashing is

that the nearest neighbor is found with high probability depending that there are enough hash tables which can be used. For binary-valued features, local sensitive hashing changes a little bit since the hash function is a subset of the signature bits. The buckets in the hash tables contain descriptors with a common sub-signature (Rublee et al., 2011).

2.1.5 Image matching using other binary descriptors

Another efficient alternative for SIFT and SURF is a technique called oriented FAST and rotated SUFR (ORB) technique. ORB uses binary-valued features like the BRIEF technique (Muja, Lowe, 2012). It extends BRIEF by adding rotation and scaling invariance to the feature detector. In ORB, FAST is used to find keypoints. ORB uses machine learning method for learning the optimal set of sampling pairs, and in addition to using machine learning, it achieves scale robustness by building an image pyramid. ORB obtains rotation invariance by using moments. Moments are computed in a circular-shaped patch around the center of the mass of the patch. As ORB uses machine learning the sampling pairs should be defined properly to get the best results. Sampling pairs should have two properties; they should be uncorrelated and to make the feature more discriminative, the chosen set of pairs should have maximal variance. Greedy search is run among all sampling pairs following a predefined binary test for ORB to fulfil the needs for the properties (Figat, Kornuta & Kasprzak, 2014).

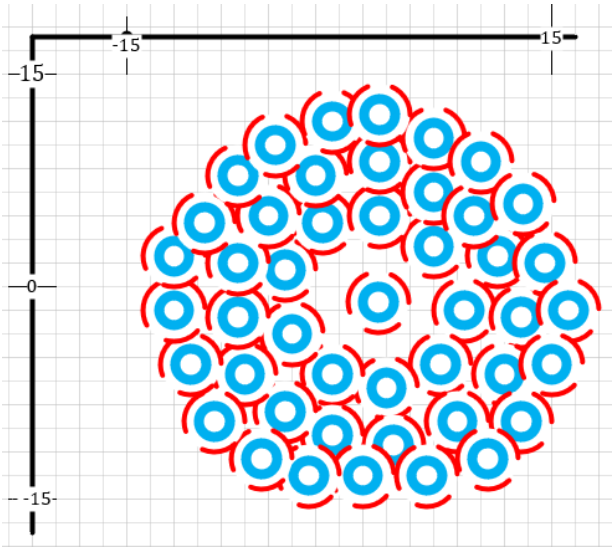

Figure 3. BRISK sampling pattern

The Binary Robust Invariant Scalable Keypoints (BRISK) is another binary-valued descriptor. It uses a hand-crafted sampling pattern which is composed out of concentric rings with more points on outer rings. Usually the sampling pattern has circles to represent the location of the sampling like in Figure 3. From the possible sampling patterns usually two types of patterns are distinguished. These sampling-point patterns are short-distance and long-distance ones (Figat, Kornuta & Kasprzak, 2014).

Fast Retina Keypoint (FREAK) descriptor is similar to BRISK, in that it uses an encoded sampling pattern. However, the sampling pattern differs from the one used in BRISK; the pattern uses overlapping concentric circles and circles have more points on inner the rings, see Figure 4. The areas are named similarly as retina areas, where the outer ring is called Peri, the second outer is called Para and the most inner one is called Fovea. FREAK samples pairs with symmetric sensitive fields with respect to a patch center so that it can achieve the rotation invariance. FREAK also learns the optimal set of sampling pairs utilizing machine learning like ORB. First, it creates a set of pairs for mimicking the *saccadic* search. Saccadic search stands for scanning of the scene by moving both eyes simultaneously. Usually the eye is not completely static but moves a little bit around (Alahi, Ortiz & Vandergheynst, 2012). Machine learning is used to select subset of sampling pairs possessing the most discriminative power (Figat, Kornuta & Kasprzak, 2014).

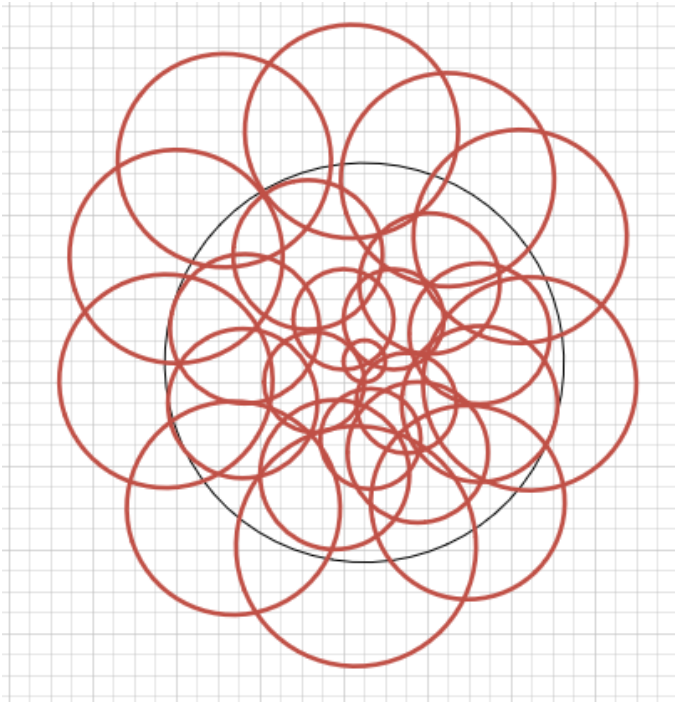

Figure 4. FREAK sampling pattern (Figat, Kornuta & Kasprzak, 2014).

Researchers constantly evaluate the performance of descriptors in order to find better training sets and algorithms. One of the well-known datasets for testing was introduced by Mikolajczyk and Shmid. Usually, the performance of a descriptor is related to the combination of detector and descriptor (Alahi, Ortiz & Vandergheynst, 2012).

2.1.6 Harris corner detector

Detecting different features from images is an important task in image matching and in Computer Vision. Corners are commonly favored due to their two-dimensional constraint. Corner constraints are usually fast for algorithms to detect and the detection is also more accurate (Mair et al., 2010).

The Harris corner detector is a widely used technique in many different image matching tasks. It is very sensitive to complicated changes in image, like the image scaling but it is still stable for less complicated changes like lighting changes or rotations of the image. Harris corner detector is especially effective in combination with detectors like the Scale Invariant Feature Transform detector. The effectiveness is because SIFT

detector is stable to scale changes, but on the other hand is not as effective in finding corners. Finding corners is an important feature in image matching as it helps finding some local structures. The Harris corner detector helps in these kind of situations as it combines with the SIFT detectors abilities (A. Amaricai, C. Gavriliu & O. Boncalo, 2014, Zeng, Liu & Li, 2010, Derpanis, 2004).

Due to its features, the Harris corner detection algorithm is being estimated as one of the most accurate corner detection method. The algorithm operates satisfactory even when matching images that contain lots of noise. However, the effectiveness also comes with a drawback, as the algorithm has high computation requirements. The computation requirements come from the use of five sub-operations which are required by the algorithm. The sub-operations are:

- Gradient computation
- Gaussian smoothing
- Harris measure computation
- Thresholding
- Non-maximum suppression

These operations are executed sequentially and each of them outputs an image which is then input to the following sub-operation.

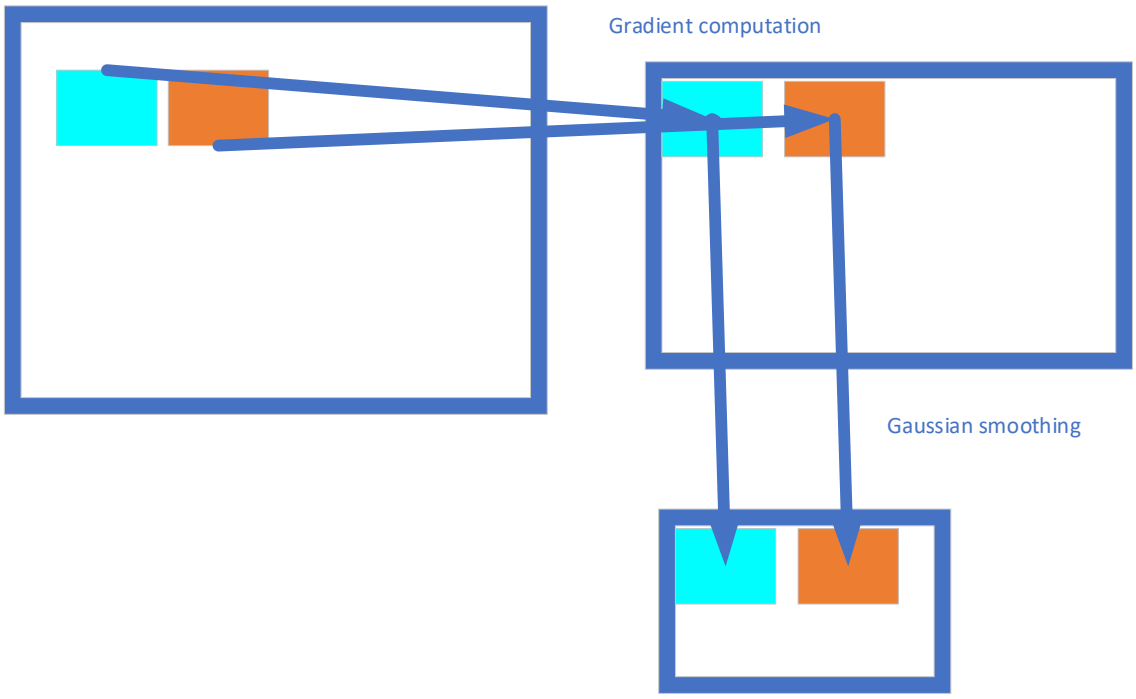

Figure 5. First two sub-operations of the Harris corner detection algorithm (A. Amaricai, C. Gavriliu & O. Boncalo, 2014).

