BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

DEEP LEARNING FOR IMAGE STITCHING

HLUBOKÉ NEURONOVÉ SÍTĚ PRO SEŠÍVÁNÍ OBRÁZKŮ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR — **DIANA MAXIMA DRŽÍKOVÁ**
AUTOR PRÁCE

SUPERVISOR — **Ing. MICHAL ŠPANĚL, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

146201

Institut:	Department of Computer Graphics and Multimedia (UPGM)
Student:	**Držíková Diana Maxima**
Programme:	Information Technology
Specialization:	Information Technology
Title:	**Deep Learning for Image Stitching**
Category:	Image Processing
Academic year:	2022/23

Assignment:

1. Get familiar with deep neural networks and their learning.
2. Get acquainted with the problems of image stitching and keypoint detection. Become familiar with existing deep learning models suitable for the image stitching.
3. Prepare a dataset for your own experiments.
4. Implement chosen models and experiment with them.
5. Evaluate and compare your results using appropriate metrics. Discuss possible future work.
6. Create a short poster or video presenting your work, its goals and results.

Literature:

- Sarlin et al., "SuperGlue: Learning Feature Matching with Graph Neural Networks", *CVPR 2020,* https://arxiv.org/abs/1911.11763.
- Nie et al., "Deep Rectangling for Image Stitching: A Learning Baseline", *CVPR, 2022,* https://arxiv.org/pdf/2203.03831v4.pdf.

Requirements for the semestral defence:

- The first three items of the assignment.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Španěl Michal, Ing., Ph.D.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2022
Submission deadline:	10.5.2023
Approval date:	3.11.2022

Abstract

Stitching digital images is not something unfamiliar to the average technology user. The most common example of stitching can be found in panoramic images, where the algorithm stitches them to achieve a seamless, high-quality picture. Various steps need to be executed to stitch the images. Feature detection, description, and matching play the most important role in achieving the goal. This thesis will dwell deeper into the stitching problematic and will discuss the possible solutions. The traditional approaches to stitching will be explained in order to understand the basic idea behind it. Later on, the neural networks will be used to enhance the feature processing. The SuperPoint and SuperGlue neural networks will be discussed and used for their experiments. The main product of this work is a matching algorithm which uses the SuperPoint and SuperGlue models to stitch the images from grids. Other experiments which helped the process of understanding this problem, will be explained and evaluated.

Abstrakt

Zošívanie obrázkov nie je taký neznámy pojem ako sa na prvý pohľad môže zdať. Určite každý bežný používateľ technológií sa už zozámil s pojmom panoramatický obrázok. V pozadí na zariadení sa prekrývajúce sa obrázky zošívajú a tým vzniká vysoko kvalitný obrázok. Na to aby tento proces fungoval, existujúce algorimy musia spoľahlivo a presne detekovať zaujímavé body, podľa ktorých sa dokáže obrázok správne umiesniť. V tejto práci budú predstavené tradičné metódy na zošívanie obrázkov a taktiež aj metódy s pomocou hlbokých neurónových sietí. Hlavné dva modely, ktoré budú opísane a použíté sú implementácie SuperPoint a SuperGlue. Implementácia bude adaptovaná na párovací systém pre viac ako dva obrázky. Ostatné experimenty, ktoré boli vyskúšané a dopomohli k pochopeniu tejto problematiky budú opísane a vyhodnotené.

Keywords

deep learning, deep neural networks, convolutional neural networks, image stitching, methods of image stitching, graph neural networks, SuperPoint, SuperGlue, SIFT, homography estimation

Klíčová slova

hlboké učenie, hlboké neurónové siete, konvolučné neurónové siete, zošívanie obrázkov, metódy zošívania obrázkov, grafové neurónové siete, SuperPoint, SuperGlue, SIFT, estimácia homografie medzi obrázkami

Reference

DRŽÍKOVÁ, Diana Maxima. *Deep Learning for Image Stitching.* Brno, 2023. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Michal Španěl, Ph.D.

Rozšířený abstrakt

Pojem zošívanie obrázkov nie je taký neznámy, ako sa na prvý pohľad môže zdať. Panoramatické obrázky či obrázky so širokým zorným poľom sú algoritmicky generované v pozadí zariadenia. Tieto obrázky sú produktom viacero obrázkov, ktoré sa do určitej miery prekrývajú. Na to, aby sa obrázky boli schopné prekryť a namapovať na seba, je potrebné použiť efektívny algoritmus na detekovanie a párovanie záchytných bodov na obrázkoch.

Intuitívny prístup ku zošívaniu obrázkov spočíva v mapovaní tam a naspať podobných zaujímavých miest. Tieto zaujimavé miesta sa môžu označovať ako kľučové body. Tieto kľúčové body sú charakterisické svojou štruktúrou, vzorom či farbou. Sú jednoznačne detekovateľné v ich najbližšom okolí.

Tieto kľúčové body sa musia opísať takým spôsobom, aby sa dali nájsť aj na inom obrázku. Tento popis pozostáva z ich farby, jasu či vzoru. Berie sa na ohľad aj ich najbližšie okolie, aby sa im dodal priestorový kontext.

Po opísaní kľúčových bodov je potreba ich spárovať. Spárovanie prebieha pomocou hľadania podobných bodov v priestore a vyhodnocovania ich priestorovej vzdialenosti. Páry, ktoré boli spolu omylom namapované sa týmto overením odstraňujú.

Spárovanie odpovedajúcich bodov na jednotlivých obrázkoch pokračuje v rozhodovaní, ako ich na seba premietnuť. Tento priemet bežne spočíva v estimácií matice homografie. Táto matica je prispôsobená tomu, aby bola dostatočne generická na to, aby dokázala správne premietnuť jednoltivé body, s čo najmenšiou chybovosťou.

Základna časť zošívania obrázkov je zakončená priemetom celého obrázku do súradnicovej sústavy druhého obrázku.

Existuje mnoho prístupov, ktoré dokážu túto problematiku riešiť. Jeden z najznámejších reprezentatov na detekciu kľúčových bodov a ich opisu je algoritmus SIFT. Tento algorithmus funguje na základe hľadania v istom zmysle opakujúcich sa bodov, ktoré vedia byť nájdené na obrázku aj po určitej transformácií obrázku. Najbežnejšia metóda na hľadanie párov, je vyhodnocovanie Euklidovskej vzdialenosti medzi bodmi. Po nájdení párov, je zvykom premietnuť body z jedného obrázku na druhý obrázok pomocou spomínanej matice homografie. Proces tvorenia tejto matice je za pomoci tradičného algoritmu RANSAC zefektívnený. Matica je vyhodnotená iteračne počas toho, ako sa hľadajú najlepšie a najsilnejšie páry, z ktorých sa daná projekčná matica vypočíta.

Existuje mnoho ďalších úspešných algorithmov na jednotlivé kroky. Jedny z najúspešnejších metód sú za pomoci neurónových sietí. Neurónové sieťe pozostávajú z vrstiev a tzv. neurónov, ktoré sú medzi sebou poprepájané. Na jednotlivých vrstvách sa nachádzajú trénovateľné váhy, ktoré sa aktualizujú podľa chybovosti predikcie s očakávaným výstupom.

Neurónové siete sú trénované na veľkých dátových sadách s rôznymi dátami. Typy trénovania sa môžu rozdeliť do dvoch skupín. Trénovanie s dozorom, kedy sú sieti poskytuté očakávané odpovede a trénovanie bez dozoru, kedy sieť sama vyhľadáva vzory a štruktúry vo vstupných dátach. Hlavný dataset na vykonanie experimentov obsahuje mikroskopické obrázky z verejných domien, ktoré sú dostupné na experimentálne účely. Táto dátová sada bude jednotlivo prispôsobená navrhnutým neurónovým sieťam.

Táto práca predstaví dve experimentálne neurónové siete, ktoré budu slúžiť k vykonaniu jedného z kroku procesu zošívania obrázkov. Následne sa k ďalšiemu experimentu budú používať SuperPoint a SuperGlue neurónové siete, ktoré sú schopné produkovať vysoko kvalitné výsledky. SuperPoint produkuje predpovede pre každý pixel, aká je pravdepodobnosť, že daný pixel je kľúčový bod. Sieť zároveň generuje aj deskriptory, ktoré popisujú jednotlivé body. SuperGlue neurónová sieť za pomoci kľúčových bodov a deskriptorov z jednotlivých

obrázkov, dokáže správne namapovať podobné body. Schopnosti neurónových sietí budú porovnané s tradičnými metódami, ktoré boli popísané vyššie.

Porovnávanie týchto metód ukázalo, že neurónové siete si poradili s mikroskopickými obrázkami oveľa lepšie ako tradičné metódy. S rozptylom maximálne troch pixelov pre chybovosť, je SuperPoint a SuperGlue schopný správne určiť s 86% presnostou správne páry na obrázkoch.

Prvá neurónová sieť sa bude bez poskytnutia správnych výsledných hodnôt trénovať na detekciu kľúčových miest na mikroskopických obrázkoch. Dataset vytvorený na tento účel obsahuje 64×64 pixelové obrázky, ktoré obsahujú detekované kľučove body pomocou SuperPointu. Vytvorená dátova sada obsahuje približne 150,000 obrázkov s rôznymi zaujímavými bodmi. Na validáciu siete, je použitá metóda Silhouette, ktorá vypočíta skóre správnosti vytvorených zhlukov v rozpätí -1,1 kde 1 je najlepšie skóre. Zhluky sú vytvorené pomocou metódy zhlukovania najbližších stredov. Pre mikroskopické obrázky je skóre 0.16 a pre jednoduhé objekty a tvary 0:67.

Druhá neurónová sieť je určená na predpoveď homografickej matice. Táto homograficka matica je produktom tradičného algoritmu a párov, ktoré poskytla SuperGlue neurónová sieť. Trénovanie neurónovej siete s dvoma typmi vstupných dát, skončila neúspešne.

Finálnym produktom tejto práce bude adaptovaný párovací algorithmus, ktorý používa kombináciu tradičných metód ako aj metód hlbokého učenia. Tento algorithmus bude odskúšaný na dátovej sade poskytnutej od firmy TESCAN.

Deep Learning for Image Stitching

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Ing. Michal Španěl Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Diana Maxima Držíková
May 10, 2023

Acknowledgements

I would like to express my gratitude for the guidance and advice provided by my supervisor Ing. Michal Španěl, Ph.D. Each idea and opinion was very useful and helped me to understand this topic in more depth. Many thanks to all my family members who supported me during my whole studies. Your encouragement was essential to me. Thank you, Mária Novákova for being here for me. Last but not least, I would like to thank my friends for all the encouraging words.

Contents

Chapter 1

Introduction

A panorama is a broad perspective of a physical area, typically represented by a photograph or painting. Taking panoramic pictures is well-known to almost every tech user. Taking pictures of this nature can be challenging due to various struggles. Obscurities like the proper movement of the camera, unsteady hands, a shallow depth of field, and lens distortion are very common problems. The author of this work, who is also a beginner photographer, is familiar with those struggles. All of these problems can create noticeable seams or distorted pictures. Even with amateur skills, outcomes are usually sufficient, but how is it possible? The process of image-stitching is running in the background of the device [28]. One can create a seamless image with the assistance of real-time stitching. Several images with overlapped fields need to be aligned as part of the image-stitching process. This process and its various approaches will be discussed.

Image stitching plays an important role not just in photography but in many other professions as well. It has applications in virtual reality, gaming, surveying, security, and surveillance. In the medical industry, stitching is also essential when it comes to tasks like taking X-ray photos. It is also essential for the procedure that generates microscopic images of the highest quality.

The main goal of this thesis is to use the power of SuperPoint and SuperGlue neural network to create a grid matching pipeline for microscopic images. During the process of getting familiar with image stitching and neural networks, various experiments are conducted and evaluated. The experiments dwell in training the neural networks on various tasks such as keypoint detection and homography estimation. Not every one of them ended up in success, but they still played an important role to understand the topic in more depth.

The SuperPoint and SuperGlue will be during the thesis evaluated and compared to handcrafted approaches such as SIFT. The final evaluation will show, that the SuperPoint and SuperGlue can nearly in any conditions accurately detect and match features on the images. The models performed well even on noisy data, with 88% accuracy with three-pixel dispersion. Evaluation of the microscopic images resulted also in success, with the models being able to correctly match the features with 86.5% accuracy on three-pixel dispersion.

Chapter 2

Stitching of Digital Images

This chapter covers the fundamentals of image stitching and key point detection and introduces the common challenges in this field. The chapter starts with a brief overview of the key definitions of the most important terms. Next, it explains and classifies the image stitching methods currently used to solve this problem. Finally, the chapter discusses the metrics used to evaluate and compare the resulting image quality.

2.1 Image Stitching

The process of image stitching is comparable to putting together a puzzle. Just like a puzzle, an image is made up of various pieces that must be precisely aligned and put together to create the final picture. The objective is to transform the pixel coordinates from the composite image to each input image, allowing the corresponding region to be rendered seamlessly in the final composite [21]. Each piece corresponds to a component of the original photos.

Image stitching has a wide range of applications, such as in creating high-resolution photo-mosaics used for digital maps and satellite photos, as well as in smartphone cameras for ultra-wide-angle panoramas [28]. However, creating stitched images comes with some challenges. One common problem is unpleasant irregular boundaries, which can be addressed with a variety of techniques such as edge blending, feathering, or seam carving [18].

To create seamless composite images, alignment, and stitching algorithms are used. The alignment algorithms estimate the transformations necessary to align the images [27]. The stitching algorithms seamlessly blend the images, taking care to deal with potential problems such as blurring or ghosting caused by parallax and scene movement, as well as varying image exposures [27].

The alignment algorithms can be based on a variety of mathematical models, such as pixel-to-pixel dissimilarities or feature-based approaches that are more robust against scene movement [28]. The feature-based approaches have the added advantage of being able to recognize panoramas, automatically discover the overlap relationships among an unordered set of images, and make them ideally suited for fully automated stitching [28].

Once the images are aligned, compositing techniques are used to seamlessly cut and blend overlapping images, even in the presence of parallax, lens distortion, scene motion, and exposure differences [27]. To achieve this, global optimization techniques can be used to compute a globally consistent set of alignments and efficiently discover which images overlap

each other [28]. Finally, a compositing surface is chosen to warp the aligned images, and the blending is performed [27].