Gradient computation estimates the derivates for each pixel in both X and Y axel. The result of the two gradients along the X and Y axes represent changes in the image. Gradient computation requires the processing on the nearest pixels around the pixel that is being investigated. This requires that the algorithm stores temporarily three gradient images after the gradient computation. These three images are patch of image that is being investigated and the X and Y derivatives (A. Amaricai, C. Gavriliu & O. Boncalo, 2014).

The final result from this computation is used in the next step by applying Gaussian smoothing into the three gradient images which were obtained from the gradient computation. The Gaussian smoothing process consists of a convolution between a Gaussian kernel and the corresponding pixel neighborhood in the three gradient images. The result from the Gaussian smoothing is three Gaussian values from which Harris measure can be taken (A. Amaricai, C. Gavriliu & O. Boncalo, 2014).

Harris measure is defined as

$$S = \begin{pmatrix} I_x^2 & I_{xy} \\ I_{xy} & I_y^2 \end{pmatrix}$$

The eigenvalues of this matrix indicate whether a pixel is a part of a flat region, edge, or a corner. In this step, eigenvalue computation could also be used but it is much more computationally intensive so Harris measure is used instead (A. Amaricai, C. Gavriliu & O. Boncalo, 2014).

Thresholding means finding corner pixels among the pixels which are associated to flat and edge regions. Thresholding is done by characterizing corner pixels by a large positive value of the Harris measure. This means that pixels associated to flat and edge regions can be found by applying a corresponding threshold value.

In non-maximum suppression all other than corner pixels are eliminated. Elimination is done by characterizing corner pixels by the largest Harris measure within a corner region (A. Amaricai, C. Gavriliu & O. Boncalo, 2014).

Similarly, as other image processing techniques and helping methods also Harris corner detector has different kind of approaches. Some of the approaches rely on specially researched integrated circuits which are designed to do some of the needed steps in Harris corner detection faster (A. Amaricai, C. Gavriliu & O. Boncalo, 2014). Due to the importance and popularity of corner detection many different corner extraction methods have been implemented. Most of these either use similar steps as in Harris corner detection but with some extra adjustments or use some different technique. Alterations can be small like interpreting the eigenvalues with or without a threshold. The global tracker SIFT uses difference of Gaussian, but in other approaches SURF is used because it is much faster. SURF is used with Haar wavelet approximation of the determinant of the Hessian. Both of these approaches are very computationally expensive. Being computationally expensive is a serious drawback in situations where the algorithm is used as a global matching algorithm. Matching globally means that the algorithm is used within the whole image. The corner detector needs then to provide a high repeatability also after large affine transformation. (Mair et al., 2010).

One of the other corner detection methods is called FAST. FAST is based on the SUSAN (Smallest Univalue Segment Assimilating Nucleus) corner detector. Nucleus in corner detection is the circular mask over the pixel to be tested. FAST differs from other corner detection methods by evaluating a smaller area of the circle. It looks for the pixels and evaluates only those pixels which are on the discretized circle describing the segment. Since FAST is based on the SUSAN corner detector it also uses some of the

same techniques as SUSAN. One of these techniques is the use of a test mask. In FAST technique a Bresenham's circle of diameter 3.4 pixels is used as test mask. The use of Bresenham's circle of this diameter means that the number of pixels which need to be compared becomes very high. For the value of the nucleus at least 16 pixels need to be compared to get full accelerated segment test. This testing might become too extensive, so in FAST the corner criterion has been made more relaxed (Mair et al., 2010).

Even though FAST has a more relaxed corner criterion and the pixel evaluation is done to a smaller area, there still remains the question on which pixel should be compared first. The obvious order is to compare first pixel, then second, then third and so forth. However, selecting the comparing order of the pixels also affects the speed of evaluation. Machine learning has been used to find the best decision tree which then helps to find the environment where FAST is applied. The effectiveness of this method depends greatly on the training data which is supplied to the machine learning algorithm. Using machine learning also means that FAST is not bullet proof so it might miss some of the pixel configurations if the training data of the machine learning algorithm was not good enough. Small changes like rotation of the camera could already produce pixel configurations which are not present in the training data. There is also the possibility that the machine learning training data includes all the pixel configurations but a small rotation about the optical axis could cause disturbance on the detection. These changes could result in incorrect or slow corner detection (Mair et al., 2010).

2.2 Content-based image matching using regions

For more than two decades, research of image matching has concentrated on region-based analysis and analyzing the whole image. During this time the research direction has also moved between using handcrafted features and machine-learned features. Region-based analysis has its own strengths and it has been proven to be especially powerful for images with rigid structures like forms and business letters. In contrast to that, region-based image matching is not limited to rigid structures since some of other kind of images can be fitted into the geometric configuration which suits for region-based image matching better. These geometric configurations are for example, documents which have very

distinct visual style. In these cases, the documents are written in a standard format which makes them recognizable even when the texts in the documents is unreadable (Harley, Ufkes & Derpanis, 2015).

One very similar approach for region-based analysis is to analyze the image as a whole. In high-level this means that the region is the whole image and image analysis can be done holistically or with the same approach as in the region-based analysis. When using region-based analysis the image regions are concatenated to create a more holistic view, this allows the use of a descriptor which contains both global and local layout characteristics (Harley, Ufkes & Derpanis, 2015).

The content-based image matching is more about analyzing the image as a whole. Methods for this type of analysis include for example, selective region matching and utilizing neural networks. Both of these methods are reviewed in this chapter as well as how content-based image matching is used for analyzing text documents.

2.2.1 Selective region matching

Content-Based Image Retrieval (CBIR) or content-based image matching has been under active research more than a decade. Like the name says, it makes use of the images visual content in order to be able to identify relevant content. During this time, several different techniques have been developed for CBIR (Shrivastava, Tyagi, 2014, Wengang Zhou, Houqiang Li & Qi Tian, 2017). However, also the region-based analysis has been proved to be powerful on image matching (Harley, Ufkes & Derpanis, 2015). One of these techniques is using regions to help finding the relevant images as suggested in the work by Shrivastava and Tyagi (Shrivastava, Tyagi, 2014).

The queried image usually consists of two regions. These regions are called relevant and irrelevant. Defining regions of interest (ROI) of images helps to define irrelevant regions. This step is important in the region-based technique as having irrelevant regions in the image makes the image matching more effective as the researched area of image becomes smaller. Regions of interests can be defined automatically by the system itself but also by the user. According to this the region-based technique can be categorized into two classes: the techniques using system designated- and user designated ROIs. From these two the

latter one is usually more interesting and more promising approach. The reason for this is that user designated ROI enables the user to express his needs when the query is formulated (Shrivastava, Tyagi, 2014).

The disadvantage for enabling user selection of ROI is that users can select different sizes for the ROIs and therefore make the query less accurate. Users can also select one or many ROIs making the query even more complicated. Calculations for several different sized and located regions of interest requires complex algorithms and therefore this approach results in increased computation times. One way to overcome this problem is to use a method which is based on region codes. Region codes method supports one concept which reduces the time needed for computations. By supporting multiple ROIs and their relative locations (Shrivastava, Tyagi, 2014).

When using the region codes, the image is first divided into suitable blocks of a fixed size. Then each of the blocks is assigned a 4-bit code. The 4-bit code depends on the spatial location of the block. Figure 6 shows an example on how a picture is divided and the blocks are assigned with 4-bit code when using layout size of 3 x 3 (Shrivastava, Tyagi, 2014).

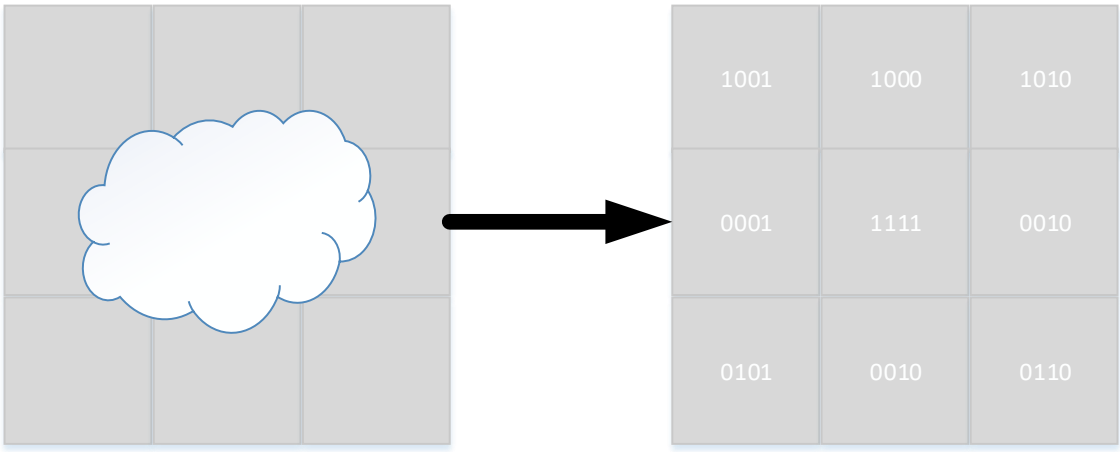

Figure 6. Region codes for a 3x3 layout of an image (Shrivastava, Tyagi, 2014).

The region codes and the number of bits are chosen based on the layout size. For example, Figure 7 shows a 5x5 sized grid with 8-bit region codes. Region code based selective region matching can be used also on layouts with higher dimensions, for example on 7x7 grid. Bits are placed to the grid so that the region code specifies the regions place in left,

right, bottom and top region of the image respectively. Images usually have the most important part of it in the middle of the image and therefore it is usually assigned with the unique region code. For example, in Figure 7 the central region has code 11 11 11 11. This type of use for the region codes works best on the layouts with uneven sizes as then the unique region code can be put in the middle of the image. The defined region codes can be further used with the irrelevant regions to filter them out (Shrivastava, Tyagi, 2014).

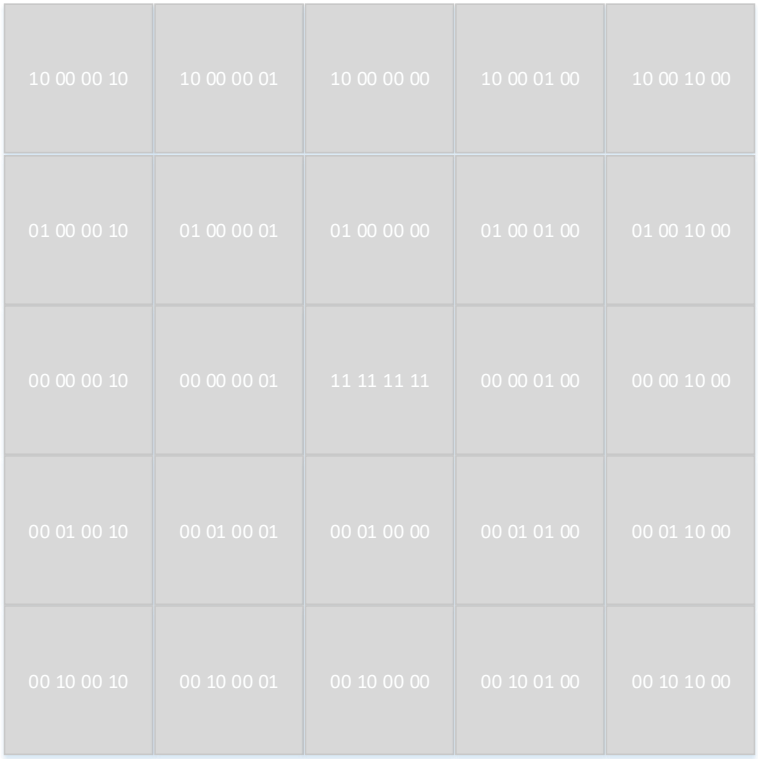

Figure 7. Example of region codes for a layout size 5 x 5 (Shrivastava, Tyagi, 2014).

The ROI-based image retrieval technique must be able to query images of all sizes and shapes for the image retrieval method to be workable. The region-based approach supports both varying size of the ROI and also multiple ROIs. The approach works by measuring the similarity in the blocks between the two images. Blocks are chosen from blocks which are overlapping with the ROI and choosing blocks from the target image which have similar region codes. Looking for the 1 in the similar bit positions is used to

determine similarity between the different region codes as shown in the Figure 8 and Figure 9.

```
  1010
  1001
------
  1000
```

Figure 8. Example of region codes with high similarity (Shrivastava, Tyagi, 2014).

```
  1010
  0110
------
  0010
```

Figure 9. Example of region codes with low similarity (Shrivastava, Tyagi, 2014).

To measure the similarity between two images first a list of blocks needs to be defined. These blocks need to correspond to the ROIs of the queried images. This list is then used to scan the target image. The target image is scanned as many times as the count of the blocks in the block list is. Last, the images are output and ordered based on their similarity score (Shrivastava, Tyagi, 2014).

2.2.2 Image matching using convolutional neural networks

Convolutional Neural Networks (CNN) are a class of neural network models. CNNs are usually used for analyzing images due to their features. One of their features is that their capacity can be controlled by varying their depth and breadth. Another feature of CNN is that they make strong and mostly correct assumptions about the nature of images (Krizhevsky, Sutskever & Hinton, 2012). CNNs are hierarchical neural networks consisting of different layers. The layers are convolutional and subsampling layers. Subsampling layers include layers for example, for classification, input and output based on to which application they are adapted (Ciresan et al., 2011). CNNs are not usually as good as the standard feedforward neural networks for general purpose classification but they require less connections and parameters which makes them easier to train and their performance is still quite close to that of the feedforward neural networks when classifying images. However, also CNNs have their drawbacks as even though being very

efficient with their local architecture, they have been unreasonable expensive to use in some cases. One of these cases is applying CNN in large scale high-resolution images. This inefficiency can be mitigated with more powerful computing hardware; with current graphical processing units (GPU) it is possible to facilitate the training of fairly large CNNs (Krizhevsky, Sutskever & Hinton, 2012).

CNNs are useable for fine-grained object recognition which makes them especially attractive for document image analysis. Usually it is best to pre-train CNN on a larger related problem before training it with the data of interest. This pre-training is done to avoid overfitting. Also in problems where region-specifig information is available, some things should be thought before using CNN; it might be potentially better to encode the region-specific information data in multiple networks which are trained on the different ROIs than train a single network on the entire image (Harley, Ufkes & Derpanis, 2015).

CNNs can be also used for image retrieval with proper training set. After a CNN has been trained on the classification, the neural network layers can be used for outputting image features at different level. Outputting at different levels can be done by forming a chain of abstractions. In this chain the lowest layers contain simple local features and the highest layers contain global features with concise and descriptive representations. The layers of the network can then be used for different tasks, then, for image retrieval by extracting output from the top of the CNN and by using the output as a feature vector (Harley, Ufkes & Derpanis, 2015).

Training of the CNN affects very much the end results of the image retrieval. Therefore there have been different image datasets for training CNNs. These databases vary greatly in their sizes. Their sizes range from around ten thousand images up to 15 million images. Simpler object recognition tasks can be done by training a CNN with smaller datasets especially some with special augmentations like label-preserving transformation. Usual object recognition tasks regocnise objects from more realistic sources, which means that the objects to be identified can vary very much. This means that larger datasets are commonly needed to be able to train the NN to recognise objects correctly. Larger image datasets have become available only in the last decade.. Newer image datasets have also been labeled which makes them even more suitable for supervised training (Krizhevsky, Sutskever & Hinton, 2012).

Currently, one of the bigger image datasets is ImageNet. The creators of ImageNet aimed

to have about 50 million cleanly labeled full resolution images in it by its completion. ImageNet is a hierarchical database which uses the structure of WordNet. Images are divided into so-called synsets, with about 500-1000 images grouped into each synset. For example, “sailboat” would have images of different sailboats but more generic synset like “vehicle” would have images of bikes, busses, tractors etc. A synonym set or synset is a meaningful concept, which is possibly described by multiple words or word phrases. This concept is used in WordNet and there is around 80000 noun synsets in WordNet. Quality-controlling and human-annotation is applied for the ImageNet data. The result of this is that ImageNet offers tens of millions of cleanly classified and labeled images making it a very good candidate for supervised neural network training (Deng et al., 2009).

ImageNet is a hierarchical database which means that the images are organized into different classes in a densely populated semantic hierarchy. Synsets of images are interlinked by several types of relations. The most important of these relations is the is-a-relation as it is the most comprehensive and useful. ImageNet is built so that the images would vary much between themselves, resulting in better neural network training. The variance in images means that the objects have different appearances, positions, view points, poses and the background clutter differs between images (Deng et al., 2009).

2.3 Text-based image retrieval

Early research on image retrieval was based on textual annotation of the images. In textual annotation images are provided with keywords which describe the image content and search engines use the keywords to find images of interest. Text-based search approaches had some drawbacks and some of them were major. One of these was that the images were annotated manually which meant extensive manual work. The amount of work also increased rapidly when the collections of images grew. First of all, the annotations were done manually with human interaction which meant that work was highly dependent on the perception of the people doing this work. This meant that some images might have missed a number of relevant keywords since different people have a subjective perception for the images which result in different description for the content of the image. This may lead to partially wrong image retrieval due to the forgotten and ignored information and

a wanted image may not be found. There might also be cases where the human classifier has thought that the information perceived is not important and he or she did not add it as a keyword for the image. As all annotation work is done manually by different persons, there is also the possibility that the keywords are picked by certain language and certain style which makes it hard for the other users to get the desired results when retrieving images. Getting desired results becomes especially difficult if the person using the retrieval system does not have any background or knowledge for the languages used in the annotations (Halawani et al., 2006).

Extracting texts from images is a challenging problem in many practical applications. Text recognition has similar challenges as object recognition, and these problems make text recognition complicated. Text can appear with wide range of different backgrounds, textures, fonts and lightning conditions. Due to the complicated recognition, many text detection and recognition systems rely on hand-engineered features to represent the data (Wang et al., 2012).

2.3.1 Convolutional neural networks for structured document retrieval

In structured documents the important information of the genre can usually be found from the reflection of layout of text and graphics. This reflection means that the documents of the same category often share region-specific features. Convolutional neural networks take a matrix of pixels as input. The matrix is then processed through a stack of convolutional layers. The output of the convolutional layers is classified using two or three fully-connected layers. In CNNs the fully-connected layers are not usually invariant to geometric transformations in the input (Harley, Ufkes & Derpanis, 2015).