Following subsection 2.2, will expand on this topic and focus on various other image stitching challenges and current approaches to their solutions.

Figure 2.1: Example of image stitching; landscape picture of the sea

2.2 Interpretation of Image Features

As previously mentioned in section 2.1, the process of image stitching is comparable to playing with puzzles. But how does one teach the computer to be able to put the puzzle pieces together as well? The focus needs to be on how people can do it.

The correct approach is to search for specific patterns or characteristics that are distinct, traceable, and comparable. These features are searched for in the puzzle pieces, and when

discovered, the pieces are placed together using other parts that have the same features. All of these skills come naturally to people.

So the next question that arises is: What are the unique features that people are looking for? In the image below, one can see a simple blue house and three squares of different colours.

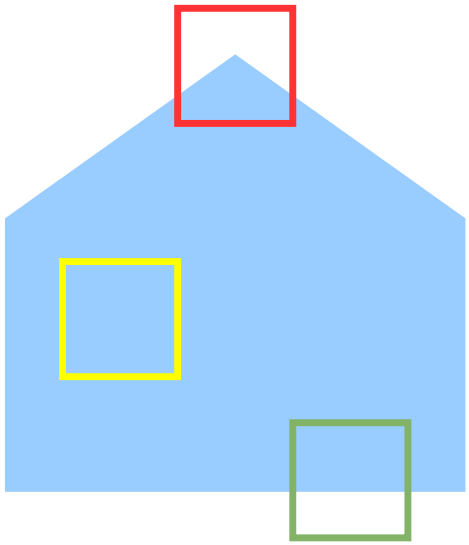
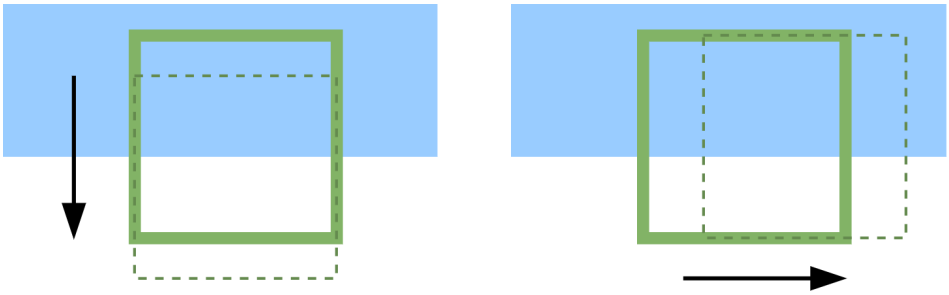


Figure 2.2: Finding features in a simple image

Determining the exact location of the squares based only on the inside of the squares varies in difficulty. The area in the yellow square would be harder to find, considering the lack of distinctive features. On the other hand, areas in the red square and the green square contain different coloured parts of the house and its background. The easiest to detect is the area in a red square because of the corner, which can be found only on the roof. Even though the area in a green square contains a line that divides the space, it is not possible to select its exact location because it can be anywhere on the edge considering the rotation of the green square.

Features are crucial in finding important regions. Distinguishable regions are detected by the colour or brightness changes [28]. These transitions can create edges, corners, or bulbs. The simplest way of thinking about detecting edges or corners is to search for regions that have the most changes when moved (by one or two pixels) in their neighbourhood [2].



(a) Region shifted down **(b)** Region shifted right

Figure 2.3: Detecting edges in the regions

An easy example of feature recognition could be found in the area of a green square (figure 2.3). This area contains an edge, transitioning between a white background and a blue house. A computer can simply detect such an edge by shifting the region within its local neighbourhood to find out whether change occurs or not. If the region is shifted right (figure 2.9b), the content of each region remains equal, meaning that there is no vertical edge. On the other hand, if the region is shifted down (figure 2.3a), the regions are now different. This indicates that the area has a horizontal edge. Moving the region in a red square will detect changes in every direction, considering that a corner is present. The area in a yellow square is not significant because moving it would not cause any changes due to the lack of features [2]. In summary, an ideal feature should be simple to locate and quick to calculate.

The process of searching for interesting regions is called *Feature Detection* (see 2.3.1). Localizing the features is one of the most important building blocks in image stitching. However, coordinates all by themselves are not sufficient. Things that make the regions interesting are not necessarily their pixel positions, but how they look. That's why it is also important to determine what colour or brightness the region is. Region-specific characteristics need to be described. A process called *Feature Description* (see 2.3.2) analyses features and extracts important information about their intensity in colour, texture patterns or edge orientation [2]. In terms of matching, these localized features are called keypoint features or interest points [28].

For understanding the following, there is a need to clarify a few terms:

- **Feature** - a general term for any visual element in an image that can be detected and used for a variety of tasks [29].
- **Interest Point** - refers to a point in an image that stands out and is likely to be useful for matching or recognition [29].
- **Keypoint** - refers specifically to a point that is distinctive, repeatable, and invariant to scale, rotation, and brightness changes [16].

2.3 Keypoint Detection and Matching Pipeline

As previously mentioned, keypoints refer to specific locations in images. They are commonly used in computer vision for image matching and comparison due to their stability across different images, even when there are changes in orientation, scale changes, or occlusion (the presence of clutter).

Finding feature points and their correspondences can be done in two ways [28]:

The first is to identify features in a single image that can be precisely monitored using a local search method like least squares or correlation. It works better when pictures are taken quickly or from close viewpoints.

The second involves finding features that are unique in each of the images considered and then matching those features based on where they appear. This method is better suited when a significant level of motion or appearance change is expected, such as when stitching together panoramas [3], creating correspondences in wide baseline stereo [24], or conducting object recognition [10].

In this section, the process of detecting and matching keypoints is divided into three distinct stages, as shown in figure 2.4. Firstly, in the **Feature Detection** stage (subsection 2.3.1), locations in each image that could match well in other images are searched. Secondly, in the **Feature Description** stage (subsection 2.3.2), descriptors that are more compact and stable are created by converting the regions surrounding detected keypoints, which can be compared to other descriptors. Lastly, in the **Feature Matching** stage (subsection 2.3.3), possible matching candidates in other images are efficiently searched for.

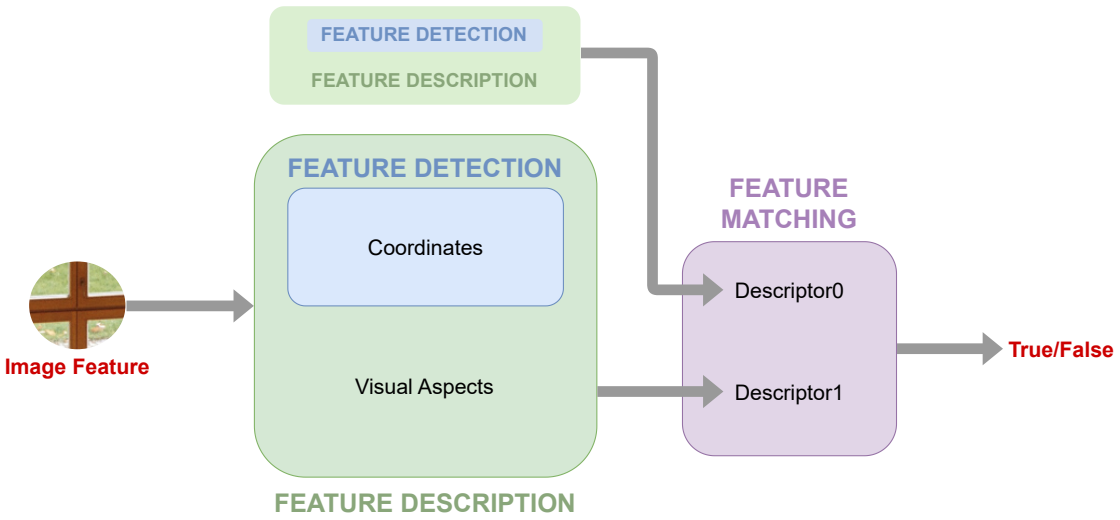


Figure 2.4: Feature Detection, Feature Description, and Feature Matching in one pipeline.

2.3.1 Feature Detection

Depending on the needs and restrictions of the application, feature detection algorithms try to locate all or a subset of keypoints in a picture. For simplicity, there are two groups of feature detection algorithms [28].

Handcrafted methods often rely on mathematical models or domain-specific heuristics to detect recognisably relevant patterns or structures. One of the most popular handcrafted feature detectors is the *Harris corner detector* [12, 19], which computes the image gradient at each pixel and estimates the local autocorrelation matrix to determine the degree of corner-like behaviour. The feature-based methods [3], which use scale-space analysis, blob detection, and binary descriptors, respectively, to find and describe keypoints, are other well-known hand-crafted feature detectors.

Learned methods utilise the power of deep learning to automatically learn feature representations that are specialised for particular tasks or datasets. These techniques typically use learned features as keypoints or descriptors and train convolutional neural networks (CNNs) on large annotated datasets. Examples of learned feature detectors include the *SuperPoint* [6] and *D2-Net* [7] methods, which use unsupervised and supervised learning approaches, respectively, to learn feature representations from raw image pixels. In this thesis, the deep learning feature detector is the SuperPoint neural network. The architecture of the net will be further explained in the next chapter 3.

Feature detection algorithms find various use cases in computer vision. It's necessary to be aware of these approaches' limitations as well. Many algorithms are sensitive to noise or lack of light, which could result in results that are unreliable. A large dataset is required to train neural networks in order to correctly detect keypoints.

2.3.2 Feature Description

It would be difficult to accurately match features if the only way to identify them was by their position. Since the physical appearance of a feature cannot be predicted by pixel position, it is crucial to specify the keypoint visuals. In the context of feature points, generated descriptors refer to the image texture or pixel grey level [15]. They are represented as vectors or matrices. Methods of feature descriptions can be divided into three types.

A method based on local gradient computes a descriptor based on an image's local area's gradient direction histogram. This method finds extreme points and describes them. Algorithms produce low-dimensional vectors containing all data on the position, scale, rotation invariance, illumination robustness, and orientation of the feature (figure 2.5). As a result of its data form, this methodology operates poorly in real-time. The most famous algorithms which use this method are *SIFT* and *SURF*. Except, SURF does not compute histograms but uses Haar wavelet transform instead [15].

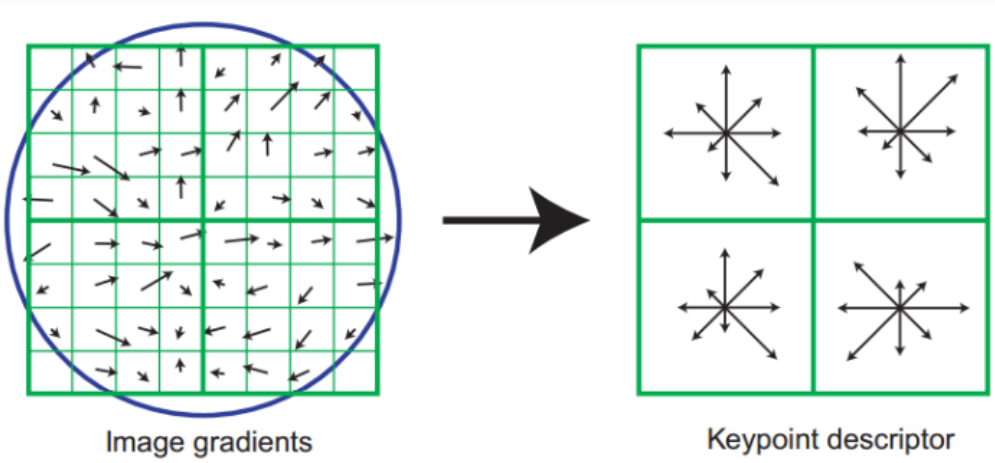


Figure 2.5: Visual representation of how gradients are computed. Image is adapted from [16].

A descriptor can be created **based on image intensity**. The most fundamental representative is *LBP descriptor*. By a combination of values in a binary string, a descriptor is generated. A binary string is created by applying a threshold[1] to the values of adjacent pixels around the feature point (figure 2.6). Later on, this approach was improved in terms of scale and rotation invariance. Although this process is quick, it isn't the best for images with dramatic changes in lighting or colour. Another popular algorithm is *BRIEF*. The method provides robustness for descriptors but does not store rotation information in them, which can lead to poor matching performance [15].

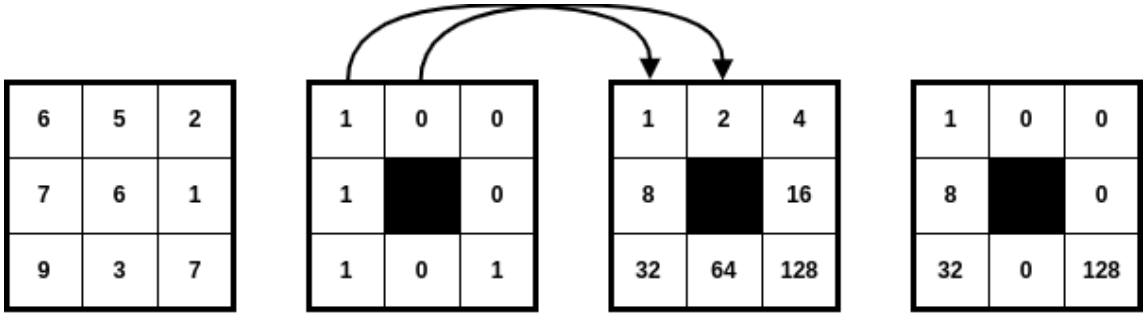


Figure 2.6: Representation of LBP algorithm of applying a threshold, which is the value of a feature, to the adjacent pixels and creating a descriptor. Inspired from [15].

Descriptors can be obtained using **learning methods**. In convolutional neural networks, each kernel detects a specific local feature such as an edge, corner, or blob. The learning-based feature descriptor is represented as a collection of features from each layer of the network. This approach is more adaptive and variable. Representatives of this approach are *SuperPoint* [6], *LeNet5* [9], *AlexNet* [14] or *VGGNet* [26].

Descriptors can be classified into two classes [29]. The **local descriptor** is computed around the feature and has a higher level of detail. They represent the colour, shape, scale, rotation, and illumination of features. They are better for image matching and object recognition. The second class, **global descriptors**, capture the image's visual characteristic

It can be difficult to find ideal descriptors, but each of these methods has proved to be effective. A feature description plays a crucial role in keypoint matching.