2.3.2 Text matching with image recognition

One of the ways to utilize content-based image matching is to find text from images. However, matching two text can be an exhaustive problem in many natural language processing tasks. The challenge is that the searchable text might be a part of one or many

words, phrases or sentences. One way to cope with this problem is to extract meaningful matching patterns so that the matching score can be evaluated (Pang et al., 2016).

There has been substantial success in the development of image recognition techniques especially with the use of CNNs. The same idea can be used to match texts by creating a matrix which can be viewed as an image. This matrix will represent the similarities between words. The similarity matrix can then be fed into the CNN to capture rich matching patterns in a layer-by-layer way. When matching texts, the rich interaction structure must also be taken into account. Choosing correct interaction structure will affect the end result greatly. There are three different interaction structures:

- Word level matching signals
- Phrase level matching signals
- Sentence level matching signals

Word level matching signals are matchings between words in two different texts. Matching also includes finding similarity matches for the words instead of the strict identities. Identical matching could mean matching just words, such as "boat-boat", "train-train", "sheep-sheep" and "dumpling-dumplings", however by matching also similar words means that also the following patterns could be found: "famous-popular" and "Chinese-china" making the word matching more usable in some cases (Pang et al., 2016).

Phrase level matching signals, are matchings between phrases. Matching also includes *n*-gram and n-term matching. n-gram matching means that text can be matched when *n* amount of words are matched. For example, "(train heads to hills) - (train heads to hills)". n-term matching then allows alternatives for order or semantic in the text, for example "(noodles and dumplings) – (dumplings and noodles)" and "(we are famous Chinese food) – (we are popular in china)" (Pang et al., 2016).

Sentence level matching signals, are the matchings between sentences. These sentences can be composed of multiple lower level matching signals, for example having three words matched in a phrase level matching as mentioned above. When matching between paragraphs that contain multiple sentences, the approach is usually to view the paragraph as a long sentence so that it generates paragraph level matching signals (Pang et al., 2016).

2.3.3 Wordnet in image matching

Wordnet was created to provide a machine-readable dictionary for varying purposes. Dictionaries have been created before but earlier dictionaries were done for human readers. Human readers were kept in mind throughout evolution of dictionaries which means that they were very convenient for human readers but not that convenient for machines. Dictionaries come handy, since meaningful sentences are composed of meaningful words. Dictionaries hold information and meaning of those meaningful words, which means that any system hoping to process natural languages needs to be able to utilize dictionaries. Wordnet provides a more efficient dictionary for machine use, by providing a combination of traditional lexicographic information which can be used in modern computing since it is provided online. Wordnet has English adverbs, nouns, adjectives and verbs organized into sets of synonyms, each representing a lexicalized concept. These synonym sets are linked by semantic relations to determine word definitions (Miller, 1995).

The vocabulary of a language is usually defined as a set W of pairs (f, s) where a form f is a string over a finite alphabet. Sense s is an element from a given set of meanings. Forms can be expressions composed of two different kind of strings. These two strings are phonemes or inscriptions composed of a string of characters. Form composed with sense in a language is called a word in that language. The dictionary lists these words in alphabetical order (Miller, 1995).

Wordnet represents form and sense in different manner. In Wordnet, a form is represented by a string of ASCII characters. Sense is then represented by the set of (one or more) synonyms that have that sense. In 1995 approximately 40% of the words in Wordnet did have one or more synonyms. Linking synonym sets by semantic relations in wordnet were chosen because they apply widely throughout English language and because they are usually familiar for user. The benefit of having familiar semantic relations to user is that the users do not need to have advanced training in linguistics to understand them. Wordnet includes following semantic relations:

- Synonymy (similar)
- Antonymy (opposite)
- Hyponymy (subordinate)

- Meronymy (part)
- Troponymy (manner)
- Entailment

Synonymy is a basic relation in Wordnet. This is because the usage of sets of synonyms to represent word senses. Synonymy is a symmetric relation between word forms.

Antonymy means opposing-name and it is also a symmetric semantic relation between word forms. Antonymy is especially important in organizing the meanings of adjectives and adverbs.

Hyponymy means sub-name and it is the opposite for hypernymy. Hypernymy means super-name. Both of these are transitive relations between sets of synonyms. There is usually only one hypernym so this semantic relation organizes the meaning of nouns into a hierarchical structure.

Meronymy is a part-name and it is opposite for holonymy which means whole-name. Meronymy and holonymy are both complex semantic relations. Wordnet uses these two to distinguish component parts, substantive parts and member parts.

Troponymy which means manner-name is for organizing verbs into hierarchical order. Usually resulting in shallow hierarchies.

Entailment relations between verbs are also coded in Wordnet (Miller, 1995).

As Wordnet is a dictionary made for machines it can be used also for image matching by utilizing it with annotated images. Wordnet helps when the task is to find an image with certain keywords, which is often the task for finding image. These keywords describe the content of the image, so it is important that the keywords are matched. Content-based image matching computes relevance based on visual content. The visual content can range from colours, textures and shapes to layout. However, sometimes this kind of approach is not enough to find correct images or it narrows the search down too much as visual similarity is not semantic similarity. This semantic gap can be a major problem for example, in cases when querying for “red ball” a content-based image matching system may answer with an image of a “red rose” (Jin et al., 2005).

Querying images with metadata can be hard as images can be described with different words which mean similar things, and some images might have too common words. For example, when querying for image with word “valley” the user might mis also images

with word “mountain” even though they could be totally correct. However, when semantic relations are taken into considerations the users query could be satisfied when query of “valley” would also return images which contain “mountain” (Jin et al., 2005).

3 APPLICATIONS FOR IMAGE MATCHING

The number of devices with internet capability and camera is increasing rapidly and becoming more and more popular. This development means that billions of people are capable of sharing their images throughout Internet but also capable for browsing huge number of images. Image searching has been a subject of research already from the early 1990s but recently it has become even more popular. New technology and new techniques are able to help with the scalability challenges. For example, traditionally the images were indexed by their metadata like title and tags. This metadata could have been very inconsistent making the image search slow and more difficult (Wengang Zhou, Houqiang Li & Qi Tian, 2017, Chen, Wang, 2002).

Image matching gives the ability to search for one or more matching images. It can be also used to find valuable information from images. Image matching techniques can find images which differ from the input image. The differences can be for example, image have been taken in different time and lighting conditions, from different viewpoint or by different sensor. The ability for being able to find interesting images from a large image set can be valuable in different fields like, in medicine (M. Brown, R. Szeliski & S. Winder, 2005).

While image matching itself includes many different methods and techniques, the usefulness of the matching techniques depends on the type of the images and the object image set. Image matching applications range from object recognition to individual identification of wildlife. Some examples, for applications are robotic mapping, robotic navigation, image stitching, 3D modeling, gesture recognition, video tracking and match moving. These applications are used within many robotic vision and image processing tasks (Montazer, Giveki, 2015).

One of the uses for image matching is video stabilization. This is why unstable videos are becoming more common, as many handheld devices carry also a camera which can record videos. Unstable videos can be caused by many things such as shake of the camera holder or unstable platform for surveillance cameras. Video stabilization is a process where disturbed, shaky or unstable motions are removed programmatically from video (Pinto, Anurenjan, 2011).

We next survey some common usage situations and tools of image matching. These include cloud-based tools like Azure computer vision API and automatic categorization for images.

3.1 Applications with content-based image matching

A benefit of content-based image matching analysis is its efficiency. It is typically used on rigid structures and the applications for it come from areas utilizing this feature. This kind of applications occur for example in digital libraries which scan lots of documents (Harley, Ufkes & Derpanis, 2015).

Another way to harness the efficiency of content-based image matching is to collect tattoo images. Tattoo images are usually collected by law enforcement to identify suspects and victims. Initially the image databases were queried by the metadata of the images. Using keywords solely to query images can be a tedious and subjective task. Also, submitting new images into image database is time-consuming as images were annotated manually. Retrieving images with content-based image matching simplifies and improves the image retrieval process. For tattoo image retrieval, content-based image matching can be used to extract low level visual content of tattoo images. This low-level content like color and texture is then used to represent and match images from the image database without the cumbersome use of keywords (Jain et al., 2009).

3.2 Azure computer vision API

Azure is a foundation for cloud computing services, offered by Microsoft. It can be used to run applications and store data on machines in internet-accessible data centers Azure can run many different kinds of applications, but the recommended model is to spread roles within Azure. Spreading roles within Azure to multiple resources means that applications should be able to scale. Azure also serves resources as platforms where for example, SQL server functionality is offered without the need to install and maintain the actual server by the user (Chappell, 2009).

As artificial intelligence is becoming more important, and many devices and applications already utilize different algorithms to improve the lives of people, it is becoming more beneficial for developers to get familiar with artificial intelligence. Hyperscale cloud vendors offer wide range of sophisticated algorithms that can be used with different approaches. Microsoft Cognitive Services is one of the services that offers such algorithms. The Microsoft Cognitive Service algorithms can be used via the representational state transfer (REST) application interface (API) by creating standard calls or for example, with .NET implementation of the API. By having REST API offering, Microsoft Cognitive Services enables developers to implement their application intelligently cross-platform and cross-device. Developers can also utilize this type of API with different kind of applications ranging from mobile application to applications used with internet browser. Azure computer vision API is a service created to understand and interpret the content of images. Azure computer vision API provides natural language description which can be utilized with other Microsoft services like the Speech API and the Translation API. Computer vision API can also be used to analyze images for optical character recognition, to detect print and handwritten words and sentences (Del Sole, 2017).