[1] The value of a feature point.

Scale-Invariant Feature Transform (SIFT)

The images in the figure collection below 2.7 depict the same object captured from different angles and with varying foreground and background objects:

Figure 2.7: Images showing the Eiffel Tower with a different background, captured from different angles, and also include different objects in the foreground (in some cases). Image is adapted from [13].

It is easy for humans to identify the object, in this case, the Eiffel Tower, regardless of changes in angle or scale. However, machines struggle with this task, making it a challenge to identify objects in images when those certain elements are changed. Nonetheless, machines can be trained to identify images at a human-like level, using a computer vision approach.

The *Scale-Invariant Feature Transform* (SIFT) is a feature detection algorithm proposed by David Lowe in his paper "Distinctive Image Features from Scale-Invariant Keypoints" [16]. SIFT detects feature points in an image that are invariant to scale, rotation, and translation as compared to other rotation-invariant corner detector methods like Harris [12].

SIFT identifies feature points based on the following steps:

1. **Scale-space extrema detection:** The *scale-space extrema detection* step in SIFT involves creating a *scale space* $L(x, y, \sigma)$ of the input image, where (x, y) are the *image coordinates* and σ is the *scale parameter*. The scale space is created by convolving the input image $I(x, y)$ with Gaussian filters (see figure 2.8) of increasing scale σ:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.1}$$

where $G(x, y, \sigma)$ is the Gaussian function:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)} \tag{2.2}$$

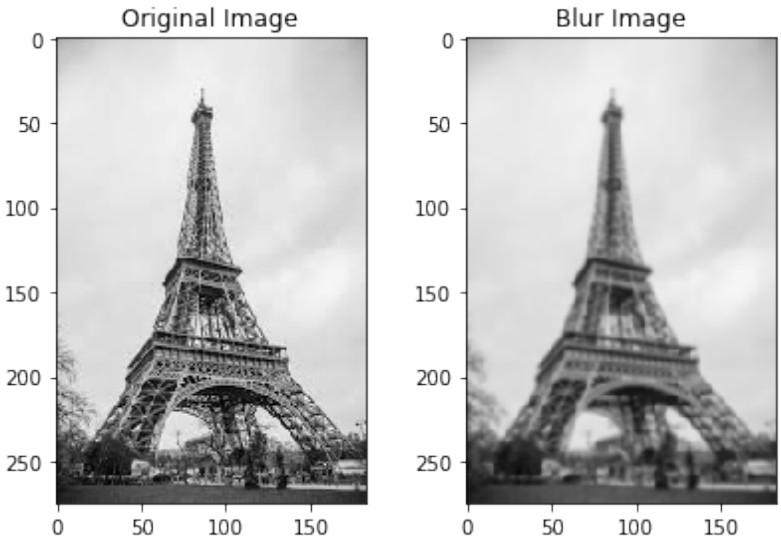
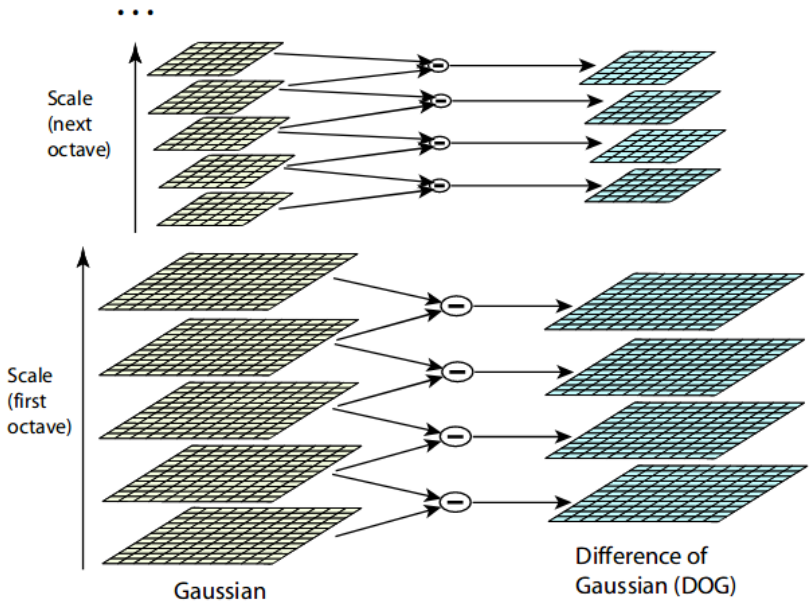


Figure 2.8: Example of an image before and after applying the Gaussian Blur. The texture and minor details had been removed from the image, and only relevant features, like the shape and edges, remain. Adapted from [13].

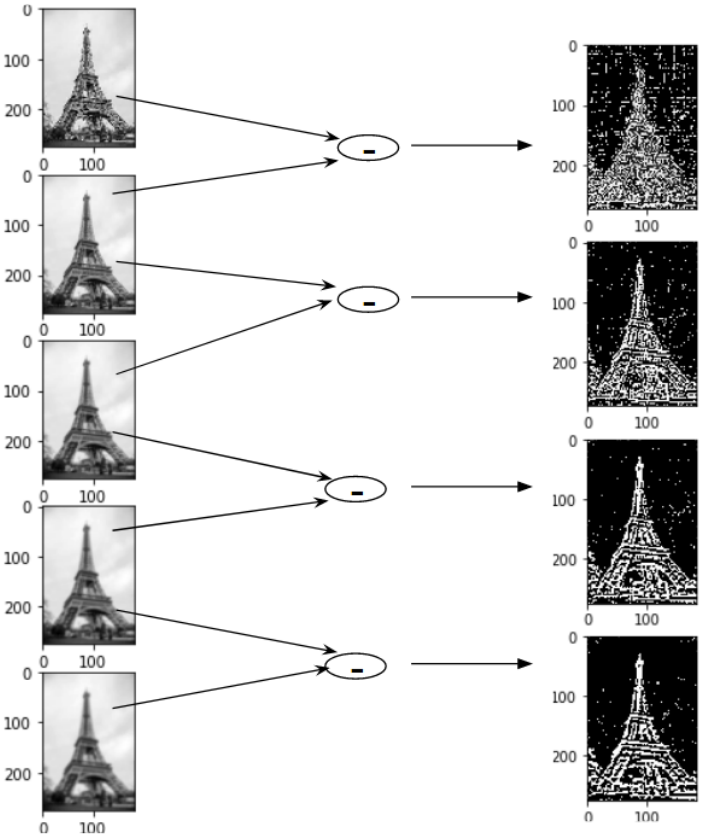
Next, the features need to be enhanced. SIFT searches the scale space for local extrema in both scale and space using *difference-of-Gaussian* (DoG) images. The DoG creates another set of images (see figure 2.9), by subtracting every image from the previous image, in the same scale (subtracting adjacent scales in the scale space):

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{2.3}$$

where k is a constant that determines the ratio between adjacent scales. The extrema in the DoG images correspond to potential keypoints.



(a) Visual explanation of how DoG is implemented in a vertical form for a clearer view. The image was taken from the original paper [16].



(b) Five subsequent images, on the left, were created by applying the Gaussian blur over the previous image. On the right, four images were generated by subtracting the consecutive Gaussians. Adapted from [13].

Figure 2.9: DoG, the feature enhancement algorithm, for the images in scale space.

Once this DoG is found, images are searched for local extrema over scale and space. To locate the local maxima and minima, SIFT goes through every pixel in the image and compares it with its neighbouring pixels. Simply, one pixel in an image is compared with its 8 neighbours, as well as 9 pixels in the next scale and 9 pixels in previous scales (see figure 2.10). If it is a local extremum, it is a potential keypoint (keypoint is best represented in that scale).

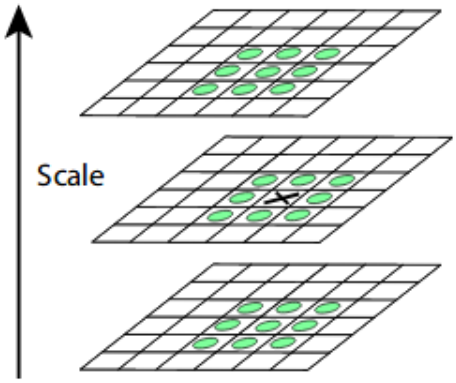


Figure 2.10: Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3×3 regions at the current and adjacent scales (marked with circles). The image was obtained from the original paper [16].

2. **Keypoint localization:** The potential keypoints are filtered out based on their stability under varying image conditions, such as *rotation* and *scaling*.

   The purpose of this step is to eliminate unstable keypoints that are sensitive to small changes in scale, rotation, or translation. To do this, SIFT fits a three-dimensional quadratic function $D(\mathbf{x})$ to each potential keypoint in the scale space using the Taylor expansion of scale-space extrema. The function is given by:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{2.4}$$

   where $\mathbf{x} = (x, y, \sigma)$ is the location and scale of the potential keypoint, D is the scale-space value of the potential keypoint, and $\frac{\partial D}{\partial \mathbf{x}}$ and $\frac{\partial^2 D}{\partial \mathbf{x}^2}$ are its first and second partial derivatives with respect to $\mathbf{x}$.

   After fitting the quadratic function, SIFT refines the locations of keypoints to get more accurate results. If the intensity of extrema is less than a threshold value (0.03 as per the paper [16]), it is rejected. DoG has a higher response for edges, so edges also need to be removed. Simply put, SIFT eliminates any low-contrast keypoints and edge keypoints and what remains are strong interest points.

3. **Orientation assignment:** An *orientation* is assigned to each keypoint based on the gradient direction of nearby image pixels. The orientation is used to make SIFT *invariant* to image rotation.

   To assign an orientation to a keypoint, SIFT first computes the *gradient magnitude* and orientation at each pixel in its neighbourhood. The orientation of the keypoint is then set to the dominant orientation of the gradient orientations in a histogram of 36 bins covering 360 degrees. The histogram is weighted by the gradient magnitudes and is smoothed with a Gaussian function to reduce the influence of noise and local variations.

4. **Descriptor generation:** SIFT generates a descriptor vector for each keypoint that represents its local image structure. The descriptor is based on the distribution of image gradients in the keypoint's 16×16 neighbourhood and is invariant to image translation, rotation, and scaling.

   SIFT computes the gradient magnitude and orientation in 16 subregions of the keypoint's neighbourhood, each with a size of 4×4 pixels (see figure 2.11). For each subregion, a histogram of gradient orientations with 8 bins covering 360 degrees is computed. The orientation histograms of all 16 subregions are concatenated to form a 128-dimensional descriptor vector. The descriptor is then normalized to reduce the influence of changes in illumination and contrast.

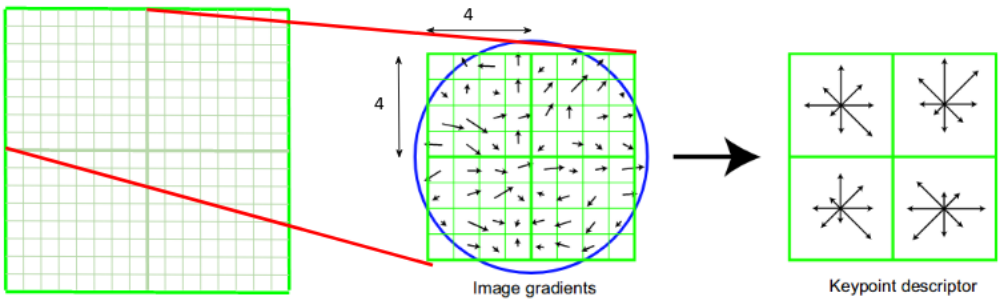
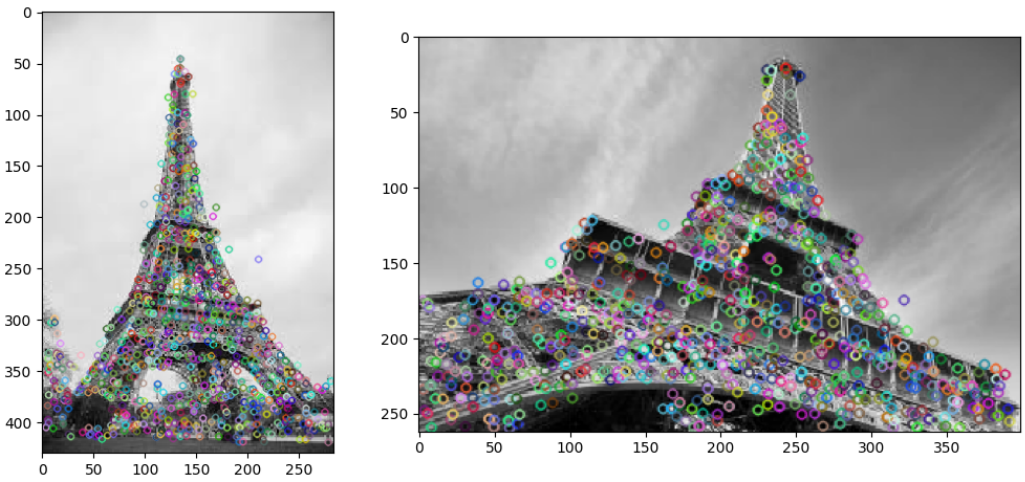


Figure 2.11: The process of creating a feature descriptor from neighbouring pixels, their orientation, and their magnitude. Adapted from [13].

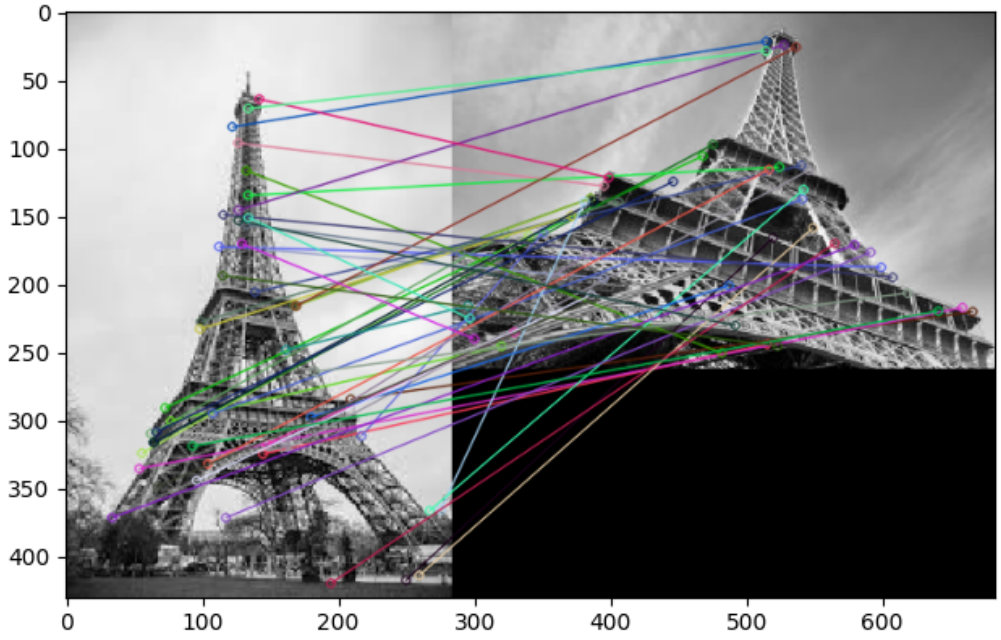
The final result is a set of keypoints, each with a corresponding descriptor vector, that represents the local structure of the input image in a scale, rotation, and translation invariant way. SIFT descriptors are robust to changes in lighting, partial occlusion, and viewpoint changes, which makes them suitable for object recognition and image-matching tasks.

SIFT has been widely used in various computer vision applications and has been shown to achieve state-of-the-art performance in many benchmarks [16]. However, SIFT is computationally expensive, and faster algorithms such as SURF and ORB have been proposed as alternatives.

In the figure 2.12 keypoints are detected using SIFT method. In each of these images, SIFT found approximately 1000 keypoints in various places. Later on, the OpenCV method *Brute Force Matcher* is applied to match detected keypoints by using obtained descriptors. The matcher found approximately 250 corresponding keypoints within the images. The process of feature matching will be discussed in the following section 2.3.3.



(a) Detected keypoints of images using SIFT `detectAndCompute` method.



(b) Matches between keypoints images using `BFMatcher` from OpenCV library.

Figure 2.12: Detection of keypoints and their matching between images using SIFT.

2.3.3 Feature Matching

After extracting descriptors from a pair of images, the next step is matching them. There can be two strategies for matching features. One is in the context of image stitching, when it is known that two images are supposed to overlap. The second is when images have no correspondence whatsoever, for instance when matching objects from a database [28].

Picking a strategy for matching plays an important role in how good the matches are. It is assumed that descriptors were designated, so the *Euclidean distances* can be used for match ranking. The simplest strategy is to set a threshold for this distance. Providing a perfect threshold is sometimes difficult, so a better strategy would be matching the descriptors to the nearest neighbour in a feature space [28], but this process can be computationally challenging.

A more efficient approach is to use an indexing structure[2], such as a multidimensional search tree or a hash table.

Verification of those potential matches is done by translating, rotating or affining the image to determine which feature matches are consistent and should be kept.

Deep learning has significantly improved matching. Neural networks such as convolutional neural networks and graph neural networks are effective tools for feature extraction and matching. An example of feature matching using deep learning is *SuperGlue* which will be explained in chapter 4.

One of the last steps of simple image stitching without post-processing is applying a mathematical transformation called *Homography* to one of the images.

[2]Simple technique is multidimensional hashing. More complex are locality-sensitive hashing, parameter-sensitive hashing or multidimensional search trees.

Homography

After finding and determining the correspondence of keypoints and descriptors between two images, the images are aligned using *homography* [1, 5]. Homography is a mathematical transformation that maps the points in one image onto the corresponding points in another image. In computer vision, it is used to align two images of the same scene taken from different viewpoints or under different conditions, as it can be used to calculate the necessary rotation and translation of an image to align it.

A homography can be represented by a 3×3 matrix H as follows:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.5}$$

where (x, y) and (x', y') are the coordinates of the corresponding points in the two images.

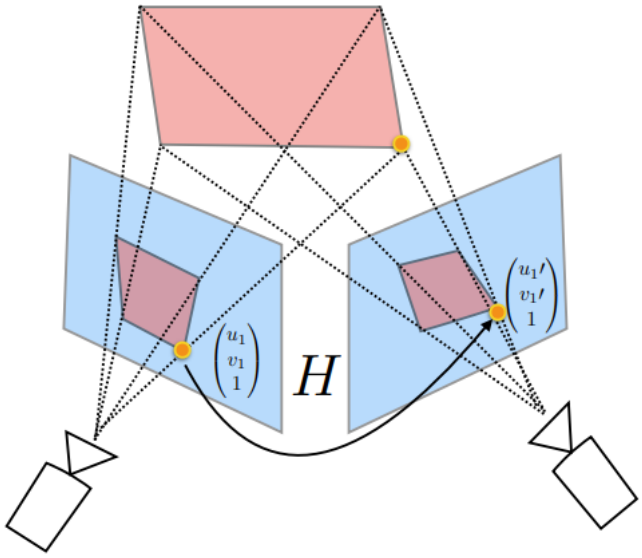


Figure 2.13: Simple representation of mapping two points from different images. The image was adapted from the original paper [5].

Because computing homography strictly depends on provided matches, it is necessary to get the best matches possible. To enhance the process of matching, the homography matrix can be estimated using the RANSAC. Given N pairs of corresponding points, the RANSAC algorithm estimates the homography matrix H by randomly selecting a subset of points and then finding the best fit to these points using a least-squares approach. Once the homography matrix H is computed, it can be used to warp one image onto the coordinate frame of the other image.

Homography is particularly useful in image stitching, where multiple images are combined into a panoramic view, and in object recognition and tracking, where the appearance of an object may vary due to changes in viewing angle or lighting conditions.

RANSAC

In order to handle a high percentage of outliers in the input data, Fischler and Bolles came up with RANDOM Sample Consensus (RANSAC) algorithm [11]. RANSAC algorithm is a statistical method that categorises *inliners* and *outlines*[3] and identifies the best set of matches that are most suitable for computing geometric transformations between two images [28].

RANSAC algorithm randomly selects a subset of points called a "minimal sample" to determine the model parameters. The algorithm then evaluates the quality of the model by measuring how many points from the set of all points fit within a predefined tolerance. If the fraction of inliers exceeds a predefined threshold (see figure 2.14), the algorithm re-estimates the model parameters using all identified inliers and terminates. Otherwise, the algorithm repeats these steps a maximum of N times.

So unlike other sampling techniques that use as much data as possible to get an initial solution and then remove outliers, RANSAC starts with the smallest set possible and adds consistent data points to it to make the set larger. This makes RANSAC more robust to outliers and able to produce good results even in noisy data.

The number of iterations N [4], is chosen high enough to ensure that the probability p (usually set to 0.99) that at least one of the sets of random samples do not include an outlier. Let u represent the probability that any selected data point is an inlier and $v = 1 - u$ the probability of observing an outlier. N iterations of the minimum number of points denoted m are required, where

$$1 - p = (1 - u^m)^N \tag{2.6}$$

and therefore,

$$N = \frac{\log(1-p)}{\log\left(1-(1-v)^m\right)} \tag{2.7}$$

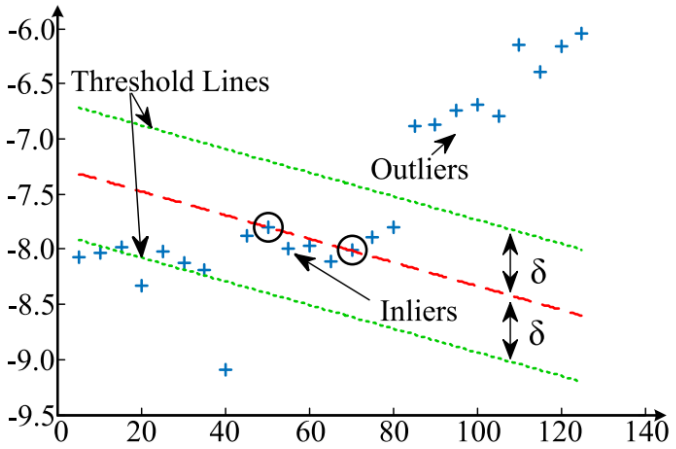


Figure 2.14: Illustration of the threshold value determined by RANSAC algorithm to detect outliers. The image was adapted from the original paper [8].

[3] Good and bad matches.

Chapter 3

Deep Neural Networks and Image Stitching

This chapter presents a basic overview of deep neural networks and their application in image stitching. The training process of the neural network will be explained, as well as different techniques to train the models. Finally, the selected deep learning methods will be explained and discussed.

3.1 Overview of Deep Neural Networks

The deep neural networks [28] are *feedforward* computation structures which contain *nodes* and *weights* in the *hidden layers*. The nodes or *neurons* perform a weighted sum of the input data. The weighted sum equation uses additional value *bias*, which together with weights are learnable parameters (see 3.1). Equation:

$$s_i = \mathbf{w}_i^T \mathbf{x}_i + b_i \tag{3.1}$$

where s_i represents output data, $\mathbf{w}_i^T$ represents learnable weights, $\mathbf{x}_i$ the input data and b_i the bias. To guarantee a non-linearity to the computation, *activation functions* are used.

The activation function based on the computed value determines whether the value is passed to the next layer or stashed. The most common activation functions and the functions used in this thesis are *ReLu LeakyRelu* and *Sigmoid*. Both ReLu and LeakyReLu operate in the same way, as they let pass the positive values, but ReLu stashes every computation below zero. The LeakyReLu lets a one-hundredth of the negative value pass. The Sigmoid function

After the output value is computed, the difference between the expected and predicted value is computed is a *forwardpass*. To compute the difference, various functions are used and are collectively called *loss functions*. One of the most common loss functions are *Cross-Entropy* function or the *MSE* function. The used loss function in all experiments in this thesis is the MSE (see 3.2). It computes a mean squared difference between the outcome and true value.

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3.2}$$

The computed value is used to change weights and biases in order to obtain more accurate predictions. The process of computing the adjustment value is called *backpropagation* and happens during *backwardpass*.

The value of change is adjusted by the *learning rate*. The learning rate is the rate at which the model is learned, meaning that only part of the computed difference is used to change weights. This update is done by an element called *optimizer*.

The basic concept of the neural network and the logic behind the update of the model was explained, and now it is important to mention the ways in which the model can be trained.

The two main types of learning are *Supervised Learning* and *Unsupervised Learning*. During supervision learning, the model obtains true values based on which it computes the loss function. Unsupervised learning does not provide the true values, and the model is learning the pattern by itself.

The process of training can be enhanced by various *regularisation techniques* to improve the accuracy of the model and to prevent the model from *overfitting*, which is a state when the model stopped being generalised and started to adapt to training data too much. Techniques such as *Dropout*, which with certain probability stashes part of the neurons and *Batch Normalisation* which normalises the input data can prevent a model from overfitting.

Although there is a lot to mention about neural networks, it is essential to explain how are neural networks useful in terms of computer vision and image stitching.

3.2 Deep Learning Approaches of Image Stitching

There are various approaches which are suitable for each step of the stitching pipeline. The most famous implementations were mentioned in the previous chapter.

3.2.1 Selected Methods

This section provides an overview of selected models in this thesis. The chosen approach for feature detection and description is the *SuperPoint* model, and for feature matching the *SuperGlue* model. All the provided information is extracted from original papers and a review of the implementation.

SuperPoint

Superpoint [6] is a deep, fully *Convolutional Neural Network*. It produces SIFT-like two-dimensional interest point locations and descriptors in a single forward pass. The neural network or the detector pathway is superior to standard corner detectors such as FAST and Harris Corner for recognising features. Even in noisy images, it is able to detect features with over 97% accuracy, while traditional methods achieve a maximum of nearly 70%.

To achieve this performance, the SuperPoint was at first trained with a self-supervised method. The *Magic Point* or the detector pathway was trained on synthetic data shapes without labels with the help of *Homography Adaptation*. The synthetic data consisted of simple shapes such as triangles, lines, and ellipses. Homography Adaptation allowed SuperPoint to notice more keypoints because of random homography sampling on an image during a training phase. This process involves creating wrapped images on which key points are detected, warped back to the original plane and combined to generate a superset of keypoints for one image.

The architecture of SuperPoint consists of a shared encoder for both detection and description and separate decoders. All the layers in each part of the net are convolutional layers.

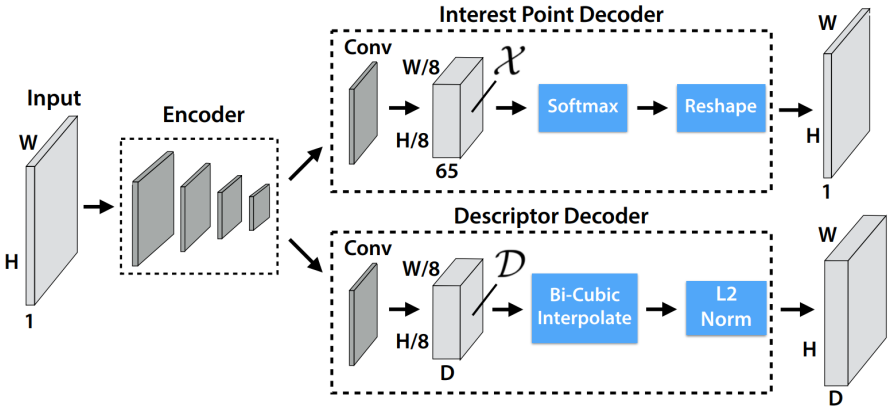


Figure 3.1: This image shows an architecture of the SuperPoint neural network. The image was adapted from the original paper [6].

Convolutional neural networks [28] are superior to nearly all tasks involving image processing and analysing. Convolutional layers contain *filters* which are able to extract the most prominent information about the input data. The filters, also known as kernels or feature detectors, are small matrices of numbers. These matrixes slide over the image and compute a *dot product*[1] of the pixel value and filter weights (figure 3.2). The output of these dot products makes a feature map.