The computer vision API provides a rich set of different RESTful APIs, which use the Javascript Object Notation (JSON) standard data exchange format. This combination of APIs and JSON enables developers to create applications which could for example, help kids to learn about the world from pictures while at same time making sure that adult content is excluded. Similarly, computer vision API could be used to create a custom solution for identifying criminals from images (Del Sole, 2017).

Computer vision API can be used by invoking the RESTful application interfaces by uploading an image or pointing to an existing image URL. The type of analysis which will be executed for the image is then chosen by sending either GET or POST HTTP request. Computer vision API utilizes JSON as data exchange format so also with image analysis the response comes in the JSON format. The response JSON structure contains the analysis results (Del Sole, 2017).

Computer vision API has a wide range of different analysis types. The APIs are usually used with a HTTP request to the related endpoint. Each Azure region has its own endpoint

URLs in order to reduce latency and have services closer to the user. The services in computer vision API include for example:

- Analyze image
- Describe image
- Get thumbnail
- List domain specific models
- OCR
- Recognize domain specific content
- Recognize handwritten text
- Tag image

Analyze image API can be used to analyze images for adult and racial content. It can also do facial recognition, tagging and dominant colors.

Describe image API can generate description for an image. Description can be created in human-readable language with complete sentences.

Get thumbnail API generates a thumbnail from the specified image.

List domain specific model's API can get the list of currently supported domain-specific models. These models can be for example, celebrity and landmark recognizers.

OCR in OCR API stands for optical character recognition which means that the API performs OCR recognition over an image and stores the detected text into machine-usable characters. Recognize domain specific content API retrieves domain specific content from analyzed images.

Recognize handwritten text API recognizes handwritten text from images.

Tag image API generates a list of words which are relevant to the content of the specified image. The API replies with JSON response where the response contains array of tags. This array includes the name for the tag and the confidence percentage (Del Sole, 2017).

3.3 Image categorization using cloud based tools

Cloud and its services provide several advantages especially when it comes to processing large data sets. Cloud provides ways for immediate access to resources and provides

resources for compute intensive tasks. However, using cloud services is not a silver bullet as integrating all of these systems can become a tedious task. Usually cloud services offer tools to do these integrations. For example, Azure Logic App workflow can be triggered by uploading images to a cloud storage where Logic App connector exists. Logic App workflow can then move the uploaded image to a specified location for further processing (Šeškar, Milašinović & Fertalj, 2018).

Azure Logic App cloud service allows passing instructions and data from a cloud service to another essentially by integrating cloud services together. Azure Logic App allows also creating workflows that manage and apply the tools and frame the workflow in a serverless architecture. Azure Logic App with its features, helps to create scalable and modular implementations for categorizing images by integrating Azure cognitive services and other needed services together. Integration is done by using various API interfaces on the Internet by utilizing Logic Apps which are designed through a GUI or written in text (Šeškar, Milašinović & Fertalj, 2018).

Azure Computer Vision API analyses an image and generates a pair of tags for the image. A tag consists of name of the image and confidence score of the result of the analyzing. Name and confidence score are then used to decide the next execution path for the rest of the workflow. Azure Vision API is also used to extract other information which will be used in other workflow components. The extra information includes optical character recognition (OCR) method which will be used to scan license plates from potential vehicles in photo (Šeškar, Milašinović & Fertalj, 2018).

Figure 10 illustrates the image classification workflow from start to end. Black arrows show the Logic App workflow. Workflow starts by uploading the images to the OneDrive from where they are categorized by sending them to the cognitive computing service for analysis. After the analysis they are moved to appropriate subfolders. The image classification process starts automatically after the user has uploaded an image to OneDrive. Logic App has a special OneDrive connector which can be set to trigger whenever a new image is uploaded to a specified folder (Šeškar, Milašinović & Fertalj, 2018).

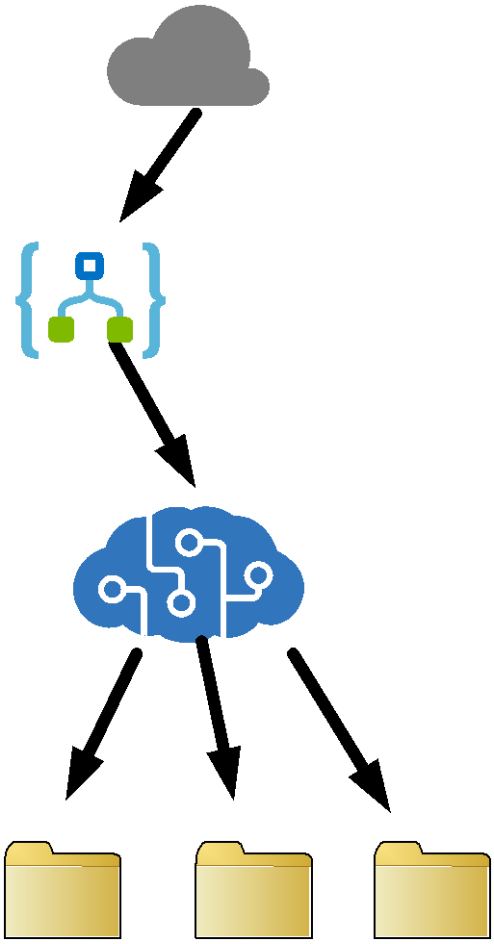

Figure 10. Example of image classification workflow

3.4 Retrieving handwritten documents

Documents are being digitized and indexed in large scale projects where the retrieval of handwritten documents has begun to be more important. Most of the retrieval techniques for handwritten documents try only to match the queried text with the document. Text matching provides a simplistic view of the document as the document might contain other interesting things such as a particular style of handwriting. The user might be interested in querying text which is written in cursive or loopy for example. This means that the system needs to be able to do query based on the content of the document. This type of retrieval systems is referred to as "*Content based retrieval*" (Bhardwaj et al., 2010).

Conventionally, retrieving text from documents is done by using OCR, but the problem with OCR is that not all text can be recognized robustly. Quality of the documents can

make the complete OCR conversion difficult. Also, some non-textual parts of the document cannot be converted into text. Because of the problems with OCR it may be sometimes beneficial to use techniques for direct characterization and manipulation of image features. These techniques for recognizing text could be for example, keyword spotting or word spotting. Keyword and word spotting give better results and they are also less complex than OCR (Varghese, Govilkar, 2015).

4 EXPERIMENTAL SEARCH ENGINE

This chapter reviews a proposed experimental search engine and the results which were obtained when testing it. The experimental search engine was built in the present study on the services of the open computer vision library (OpenCV). .NET C# was chosen as the programming language for the project and Windows Presentation Framework (WPF) was used as the graphical user interface. OpenCV has been built with C and C++ programming languages and its primary interface is by C++ which means that it cannot be directly used with the chosen C# programming language. EmguCV is a .NET wrapper for the OpenCV image processing library. EmguCV allows OpenCV functions to be called from C# language.

4.1 OpenCV

OpenCV is an open source computer vision library. The project was started in 1999 by Gary Bradski while he was working at Intel Corporation. OpenCV was created to accelerate computer vision and artificial intelligence projects by providing a solid infrastructure for everyone. OpenCV project is currently supported and maintained by multiple companies such as Intel, Google and Itseez (Kaehler, Bradski, 2016).

OpenCV was designed to provide computationally efficient interface for real time applications. It is written in optimized C++ and has been designed so that it takes advantage of multicore processors. OpenCV also supports different low-level optimized routines like Intel's Integrated Performance Primitives (IPP) library by automatically selecting appropriate IPP library at runtime if that library is installed (Kaehler, Bradski, 2016).

One of the goals for the OpenCV project is to provide a simple interface for building sophisticated computer vision applications quickly. The OpenCV library contains multiple functions which can be used for different purposes. Library functions span over multiple areas in vision such as factory product inspection, medical imaging, security and robotics. OpenCV also contains a full general-purpose *Machine Learning library* which is focused on pattern recognition and clustering while being general enough so that it can

be used in any machine learning problem. This library has been built-in, as often computer vision and machine learning go hand-in-hand (Kaehler, Bradski, 2016).

4.2 Requirements for the search engine

Some requirements were given for the search engine to have enough features for application. Requirements were kept narrow enough to keep the development time reasonable. The requirements for the experimental search engine were:

- .NET standard framework
- C# as programming language
- OpenCV library will be used
- Search engine needs to be able to iterate through directory tree structure
- Application needs to be responsive through use
- At least two different feature detectors need to be tested

Some of the requirements like having .NET standard framework and C# as programming language were good to have as strict requirements as they help much in the software design. However, having OpenCV library would not work without having an EmguCV kind of .NET wrapper. Searchable images are often stored in directory structures where images are stored in multiple subdirectories. This way of storing makes it reasonable to let the search engine search through these directories. The proposed search engine was required to have at least two different feature detectors which can be used with small modifications for the engine.

4.3 Design of the software for the search engine

The search engine was meant to be able to scan directory structures and find images from there. Another feature was to search for similarities between the images and display results to user. The results would possibly also include similarity scores between other images than the best one. The user interface of the software is simplified to include only relevant features. The user interface can be seen in Figure 11.