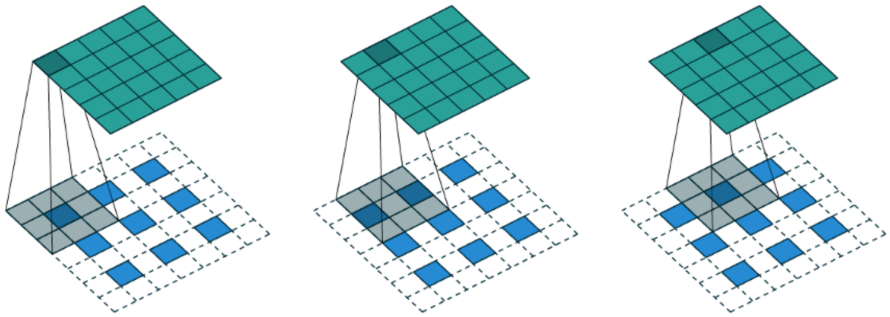


Figure 3.2: This image shows an architecture of the SuperPoint neural network. The image was adapted from the original paper [28].

After extracting the keypoints many of them are eliminated due to post-processing conditions. The two post-processing methods involve *Non-maximum suppression* (NMS) and applying a threshold for prediction. The NMS is a technique which eliminates redundant detections by selecting the pixel with the highest prediction value.

The SuperPoint provides accurate predictions, and its keypoints and descriptors can be further used in an image-stitching pipeline.

[1]The mathematical operation of multiplying two vectors of equal length and summing the values to one final value.

SuperGlue

SuperGlue [23] is a neural network suited for matching two sets of local features. The set of local features contains keypoint positions and descriptors. Assignments are created by solving a *differentiable optimal transport problem*[2] whose costs are predicted by a *graph neural network*. The implementation uses *self-attention* and *cross-attention* (figure 3.4) to get an advantage of the spatial relationship of the keypoints and their visual appearance.

The architecture of the neural network consists of *Attentional Graph Neural Network* and *Optimal Matching Layer* (figure 3.3).

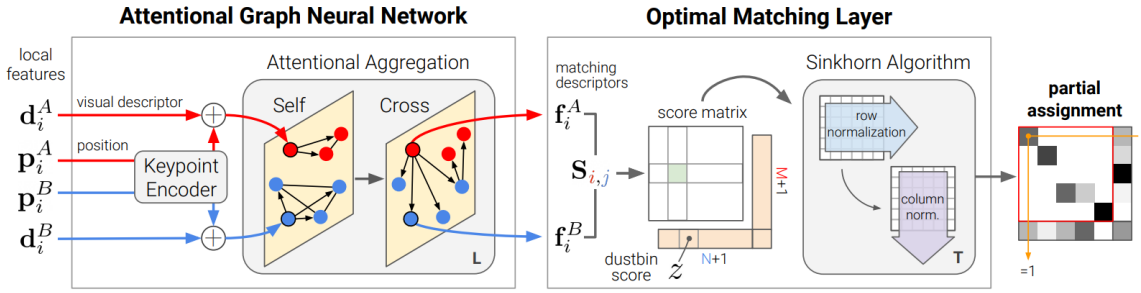


Figure 3.3: This image shows an architecture of SuperGlue implementation. The image was adapted from the original paper [23].

To obtain cost predictions by the **Attentional Graph Neural Network**, the input features are altered. In other words, descriptors are enhanced by the normalised keypoint's location and their probability of being a keypoint. The new keypoint representation is used as a node in the graph. This graph or *multiplex graph* has two undirected edges, which represent self or intra-image edges and cross or inter-image edges.

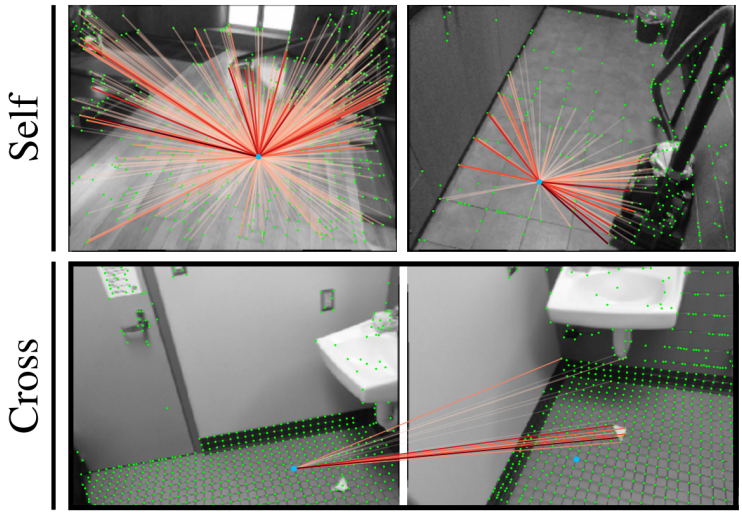


Figure 3.4: This image shows an example of self-attention and cross-attention. The SuperGlue can focus on global and local context, self-similarities, distinctive features or potential matches, The image was adapted from the original paper [23].

[2]The problem dwells in finding the most effective solution for a given problem.

The self edges are connected between keypoints within the same image, and cross edges represent a relationship with keypoints from the other image.

The graph neural network starts with a high-dimensional state for each node and updates its representation at each layer by continuously aggregating information over all nodes. The number of layers is static, and each layer has its own projection parameters which can be learned. To explain this on a higher level, each node updates its edges based on aggregated information separately for self and cross edges on each layer. The node's self-edges are updated on the odd layer and cross edges on the even layer. This way, the nodes obtain a robust representation of its characteristic in their own image and a second image.

The *Attentional Aggregation* is a mechanism, which performs aggregation and computes a message. This message is used to update the representation of nodes in the graph separately on for each self and cross edge.

The second block of SuperGlue **Optimal Matching Layer** is responsible for producing the partial assignment matrix between the keypoints of two images. The score prediction is expressed as the similarity of matching new descriptors. Each keypoint from one image is assigned to a single keypoint in another image or to a *dustbin* which explicitly states that keypoint does not have a match.

The score prediction and expected keypoint assignments are used to solve the optimal transport problem. The implementation uses *Sinkhorn Algorithm* to solve the optimal transport problem with keypoint assignments and score predictions, which for T iterations normalises the score predictions along rows and columns. After the iterations, the dustbins are dropped and the assignment for keypoint in another image is recovered.

By provided evaluation metrics, SuperGlue as the middle-end matcher together with SuperPoint as the front-end detector and descriptor surpass the handcrafted approaches.

Chapter 4

Proposed Solutions for Image Stitching

4.1 Definition of the Task

The main goal of this thesis is to get acquainted with image stitching and deep learning. This includes experimenting with various approaches, methods, and implementations. There are various implementations used for keypoint detection and matching between key points (see 3) as well as estimating homography. The product of this work is an image-stitching pipeline using deep learning methods. The pipeline is suitable for stitching more than two images together.

4.2 Used Datasets

In this section, it will be discussed which datasets were used to accomplish the goal. The first part of the section will introduce some basic image-processing techniques such as rotation, noise injection, and brightness adjustments. The dataset generated with the help of these techniques is essential for evaluating the chosen models. The second part of this section will cover the introduction to another dataset with microscopic images. This dataset is used for the final product, which uses a deep-learning model to stitch not only pairs of images but a grid of images from the microscope.

4.2.1 Generated dataset with image processing techniques

In the earliest stages of implementation, ten basic images of random objects and sceneries were used. These images underwent some transformations in their appearance. There are lots of image processing techniques to make datasets more robust. Simple methods like cropping, rotating, or colour and brightness adjustment enhance the power of a dataset. Each of these techniques adds uniqueness to the image. Augmenting input images in lots of sets can increase input features to neural networks multiple times. The effectiveness of augmenting images in the process of training neural networks is undeniable [25]. This dataset will be during the thesis, called **dataset A**.

The most common augmentation techniques for image processing are brightness, adding noise, or rotating the image, and all of these methods will be applied to the images to enhance the dataset. The selection of parameters for each one of the methods is selected randomly to add robustness to the dataset (figure 4.1).

Figure 4.1: Noise injection, brightness enhancement, rotation.

The dataset should be suited for the image stitching process, which is why input images are split in approximately, which generated a synthetic pair of images (figure 4.2).

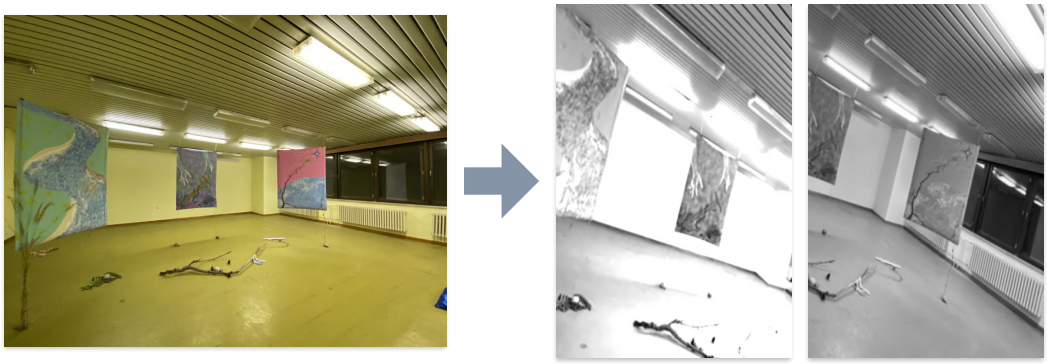


Figure 4.2: Generated pair from an image using all mentioned augmenting techniques.

The augmentation process is done separately on these images to create bigger diversity between images. To accelerate the production of the dataset, each pair from the set is augmented several times.

Numerous techniques for cropping additional background after rotating didn't produce adequate results. Either they returned a low-quality image or they improperly cropped the image. To prevent misleading key points on the edge between the black background and the image, a high-quality crop is needed.

The chosen approach is rather simple. It searches in pairs, left-top and right-bottom, depending on the rotation angle. In consideration of triangular similarity, it is sufficient to identify one pair in order to calculate the other pair.

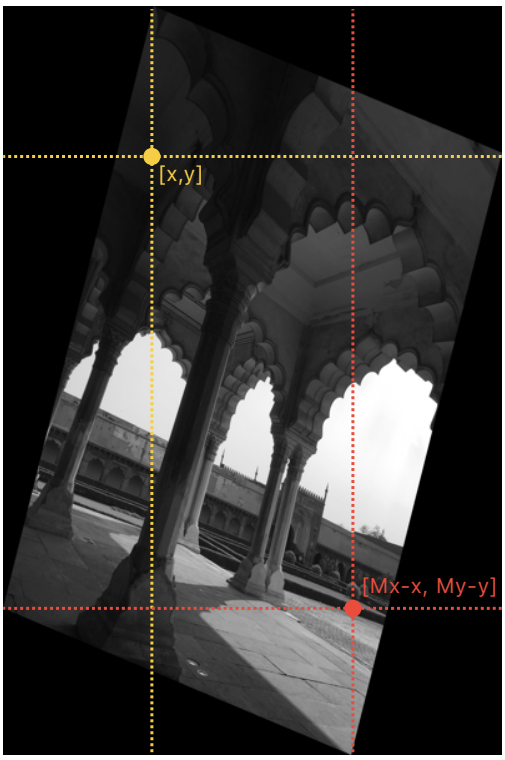


Figure 4.3: Points detecting.

In the image 4.3 are four lines, each describing one of the four borders. Considering the negative angle of rotation, the yellow lines represent the left and top borders. `Mx` is the maximum `x` and `My` is the maximum `y` coordinate. Searching for the first non-black pixel for the left border starts in the first row and continues until the pixel is not found. The starting point of the search for a top border is at the last column, and continues until the first non-black pixel is found.

By founding two of the four borders, it is easy to determine the rest. This method produces a cropped image which is sufficient and does not cause additional noise or worsen the quality of an image. This technique has limitations in not being able to crop an image if the image has black background without rotation.

The rotation range is from -15 to 15 degrees.

4.2.2 Microscopic images

The second dataset contains microscopic images. TESCAN[1] supplied a subset of images for this dataset that was specially selected for the image stitching problem. The rest of the images were found from different sources.

In comparison to the previous synthetic dataset, the **TESCAN dataset** presents a more challenging image-stitching problem since it includes microscope images with grid sizes of 2×2 and 8×3, some of which may have complex geometries and patterns that need to be precisely aligned for a seamless panorama.

The folder has a simple tree structure 4.4.

```
VolumeEM-Image_stitching_sample_data_202209
├── 2x2
│   ├── dataset3_long_g0001_t0000_s00010.tif
│   └── ...
└── 8x3
    ├── 2022-07-15_brain_dataset_g0000_t0000_s00010.tif
    └── ...
```

Figure 4.4: Tree structure of folder.

A total of thirteen images from the microscope are in folder `2x2`. Each image is separated into four individual pieces, as the name suggests (figure 4.5).

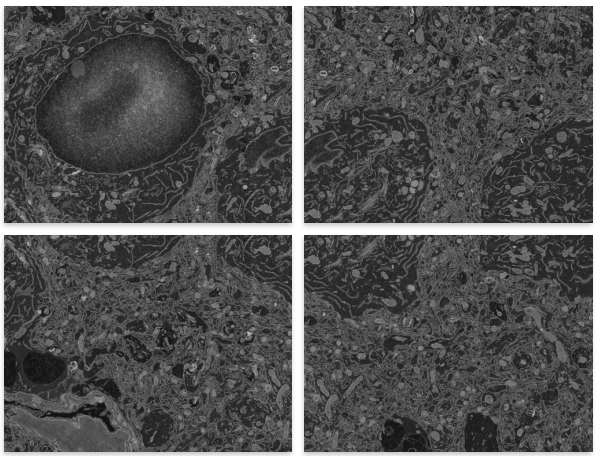


Figure 4.5: Images tiles in 2×2 folder.

The name of the images indicates a position in the grid. The top-left tile has in name `t0000`, the top-right tile has `t0001`, the bottom-left tile has `t0002`, and the bottom-right tile has `t0003`. These four pieces make up each image. Another index was used to differentiate between the images. Each image has a `s0xxxx` suffix in its name, which is used to determine individual images. As a result, a set of four tiles can be used to represent each image in the first folder.

[1] see https://www.tescan.cz/

Folder 8×3 consists of five images in total. Each image is divided into twenty-four tiles which complete the grid (figure 4.6).

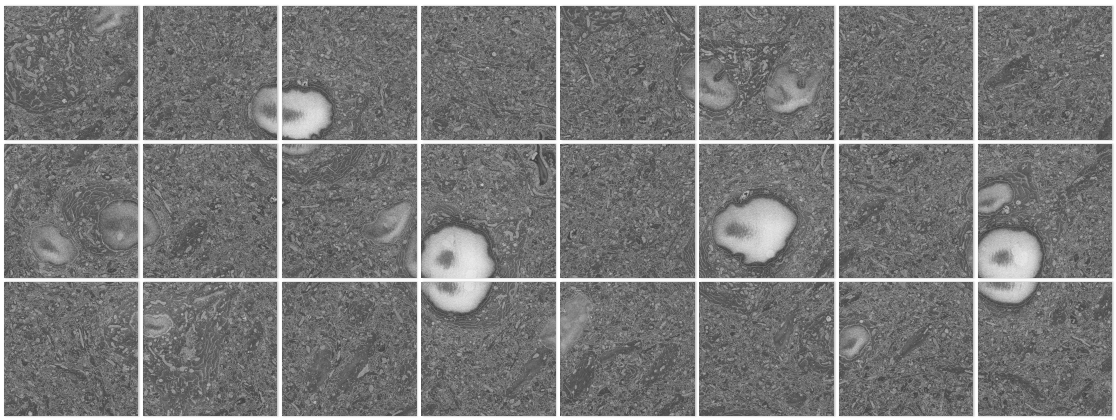


Figure 4.6: Images tiles in `8x3` folder

Each corresponding image overlaps only about 10% of the image.

For fine-tuning models diverse images from Image Data Resource [30] were selected[2]. Image data resource contains lots of microscopic images which are public and can be used in personal projects. The final size of the dataset is x images. The baseline of this dataset will be during thesis called **dataset B**.

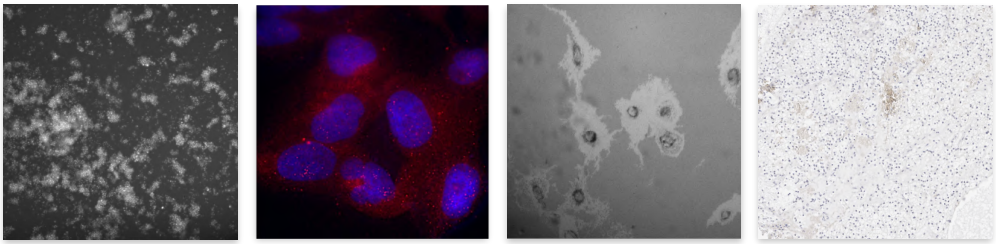


Figure 4.7: Images from Image Data Resource.

As it was mentioned earlier, the cropping algorithm was not performing well on images with a black background. That is why for microscopic images, the cropping algorithm was altered. The top border is calculated with an equation 4.1:

$$\overline{y} = \left\lceil \frac{|angle * y|}{90} \right\rceil \tag{4.1}$$

where y represents original coordinate, $\overline{y}$ new coordinate, $angle$ angle of rotation. The bottom border is calculated by subtracting the top from the height of the image. The left border is located at two times the top, and the right border is created by subtracting the left from the width of the image. The equation was deduced from triangle similarity rules.

[2]see https://idr.openmicroscopy.org/

4.3 Generating dataset for keypoint detection

As it was mentioned in the previous section, the process of training and validating the models would be done with microscopic images. Because training requires a proper and robust dataset, images must be augmented and processed. For each image in dataset B, SuperPoint's keypoint locations were extracted and used. These locations are represented with coordinates x and y. Around coordinates a 64×64 area is selected to get a close-up of the feature. The images which contained less than 20% brightness and more than 60% were eliminated due to wanting only to keep images with visible differences in brightness (figure 4.8). Images contained various selected features with different shapes. It contains edges, corners and blobs. The final dataset named **dataset C** consisted of approximately 150,000 images.

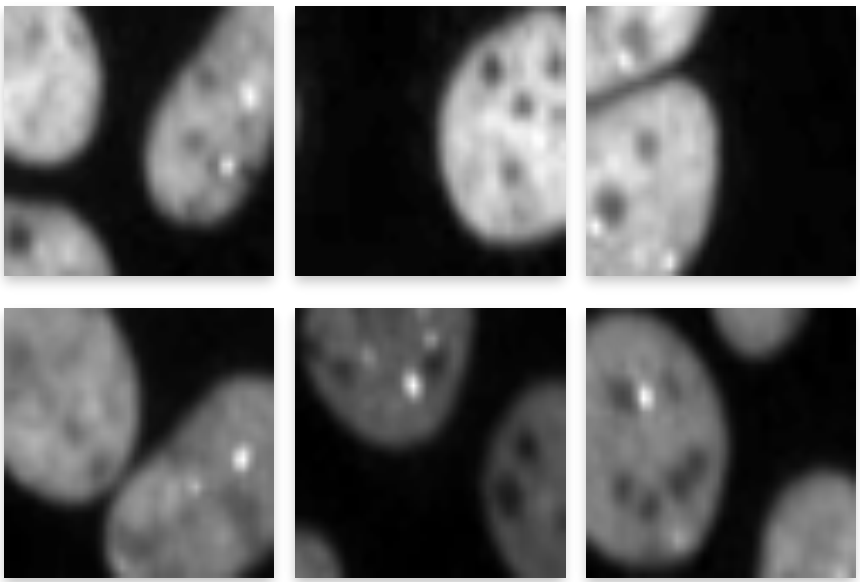


Figure 4.8: Example of the images in the dataset with the most noticeable shapes.

4.4 Generating dataset for homography estimation

To experiment with the creation of homography matrixes, the dataset will contain pair of images generated from dataset B. These pairs of images will be assigned a label which will be a matrix. This matrix is generated by using SuperPoint and SuperGlue to obtain matches between keypoints. The homography matrix will be computed with the traditional OpenCV `findHomography`[3] function with the help of RANSAC. The images were resized to 128×128 resolution as it is mentioned in the paper. During labelling the images, around 6% of paired images did not obtain a homography matrix so they were excluded. This dataset will be during thesis called **dataset D**. It consisted of x images.

4.5 Proposed models

This section provides an overview of two neural networks that were used to experiment with image stitching pipeline. A first neural network is a convolutional network which detects features on an image 4.5.1. A second model will be trained to estimate the homography matrix 4.5.2. Both of these models will be tested and evaluated in chapter 5.

4.5.1 Keypoint detection neural network

Because keypoint detection is one of the most crucial steps in image stitching, an appropriate detector is needed. The purpose of this model is to get familiar with feature detection. This neural network was trained unsupervised with no labels. The architecture of the neural network consists of an encoder and a decoder with a sigmoid on the output layer.

The model is simple, the encoder contains three convolutional layers and the decoder three transpose convolutional layers. The Dropout layer with 0.2 probability is between the encoder and decoder. The activation function for this neural network is *LeakyReLu* and at the output, there is a *Sigmoid* function (figure 4.9).

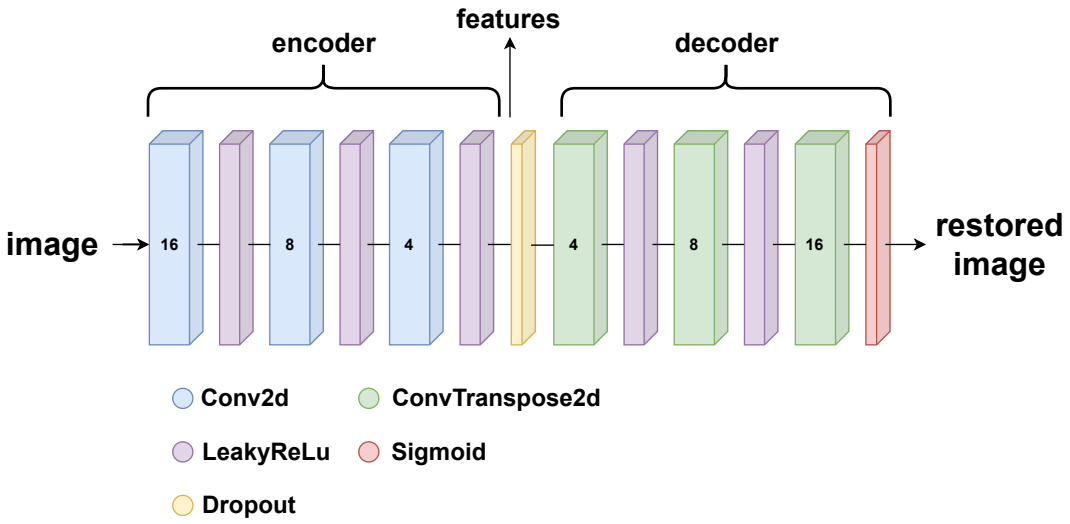


Figure 4.9: Architecture of Autoencoder neural network which is used for unsupervised learning.

[3] see https://docs.opencv.org/3.4/d1/de0/tutorial_py_feature_homography.html

4.5.2 Homography estimation neural network

The second model that was proposed is a neural network for homography matrix estimation. This model is trained to compute the homography matrix between two images The authors of SuperGlue and SuperPoint proposed an architecture for homography estimation [5].

This model consists of convolutional layers and produces a 4-point parameterization of the homography. The architecture of this model will be adapted with a change in an output layer. The proposed model will be producing a classic 3×3 matrix.

During training, problems occurred with the architecture not working as it should. The input to the neural network was two greyscale images which are related by a homography. Input to the neural network is represented with $batchsize \times 2 \times 128 \times 128$. After eight convolutional layers, it was impossible to obtain their requested size for connecting to a linear layer. That is why the architecture was altered to match 128×128 images. The neural network consisted of four convolutional layers. The linear layers were also resized.

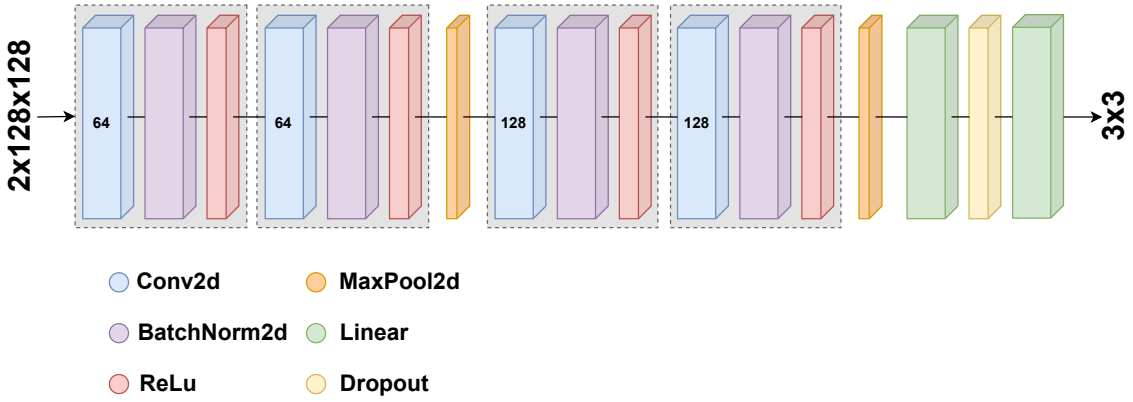


Figure 4.10: Architecture of a neural network suited for predicting homography matrix between images.

This chapter provided an overview of used datasets as well as the process of generating synthetic images for training the neural network. The purpose of each dataset is to evaluate the performance of selected approaches.

	size	purpose
dataset A	10	SuperPoint and SuperGlue accuracy
dataset B	3000	baseline of dataset C and D
dataset C	150,000	training the neural network on feature detection
dataset D	2000	training the neural network on homography estimation
dataset TESCAN	18	final stitching product

Table 4.1: Size and purpose of all datasets used in this thesis.

Two architectures of neural networks which were used in order to understand image-stitching pipeline in more depth were introduced. The first neural network is suited for detecting features in images which contain selected features detected with SuperPoint neural network in microscopic images. The second neural network is for estimating homography between two images.

Chapter 5

Experiments and results

This chapter provides an overview of experiments and their results. Firstly, it will be described how the proposed models were trained and what were their results. The SuperPoint and SuperGlue models will be compared to the traditional image stitching pipeline from OpenCV. The models will be fine-tuned on microscopic data and evaluated if their accuracy changes. Finally, an image stitching algorithm for grid stitching will be explained and evaluated.

5.1 Evaluation Metrics

In this section, chosen evaluation metrics will be discussed and explained. The evaluation metrics for unsupervised learning [20] used in this thesis are:

- **Calinski Harabasz Score** also known as *Variance Ratio Criterion* computes a ratio of the sum of between-clusters dispersion and within-cluster dispersion. This score does not have a specific range but the higher the score the better the clustering.
- **Silhouette Score** is computed for each sample using the *mean intra-cluster distance*[1] and the *mean nearest-cluster distance*[2]. The range for this score is [-1,1] where 1 represents a high separation of clusters.

All of these scores are using clusters generated by *KMeans* clustering method[3]. This method iteratively assigns data points to the closest cluster centroid, with the goal of minimalising the sum of squared distances between data points and their assigned centroids.

For evaluating the correctness of the matches, a difference between transformed coordinates will be computed. For the transformation of the coordinates, a homography matrix will be generated. Different thresholds will be set in order to determine the accuracy rate of transformation.

[1] The mean intra-cluster distance for a sample represents the average distance between the sample and other samples in the same clusters.

[2] The mean nearest-cluster distance represents the average distance between the sample and the nearest cluster.

[3] see scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

5.2 Training and Validation of Models

The training process of neural networks is challenging and time-consuming. Large and robust datasets are needed to train the models to achieve high accuracy. The architecture and selection of hyperparameters for the training process are important as well.

5.2.1 Neural network for keypoint detection

The model was trained with a dataset that contained approximately 150,000 images with 64×64 pixels of blobs, corners, and edges. The dataset was divided into a train set and a validation set with an 80:20 ratio.

The loss function for this task was the *Mean Square Error function*, which computes the square difference between the prediction and target. As an optimizer algorithm, *Adam* was chosen and the learning rate was set to 0.0001. The batch size was set to 64. The number of epochs was set to 30.