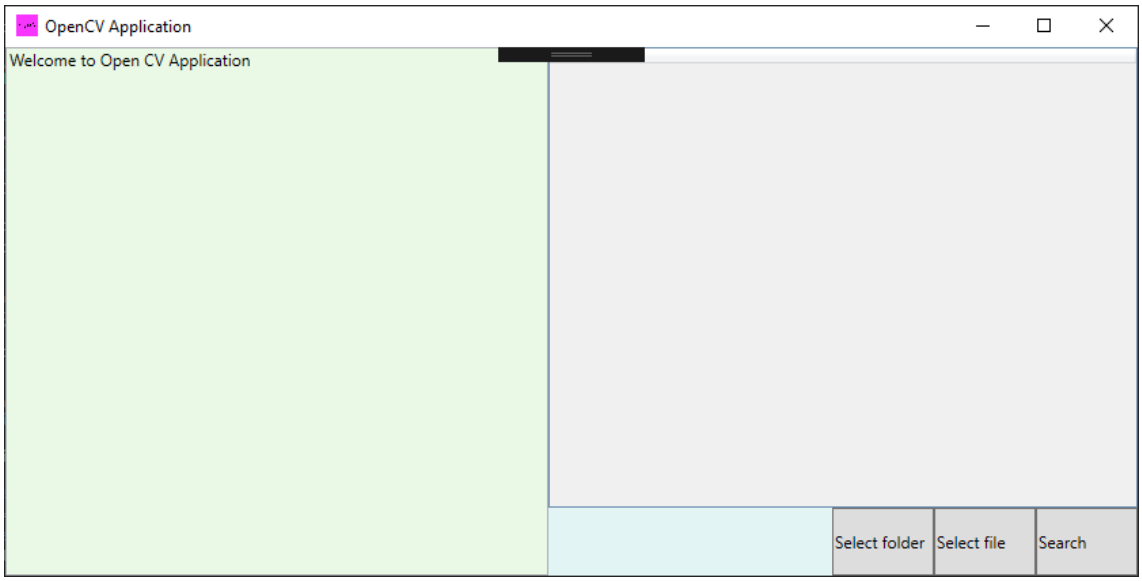

Figure 11. User interface for experimental search engine

Going through possibly hundreds of images could take very long time which means that with synchronous programming the graphical subsystem would need to wait long time until processing is done. Waiting for processing in user interface thread means that the user interface is completely frozen during that time. Freezing makes user experience very poor and longer freezing might make operating system to shut down the process entirely which would then mean lost results for image scanning.

The experimental search engine was programmed with asynchronous programming to mitigate multiple problems. One of the biggest problems was the freezing of the user interface where asynchronous programming makes great difference. .NET framework implements asynchronous programming with keywords async and await. Await means that the line of code is asynchronously waited.

The proposed program also follows Microsoft's Model-view-viewmodel (MVVM) software architectural pattern. The pattern helps to reduce the amount of code from the view layer making it easier to separate the responsibilities of the layers, see Figure 12. Code interacting with the GUI is bind into the viewmodel, which makes moving of the code to different layer possible. Architectural choices on the programming in the search engine make it very effective on using system resources. It can divide its tasks into multiple central processing units (CPU) making it faster to process the images.

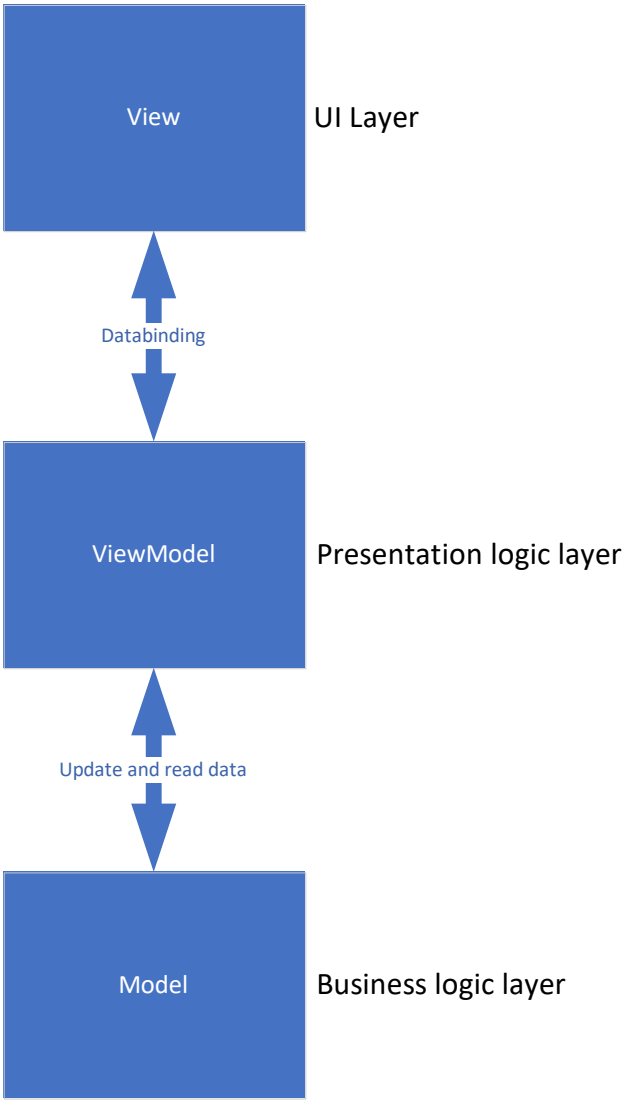

Figure 12. MVVM Software architecture pattern

Using a .NET wrapper for OpenCV made programming with .NET framework and C# programming language much easier since all the needed interfaces were already implemented. Programming with the .NET wrapper made it possible to concentrate more on the design and implementation instead of doing wrapping in the program itself. However, using .NET wrapper also brought some problems as the exceptions in EmguCV were not treated properly so the exception messages were very confusing. For example, running out of memory resulted with exception message “Unhandled Exception: Emgu.CV.Util.CvException: OpenCV: u != 0”.

The experimental search engine has 437 lines of code in total spread to 15 files. Testing was done with system having Windows 10 as operating system, Intel i7 2700k CPU and 16 gigabytes of RAM.

4.4 Tests

The experimental search engine was evaluated with a set of photos taken by professional photographer Juha Kurri. The photo set was chosen from a professional photographer to have high quality photos with different lighting and colors to see how the experimental search engine detects similarities between photos. The photos were used at a relatively high resolution, typically at 6016x4016. The photo set included 49 different photos from where one of the photos was chosen to be used as an evaluation photo. This evaluation photo was modified to get four different styles of the image. Styles were:

- Original photo
- Original photo scaled down to 10%
- Original photo converted to black and white
- Original photo with inverted colors

Evaluation was done with two different feature detectors. These feature detectors were KAZE and SURF. The SURF feature detector was also evaluated with three different Hessian thresholds to see the difference. These three Hessian thresholds were 100, 300, and 1000.

Figure 13. Photo by Juha Kurri. Original photo.

4.5 Results

Photos used in this evaluation were high resolution photos which brought some problems in testing. It was necessary to reduce the resolution as processing images with very high resolution made the application to consume more memory than what is allowed for 32-bit programs. Consuming too much memory made the EmguCV to throw exceptions which essentially stopped all testing.

The evaluation was started with the KAZE feature detector. Results did not differ much between original and black and white photos but scaling down or inverting colors brought the count of matching features down. As the test image was not removed from the searched images the top match count was always with similarities between the same image except for the case when colors were inverted. Inverting colors caused the highest similarity to be with the photo that had the same blue and white pattern as the image with inverted colors. With the KAZE feature detector, the second-best match count dropped down to 2476 when tested with the original photo. Match count is the count of matching similarities between the matched images.

Table 1. Results from the test run with KAZE feature detector

Photo	Elapsed time (minutes)	Match count
Original	6,7	14746
Scaled down to 10%	6,3	1726
Black & white	6,7	14166
Inverted colors	6,5	1244

The experimental search engine also draws lines between the two tested images to illustrate which features were detected. Figure 14 shows an example of the results with the KAZE feature detector.

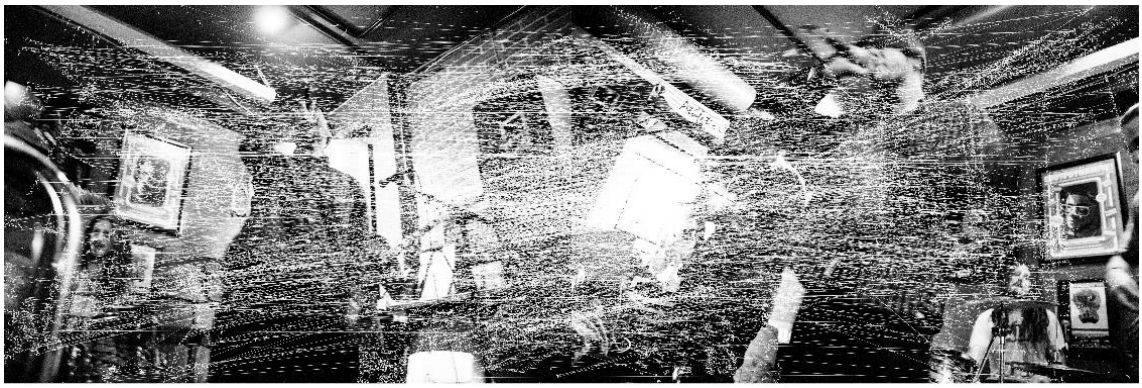

Figure 14. Photos by Juha Kurri. Lines drawn between the similarities between the most matching images.