The training process lasted for 25 epochs till the validation loss became stable, and the model converged to an optimal solution. The *Early Stop* [22] regularisation element was triggered. The loss function for each training set and validation set kept decreasing over iterations (figure 5.1).

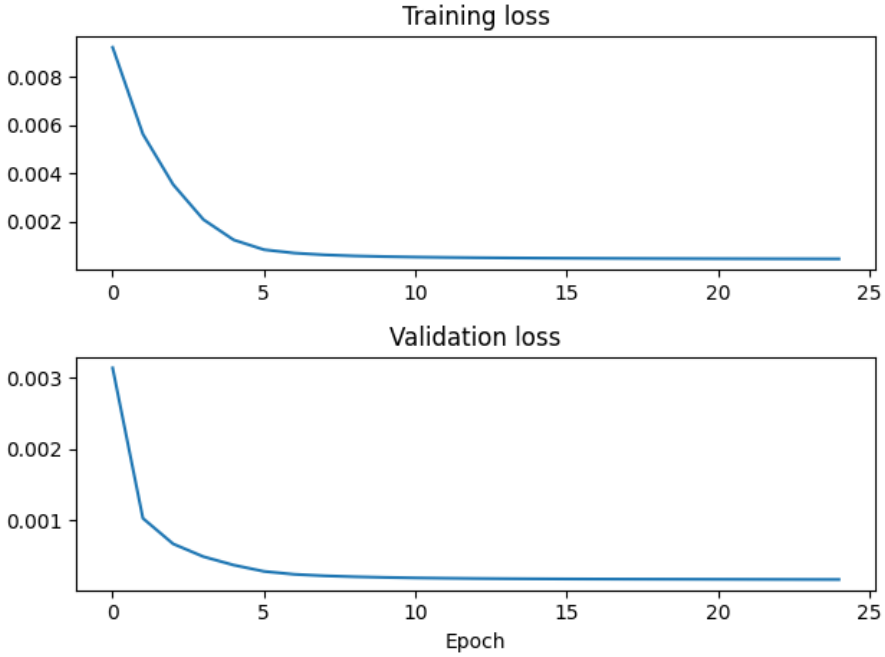


Figure 5.1: The loss function over 25 epochs of the training process with 150,000 images.

The training process ended up earlier because the regularisation element *Early Stop* was triggered. Early Stop is a regularisation element, which stops the training if the validation loss stops decreasing or starts to increase. For this training, *patience* was set to one and the *minimal difference* to zero.

The testing images are a primitive shape and microscopic image from the TESCAN dataset. The neural network performance was compared side to side with a *score* map produced by SuperPoint neural network. This score map was extracted before a *Non-Maximum Suppression*[4] algorithm was applied in SuperPoint implementation.

The features were extracted from the encoder of the neural network and upsampled to map them into an image. To each element in this map, a threshold of 4.5 was applied. The microscopic image was resized to an 600×400 image. Figure 5.2 represents side by side comparison between the two models. The blue marking on the images represents detected features. Even though the evaluation metrics (table 5.1) are not the best for the microscopic image, the model was able to detect the most visible features. Validating on primitive shapes, the model performed well and was able to detect each side of the square. SuperPoint performed well on microscopic image and detected corners of the square[5].

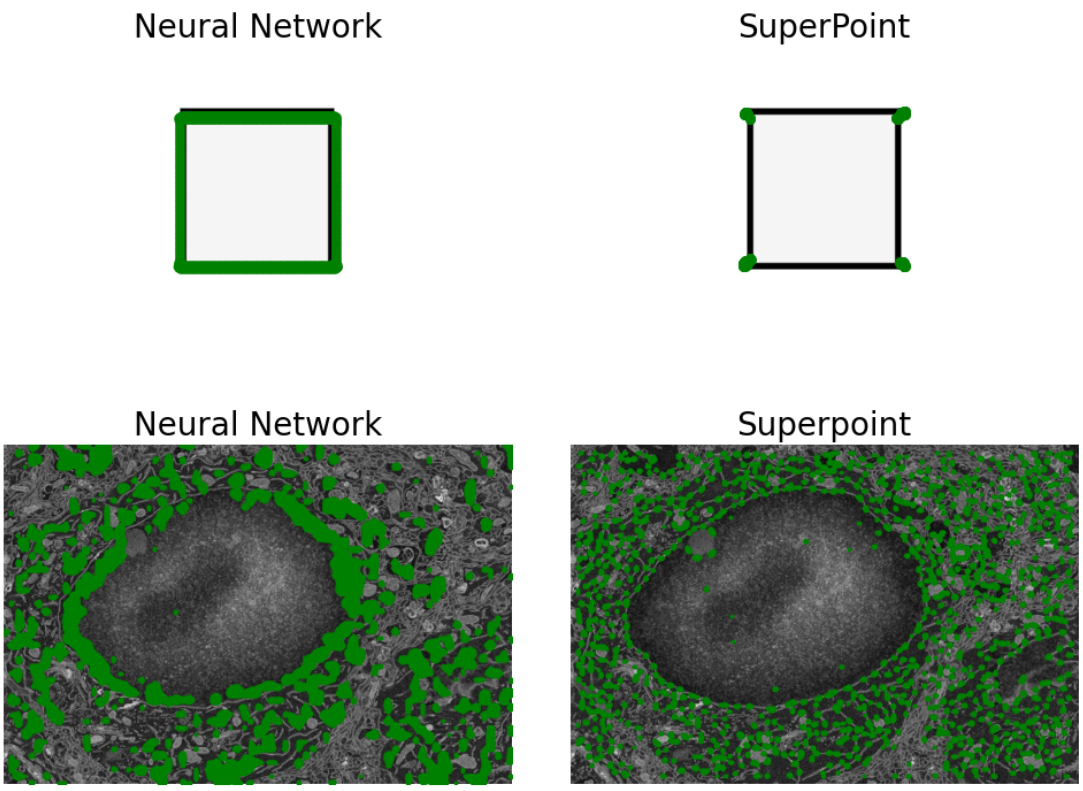


Figure 5.2: Feature detection in two images. Comparison between experimental neural network and pre-trained SuperPoint model.

[4]This algorithm removes nearby points by choosing a pixel with the highest local score.

[5]SuperPoint was trained with labels provided from a neural network that was learnt on detecting corners of primitive shapes, so this was expected.

The table 5.1 represents the results of evaluation metrics for the neural network. Chosen evaluation metrics for unsupervised learning showed that the neural network does not perform perfectly on microscopic images as SuperPoint does, it is able to detect blobs rather than outlines. Despite that, the model was able to detect the whole primitive object.

	Calinski Harabasz		**Silhouette**	
	5 clusters	**10 clusters**	**5 clusters**	**10 clusters**
Microscopic image	44.4	30.3	0.13	0.16
Primitive shape	3762.7	4794.41	0.72	0.67

Table 5.1: The evaluation metrics for the model on two different images.

5.2.2 Neural network for homography estimation

The model was trained only on regression prediction on a 3×3 matrix. The hyperparameters for the training process are *MSE* as loss function, learning rate 0.001 for optimizer *Adam* and epochs set to 10. The batch size was set to 64.

The training process resulted in big unsuccess due to the high loss function for the homography estimation. The model could not properly predict the homography matrix and images were distorted.

For additional experimenting with the model, another approach was tried. The architecture of the model changed slightly to match the new input data. The new input data represented matched descriptors which were generated by SuperPoint and SuperGlue. The image dataset remained the same as for the first training, as well as the hyperparameters.

The training once again could not provide sufficient results. The error rate for both approaches was high, and the predicted matrix could not be applied to warp the images.

5.2.3 Fine-tuning SuperPoint

The selected models perform good keypoint detection as well as matching between those keypoints on various images, including microscopic images. The SuperPoint pre-trained model `superpoint_v1` is the only provided model in the implementation.

To experiment if the SuperPoint detection accuracy can be enhanced, the model was fine-tuned. The proposed neural network generated the feature map for input images as their labels. The difference between the proposed model and SuperPoint is that the SuperPoint can properly detect corners and the proposed model has learnt how to detect whole shapes.

Approximately, 20000 images from the same dataset which was used in order to train the proposed model were used to evaluate if the SuperPoint neural network can be enhanced. The model was fine-tuned on the dataset, which was used to train the neural network for feature detection. The descriptors were not altered, which is why the model's weights which are responsible for descriptor detection were frozen.

In this training process, the chosen loss function is *MSE*, the *Adam* optimizer's learning rate is set to 0.001 and batch size to 64. The training process lasted for 5 epochs.

The training did not enhance SuperPoint's detection of keypoints. The loss function value during epochs was minimal. The number of keypoints detected and their probability rate of being a keypoint on an image with the original model and pre-trained model remained the same.

5.3 Image stitching pipeline with deep learning

This section provides an overview of the final deep learning image stitching pipeline using SuperPoint and SuperGlue as the two main actors who perform feature detection, description and matching. The pipeline will be suitable for stitching not only pair of images but grid-like structures. The dataset provided by TESCAN contains 2×2 images and 8×3 images which will be used to evaluate the stitching pipeline. The prepared dataset will also be used to evaluate the stitching process on various grid structures. The stitching of more than two images can be done with various methods. For this thesis, the stitching will be applied in rows starting from left (figure 5.3).

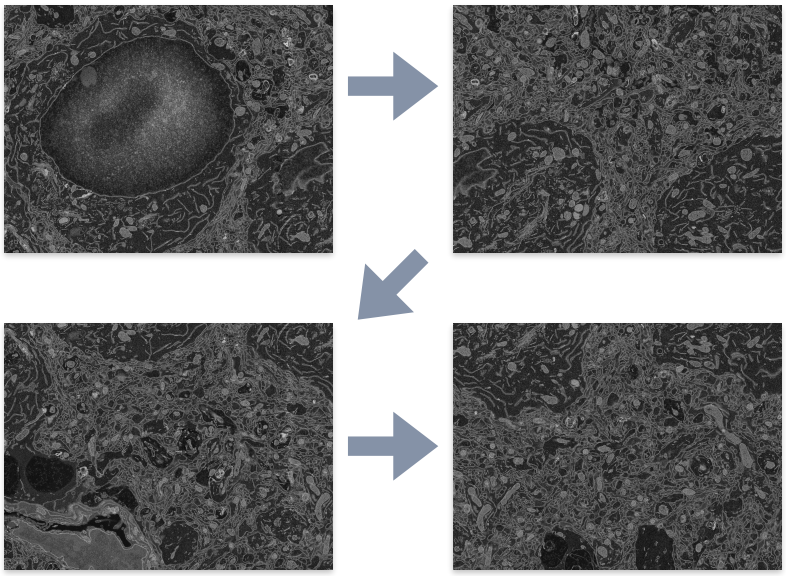


Figure 5.3: Proposed stitching method on grid images.

The idea behind the stitching algorithm is very simple. For each image, keypoints and descriptors are computed. The SuperPoint extracts keypoints and descriptors and SuperGlue processes them and returns matches. The matching pipeline returns all necessary information about images. The images are processed in pairs and the first image in the pair is during the process of matching updated.

In the first iteration, the first two images are processed and stitched, and the new image is created to replace the first position in the following pair. The algorithm stitches together all the provided images and ends when only one image is left. For the tomography, an OpenCV function is applied, and the image is wrapped. The coordinates are stored and used later for plotting on clean background. After evaluating and processing every image that was part of a grid, the algorithm ends up returning a stitched image. The algorithm is generic in stitching images with various grid sizes.

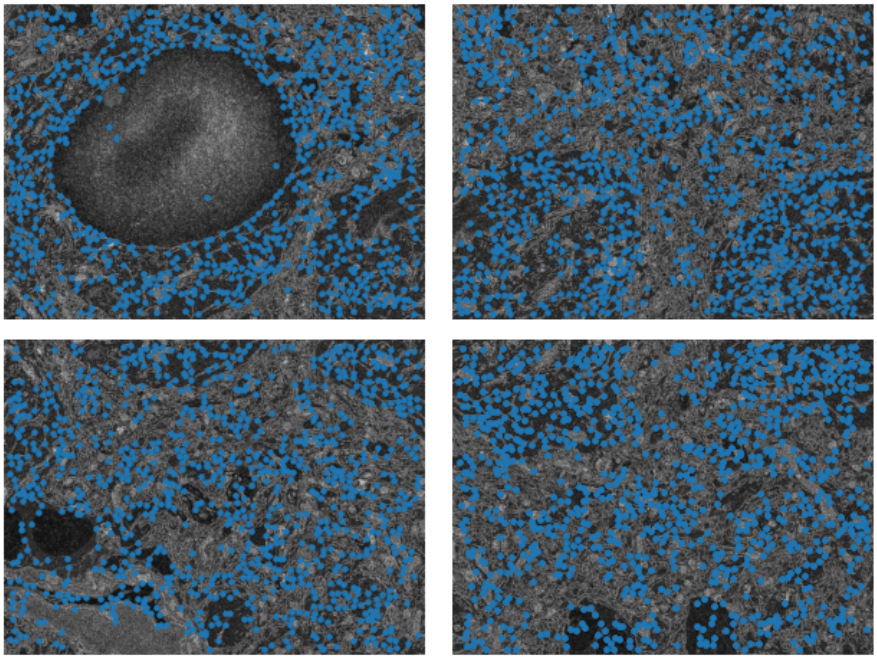


Figure 5.4: For each image, a set of keypoints and descriptors is extracted and stored for the following evaluations.