The next evaluation was done with the SURF feature detector using 100 as the Hessian threshold. Image matching with SURF was much faster than with KAZE, SURF being more than three times faster. The highest similarity was between the original image and its copy. Second best match count was when the original image was compared with black and white image. Other results were similar to that of the KAZE feature detector. SURF as a feature detector was able to find more matches between images than KAZE. The higher number of matches can be seen by comparing Figure 14 and 15 together. In Figure 15 there is clearly more white lines pointing out similarities between these two photos.

Table 2. Results from the test run with SURF feature detector with Hessian threshold 100

Photo	Elapsed time (minutes)	Match count
Original	1,80	21904
Scaled down to 10%	1,70	2596
Black & white	1,90	16850
Inverted colors	2,00	2538

Figure 15. Photos by Juha Kurri. Lines drawn between the similarities between the most matching images.

The third test run was done by setting Hessian threshold to 300. This change of the Hessian threshold made the similarity search to run somewhat slower but bigger change was on the results for matchings. Similarity between the original image and copy of it was reduced almost to half now 12078 matches were found. Match count with the rest of the test images went down, see Table 3. The reduced match count can be also seen in Figure 16 which now has fewer white lines than the Figure 15.

Table 3. Results from the test run with SURF feature detector with Hessian threshold 300

Photo	Elapsed time (minutes)	Match count
Original	1,89	12078
Scaled down to 10%	1,85	1802
Black & white	1,98	10364
Inverted colors	2,07	1762

Figure 16. Photos by Juha Kurri. Lines drawn between the similarities between the most matching images.

The fourth and last test run was done with SURF feature detector using 1000 as Hessian threshold. In this test it is clear that 1000 for Hessian threshold is too large as it lowers the accuracy greatly. Raising Hessian threshold from 300 to 1000 reduced match count almost to half. The match count reduction can be again seen by comparing Figure 16 and Figure 17. In Figure 17 the amount of white lines showing similarities between images a clearly reduced. Match counts went down on all of the test photos. However, while match counts went down the running time of the similarity search also went down which can be seen in Table 4.

Table 4. Results from the test run with SURF feature detector with Hessian threshold 1000

Photo	Elapsed time (minutes)	Match count
Original	1,75	5684
Scaled down to 10%	1,66	748
Black & white	1,47	5230
Inverted colors	1,48	814

Figure 17. Photos by Juha Kurri. Lines drawn between the similarities between the most matching images.

4.6 Conclusions

Running similarity search with this experimental search engine takes time depending on the used feature detector. For example, running similarity search with the KAZE feature detector on a batch of 50 photos took almost seven minutes. Search engine gives results after running similarity search. User can also see what features are matching in the photos.

With these tests, it is clearly visible that SURF is a faster feature detector than KAZE. Although in this test tweaking the software especially to be used with KAZE was not done which would give different results. Also, reducing the image resolutions before testing affects the results but because of the already mentioned problems, tests could not be run without making the tested images suitable for the test system.

When running tests with SURF, the match count reduces if the Hessian threshold is raised. Therefore, the best similarity search could have been when using Hessian threshold under 100. SURF could have also made better similarity searches by tweaking the code to fit its use better.

The set of images for testing was not ideal for this test. Images seemed to be too dark with lots of black color which meant that the similarities found were between the black or very dark areas resulting in higher match count.

Open computer vision library is a great library and it helps quite much when it comes to developing applications for computer vision use. OpenCV has implement libraries to

many feature detectors and includes many examples on how to make development of an image search application in a straightforward way.

5 PUBLIC IMAGE SEARCH ENGINES

Some software companies provide image search engines publicly on internet. While many of them are provide generic image search engines, there are also engines for reverse image search. Reverse image search is a type of search which takes image as input and returns results which are related to the queried image. Reverse image search analyses the content of the image which helps the user to find exactly the same or similar image from web. Reverse image search engines can be used to detect unauthorized or copyright images which means that the search engine needs to be able to find also images which may be cropper, transformed or may have other changes (Gaillard, Egyed-Zsigmond, 2017).

5.1 Test plan

The aim was to test four different public image search engines which offer also reverse image search functionality. The search engines which were taken into test were:

- www.google.com
- www.bing.com
- www.tineye.com
- www.yandex.com

Google, Bing and Yandex are more traditional search engines which also provide the functionality for reverse image search while tineye is more concentrated on the image search. All search engines work pretty much similarly when doing the image search. First, the user gives the url of the image that he wants to search and then user presses the search button. After that, the search engine gives the results. Some search engines like Google gives also approximated information on how many results there were, how long did the search take and the web pages where similar picture could be.

5.2 Evaluation of public image search engines

Public image search engines were evaluated by running an image search with the standard test image called Lena. Standard test image of Lena is a colored image with its dimensions being 512 pixels on height and width same 512 pixels. The public search engines do not publish information how they do the similarity searches. To get more information on how well these search engines can find images the engines were also evaluated with five additional image types. This means that there are altogether six different images to be tested. These images are:

- Standard image without any modification
- Image scaled down to 10%
- Black and white image
- Image with inverted colors
- Image with Gaussian blur
- Image rotated 45 degrees counter-clockwise

Figure 18. Standard image without any modification

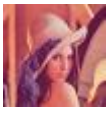

Figure 19. Image scaled down to 10%

Figure 20. Black and white image

Figure 21. Image with inverted colors

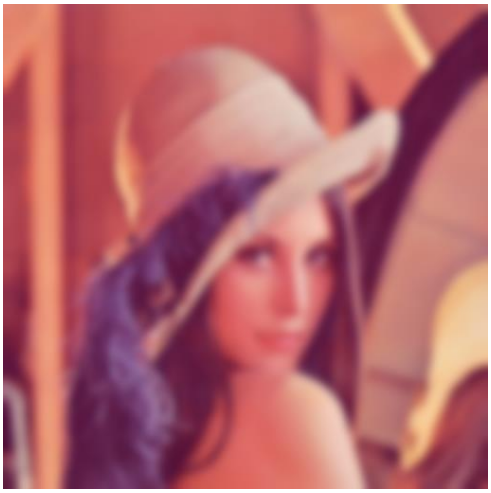

Figure 22. Image with Gaussian blur

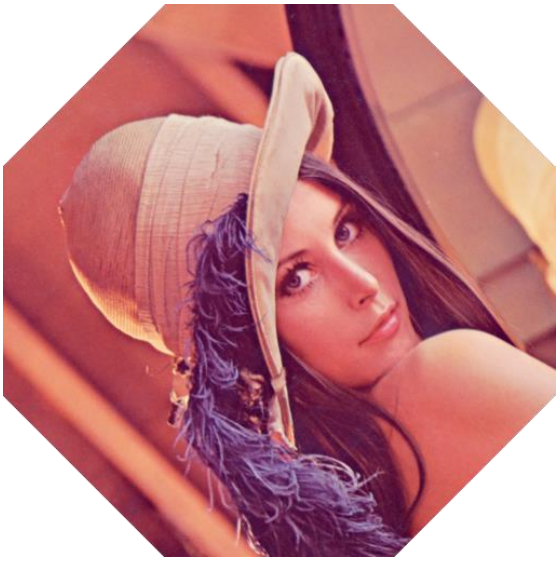

Figure 23. Image rotated 45 degrees counter-clockwise

All test images were uploaded into new locations to get new URL for each image. This new URL was created to prevent search engines having any caching made for the images.

5.3 Results

From the chosen search engines only Tineye and Google gave more information about the search like how many results did those manage to get. Google also shows how long does the search take. Bing and Yandex do not give any additional information about the search other than the results itself.

Google is most likely the most used search engine on Internet. When doing similarity search for images it could find lots of similar images for each tested image. For each tested image Google was able to find more than millions of similar images. The exact number was not given and with each result the count was approximated. Doing similarity search with black and white image did give most hits. Blurring the image dropped the result count but Google search engine was still able to find many similar images to it. Biggest drop in the result count came when image with rotation was used.

Table 5. Results from Google similarity search

Image	Hit count
Standard	2360000000
10Percent	2160000000
Black&White	25270000000
Inverted colors	1910000000
Gaussian blur	2300000000
45Deg	1690000000

Tineye was not able to find as much hits as Google did. That is most likely because the use of Tineye is different from that of the Google image search engine. Results with Tineye were however, still similar as with Google. Rotating image dropped the count of found matches down to seven while image without modifications and black and white image gave the best results.

Table 6. Results from Tineye reverse image search

Image	Hit count
Standard	4131
10Percent	3581
Black&White	4124
Inverted colors	38
Gaussian blur	3135
45Deg	7

Since Bing and Yandex did not give any count on found matches it was difficult to compare them with the other search engines. However, it was visually verified that both search engines were able to find similar images. Yandex was able to keep the search results more relevant than Bing. For example, when doing similarity search with Bing for the rotated image, the software found several completely different images given as result.

5.4 Conclusions

When doing similarity searches on these four search engines it is clearly visible that the same image has been used quite much in different machine vision projects. When doing search with the standard Lena image the search engines gave also black and white photos, rotated photos and cropped photos as results. This affects the results and counts of found images for Google and Tineye.

6 CONCLUSIONS

Image matching and retrieval has concentrated on the content-based image matching and content-based image retrieval. Initially research was done by comparing images pixel by pixel and try to find matching pixels. However, this approach has its own drawbacks which has resulted the research to aim to another direction. From the early 2000's the research has concentrated more on recognizing objects and features from the images. Research around the object recognition approach has created more techniques to describe images and find features from them. The features for object recognition have their own strengths and therefore users of these techniques need to think which one to use.

As not all the recognition is done for images there are also more text-based approaches for identifying and reading documents. These techniques can be used for example for recognizing hand written text or for reading texts from paper to digitalize it. Optical character recognition is also one of the technologies which are developed and used in different applications.

Computer vision is a hot topic on current technology and it is under constant research as there are many applications where it can be used and more of those applications are constantly invented. Image matching and image retrieval are also big topics in the area of computer vision and therefore there is also lots of research being done for those. Image matching includes lots of mathematics and lots of different techniques and methods which makes it little bit hard to grasp the concepts on how image matching works and how it should be used. The different approaches might also make it hard for the starters to figure out which approach and technique to use. However, there is many open source libraries for doing image matching which makes it easier to create applications without knowing the very specific details of how image matching works.

Hobbyists can use different open sourced computer vision libraries for their needs or write their own algorithms. For example, finding certain lines from video feed is rather simple to create even when starting from scratch which makes computer vision an interesting area.

Computer vision is mixed with machine learning to utilize neural networks in teaching machines to identify images more accurately and also to identify images without the exact source image. However, there is still room for simpler feature detectors like the KAZE

and the SURF. As tested also with the proposed experimental search engine there is a lot of differences between different image matching algorithms and techniques. This big number of algorithms and techniques means that users need to find out the most suitable technique and algorithm for their use. It also depends on the use case whether it makes more sense to adapt neural networks or simpler methods on finding the features. Configuring neural networks in small applications might not be worth the work.

REFERENCES

A. Amaricai, C. Gavriliu & O. Boncalo 2014, "An FPGA sliding window-based architecture harris corner detector", *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1.

Alahi, A., Ortiz, R. & Vandergheynst, P. 2012, "Freak: Fast retina keypoint", *2012 IEEE Conference on Computer Vision and Pattern Recognition*Ieee, , pp. 510.

Bay, H., Ess, A., Tuytelaars, T. & Van Gool, L. 2008, "Speeded-up robust features (SURF)", *Computer Vision and Image Understanding,* vol. 110, no. 3, pp. 346-359.

Bhardwaj, A., Thomas, A.O., Fu, Y. & Govindaraju, V. 2010, "Retrieving handwriting styles: A content based approach to handwritten document retrieval", *2010 12th International Conference on Frontiers in Handwriting Recognition*IEEE, , pp. 265.

Chappell, D. 2009, "Introducing windows azure", *Microsoft, Dec,* .

Chen, Y. & Wang, J.Z. 2002, "A region-based fuzzy feature matching approach to content-based image retrieval", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 9, pp. 1252-1267.

Ciresan, D.C., Meier, U., Masci, J., Gambardella, L.M. & Schmidhuber, J. 2011, "Flexible, high performance convolutional neural networks for image classification", *Twenty-Second International Joint Conference on Artificial Intelligence*.

Del Sole, A. 2017, *Microsoft Computer Vision APIs Distilled: Getting Started with Cognitive Services,* Apress.

Deng, J., Dong, W., Socher, R., Li, L., Li, K. & Fei-Fei, L. 2009, "Imagenet: A large-scale hierarchical image database", *2009 IEEE conference on computer vision and pattern recognition*Ieee, , pp. 248.

Derpanis, K.G. 2004, "The harris corner detector", *York University,* .

Figat, J., Kornuta, T. & Kasprzak, W. 2014, "Performance evaluation of binary descriptors of local features", *International Conference on Computer Vision and Graphics*Springer, , pp. 187.

Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D. & Petkovic, D. 1995, "Query by image and video content: The QBIC system", *computer,* vol. 28, no. 9, pp. 23-32.

Gaillard, M. & Egyed-Zsigmond, E. 2017, "Large scale reverse image search", .

Halawani, A., Teynor, A., Setia, L., Brunner, G. & Burkhardt, H. 2006, "Fundamentals and Applications of Image Retrieval: An Overview.", *Datenbank-Spektrum,* vol. 18, no. 14-23, pp. 6.

Harley, A.W., Ufkes, A. & Derpanis, K.G. 2015, "Evaluation of deep convolutional nets for document image classification and retrieval", *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*IEEE, , pp. 991.

Harris, C. 1993, "Geometry from visual motion, Active vision", .

Hutcheson, G.D. 2011, "Ordinary least-squares regression", *L.Moutinho and GD Hutcheson, The SAGE dictionary of quantitative management research,* , pp. 224-228.

Jain, A.K., Lee, J., Jin, R. & Gregg, N. 2009, "Content-based image retrieval: An application to tattoo images", *2009 16th IEEE international conference on image processing (ICIP)*IEEE, , pp. 2745.

Jin, Y., Khan, L., Wang, L. & Awad, M. 2005, "Image annotations by combining multiple evidence & wordnet", *Proceedings of the 13th annual ACM international conference on Multimedia*ACM, , pp. 706.

Kaehler, A. & Bradski, G. 2016, *Learning OpenCV 3: computer vision in C with the OpenCV library,* " O'Reilly Media, Inc.".

Karami, E., Prasad, S. & Shehata, M. 2017, "Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images", *arXiv preprint arXiv:1710.02726,* .

Kniss, J., Premoze, S., Ikits, M., Lefohn, A., Hansen, C. & Praun, E. 2003, "Gaussian transfer functions for multi-field volume visualization", *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*IEEE Computer Society, , pp. 65.

Krizhevsky, A., Sutskever, I. & Hinton, G.E. 2012, "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, pp. 1097.

Lowe, D.G. 2004, "Distinctive Image Features from Scale-Invariant Keypoints", *Computer Science Department, University of British Columbia, Vancouver, B.C., Canada,* , no. 2.

M. Brown, R. Szeliski & S. Winder 2005, "Multi-image matching using multi-scale oriented patches", *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pp. 510.

Mair, E., Hager, G.D., Burschka, D., Suppa, M. & Hirzinger, G. 2010, "Adaptive and generic corner detection based on the accelerated segment test", *European conference on Computer vision*Springer, , pp. 183.

Miller, G.A. 1995, "WordNet: a lexical database for English", *Communications of the ACM,* vol. 38, no. 11, pp. 39-41.

Montazer, G.A. & Giveki, D. 2015, "Content based image retrieval system using clustered scale invariant feature transforms", *Optik,* vol. 126, no. 18, pp. 1695-1699.

Moravec, H.P. 1981, "Rover Visual Obstacle Avoidance.", *IJCAI*, pp. 785.

Muja, M. & Lowe, D.G. 2012, "Fast matching of binary features", *2012 Ninth conference on computer and robot vision*IEEE, , pp. 404.

Müller, H., Michoux, N., Bandon, D. & Geissbuhler, A. 2004, *A review of content-based image retrieval systems in medical applications—clinical benefits and future directions*.

Oliveira, F.P.M. & Tavares, J.M.R.S. 2014, "Medical image registration: a review", *Computer methods in biomechanics and biomedical engineering,* vol. 17, no. 2, pp. 73-93.

Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S. & Cheng, X. 2016, "Text matching as image recognition", *Thirtieth AAAI Conference on Artificial Intelligence*.

Pinto, B. & Anurenjan, P. 2011, "Video stabilization using speeded up robust features", *2011 International Conference on Communications and Signal Processing*IEEE, , pp. 527.

Rublee, E., Rabaud, V., Konolige, K. & Bradski, G. 2011, "ORB: An efficient alternative to SIFT or SURF", .

Rui, Y., Huang, T.S. & Chang, S. 1999, *Image Retrieval: Current Techniques, Promising Directions, and Open Issues*.

Šeškar, A., Milašinović, B. & Fertalj, K. 2018, "Workflow for image categorization using cognitive computing services", *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*IEEE, , pp. 1551.

Shoujia, W., Wenhui, L., Ying, W., Yuanyuan, J., Shan, J. & Ruilin, Z. 2012, "An Improved Difference of Gaussian Filter in Face Recognition", , no. Journal of multimedia, vol. 7, no. 6, december 2012.

Shrivastava, N. & Tyagi, V. 2014, *Content based image retrieval based on relative locations of multiple regions of interest using selective regions matching*.

Stanković, R.S. & Falkowski, B.J. 2003, "The Haar wavelet transform: its status and achievements", *Computers & Electrical Engineering,* vol. 29, no. 1, pp. 25-44.

Varghese, B. & Govilkar, S. 2015, "A survey on various word spotting techniques for content based document image retrieval", *International Journal of Computer Science and Information Technologies,* vol. 6, pp. 2682-2686.

Wang, T., Wu, D.J., Coates, A. & Ng, A.Y. 2012, "End-to-end text recognition with convolutional neural networks", *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*IEEE, , pp. 3304.

Weisstein, E.W. 2004, "Affine transformation", .

Wengang Zhou, Houqiang Li & Qi Tian 2017, "Recent Advance in Content-based Image Retrieval: A Literature Survey", vol. abs/1706.06064.

Zeng, Q., Liu, L. & Li, J. 2010, "Image registration method based on improved Harris corner detector", *Automation Department, School of Electronic & Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China,* vol. 8.