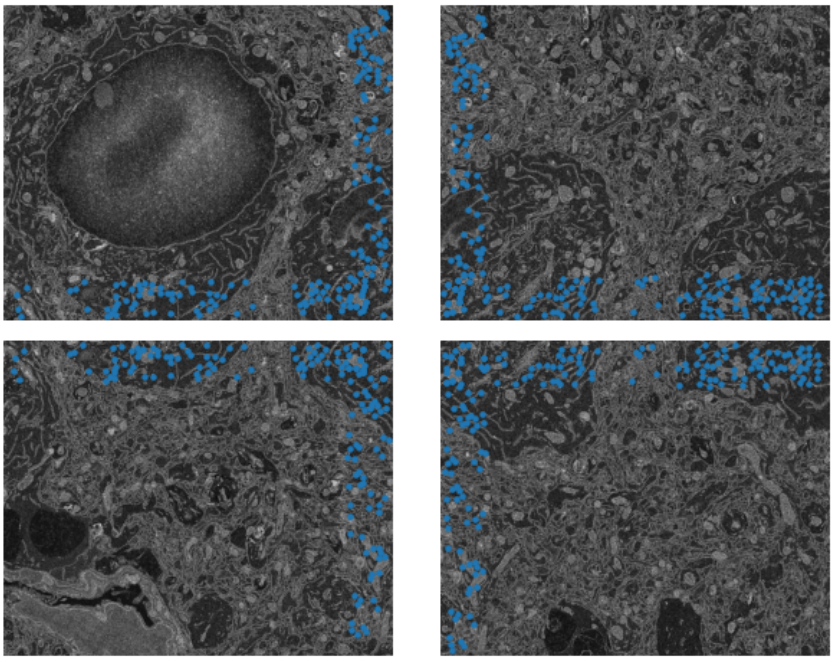


Figure 5.5: Example of matched keypoints between the images.

As previously mentioned, the algorithm begins with the first two images and computes matching keypoints. For the first two images, a homography matrix is computed. The

homography matrix is later used on warping the image and providing new coordinates for the corresponding image.

A new image is stored in an array of images and used for another step in the algorithm. The pairs of images are replaced by one warped image. After warping all the images, matching algorithm ends and images can be projected into a clean background with their new coordinates.

This approach is rotation sensitive, so the stitched image can be slightly off when corresponding images are highly rotated. Despite that, the results are acceptable. Images are stitched into the grid as it is shown in figure 5.6.

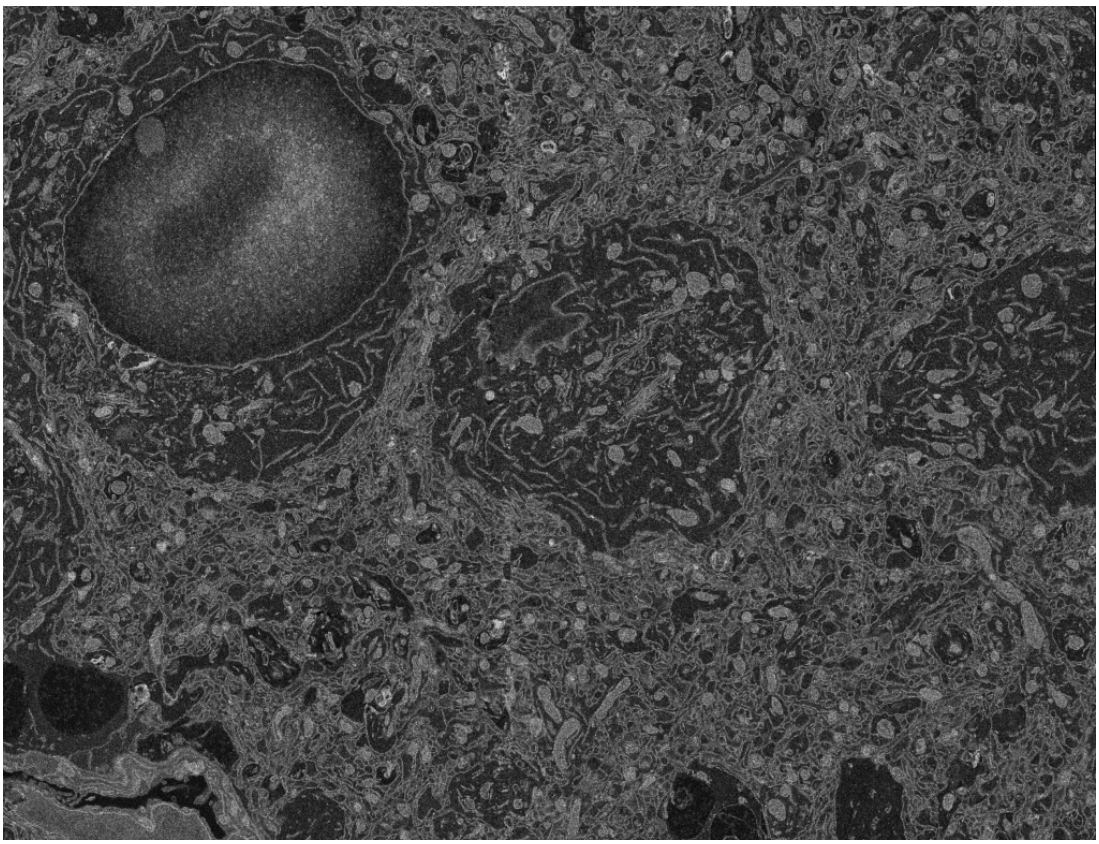


Figure 5.6: Final product of image stitching pipeline using deep learning components.

5.4 Comparison between approaches

In this section, comparison between the traditional pipeline (TP) and the deep learning pipeline (DLP) will be evaluated (table 5.2). The functions for traditional image stitching are from the library `OpenCV`.

	detection	**description**	**matching**
DLP	SuperPoint	SuperPoint	SuperGlue
TP	SIFT	SIFT	BFmatcher

Table 5.2: Methods used for comparison of the traditional approach and deep learning approach.

Each step of image stitching will be compared and evaluated. The comparison will be done on microscopic images in the dataset provided by TESCAN.

5.4.1 Feature detection and description

Both SuperPoint and SIFT performed well on detecting keypoints in the image. SIFT detected much more keypoints than Superpoint but by examining the produced keypoints, it is clear that many of the keypoints are not correctly assigned because many of them highlighted areas that do not provide useful information.

On the other hand, SuperPoint detected fewer keypoints, but each of them clearly highlighted interesting areas. The detected features are marked with red dots on both of the images.

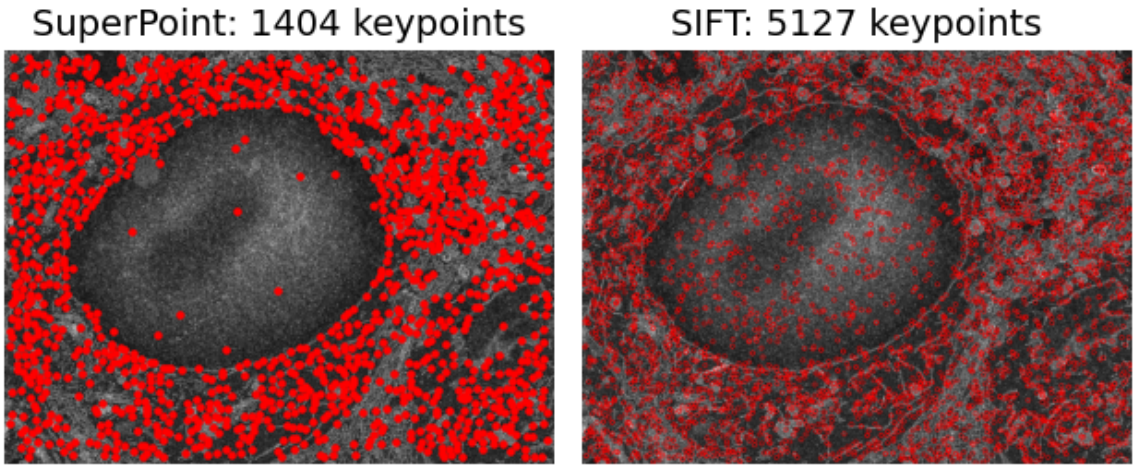


Figure 5.7: Feature detection in two images. Comparison between experimental neural network and pre-trained SuperPoint model.

By visually examining the images, it is possible to say, that the SuperPoint keypoints are more refined, and they try to outline features.

The descriptors for each image were provided by SuperPoint and SIFT as well. The correctness of the descriptors will be evaluated in the following section.

5.4.2 Feature matching

The generated descriptors will be used in order to match the pair of images. The pair of images overlap in very few keypoints which can be challenging. The function `BFmatcher` from the OpenCV library as the representative of traditional approaches for feature matching did not perform well. Using the *match* method, nearly 3000 matches were found. This method selects the best match for each feature descriptor.

The provided descriptors were not accurate enough for the matcher to match properly detected keypoints. At first sight, it is possible to state that the matches did not represent corresponding keypoints in each image.

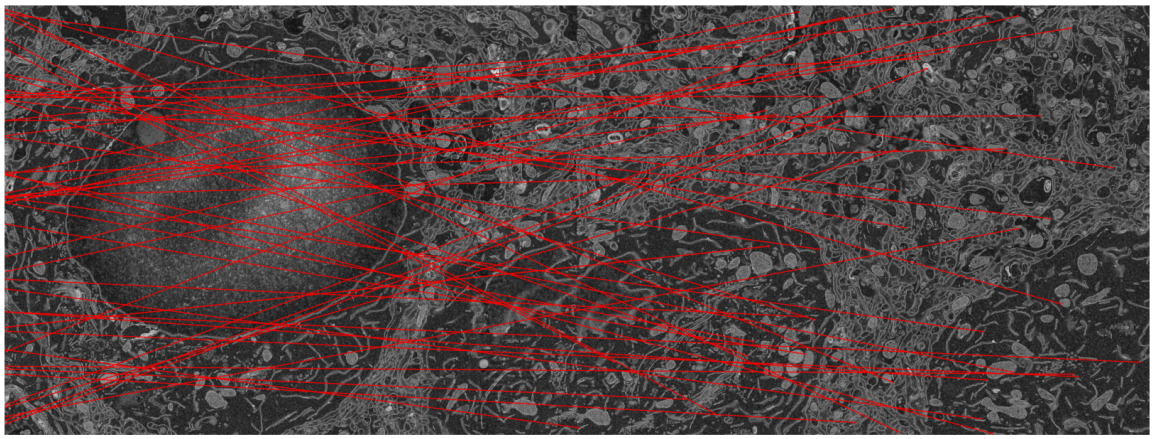


Figure 5.8: Feature matching on a microscopic image.

The BFmatcher's method *knnMatch* provides k best matches for each feature. The parameter was set to two, which resulted in two best matches for each feature descriptor. In order to differentiate between wrong and good matches, *David Lowe's ratio*[6] with threshold 0.7 was applied. If the Euclidean distance between evaluated matches was above a threshold, the match is eliminated [17].

The evaluation metric for validating the traditional matcher is *Precision*.

The precision metric calculates a ratio between true positive values and all positive values. The true positive outcomes are the ones, which passed the distance test. The number of matches which did not pass the test represents a false positive outcome.

$$precision = \frac{true\ positive\ matches}{true\ positive\ matches + false\ positive\ matches} \tag{5.1}$$

Between 4000 matches that the matcher found only 7 passed the ratio test, which makes the precision score 0%. After visually re-checking the remaining matches, none of them were correct. It can be determined that traditional approaches are not suitable for some images in the TESCAN dataset.

[6]The distance between each assigned feature descriptor and its best matches should be below a threshold to pass the ratio test.

The descriptors provided by SuperPoint are used in order to compute matches between features. The SuperGlue performs visibly better than the traditional approach did (figure 5.9). The SuperGlue produced 74 matches which visually match much better than the SIFT matches. The colours of stitches represent the confidence in a match which is also a product of SuperGlue. The red lines represent the most confident matches.

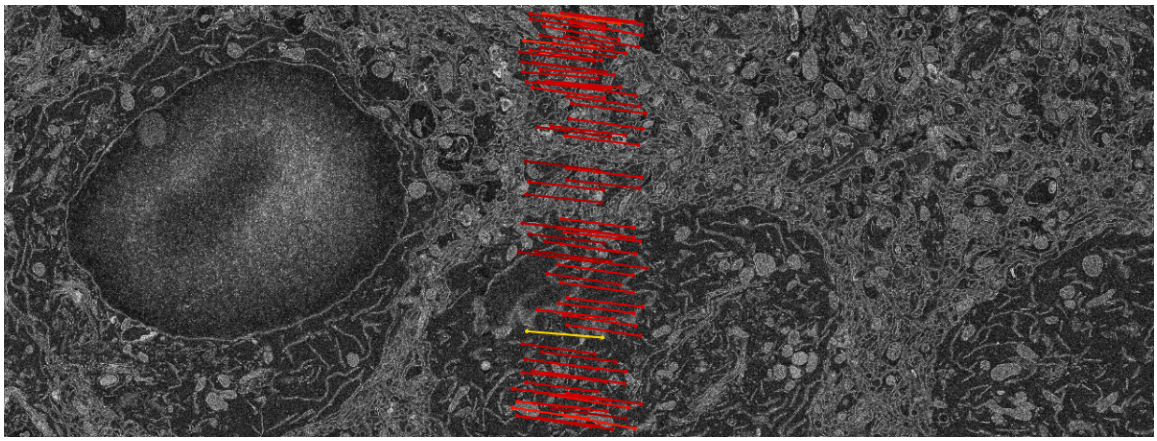


Figure 5.9: Feature matching on a microscopic image.

Although the matches seem reasonable, a proper evaluation is needed. To determine whether the keypoints are properly matched, a homography matrix will be applied to all points. The points are transformed into a homography coordinate system. The homography matrix is generated by the function `findHomography` from the OpenCV library. The transformation is properly described in the equation 2.5.

The coordinates from one image were compared with the coordinates in the second image. Various thresholds for maximal difference were applied. The accuracy grew as the maximum difference in pixels was higher. The accuracy is stated for both x and y coordinates. The threshold set to zero resulted in 0% accuracy which is more or less a float rounding problem.

Threshold	**0**	**1**	**2**	**3**	**4**	**5**
SuperGlue Accuracy	0%	23%	65%	86.5%	96%	98.6%

Table 5.3: The accuracy rate for transforming coordinates from one image to another image.

By executing the evaluation metric on SIFT, the results remained the same and that is 0% accuracy. The accuracy did not get better until the threshold was set for nearly the entire image.

It is safe to state, that SuperPoint and SuperPoint outperformed SIFT on this specific microscopic image. Even though the SIFT was able to detect more keypoints, the SuperPoints' keypoints looked more stable and were clearly outlining the features. SIFT's keypoints did not construct some kind of structure.

The matching performance based on provided descriptors was tested with the same method for generating a homography matrix. The SuperGlue outperformed SIFT by providing accurate matches within a few pixels radius.

5.4.3 Limits of deep learning approach

Even though SuperPoint's and SuperGlue's performance is without a doubt better than the performance of traditional methods, they have their limitations too.

The SuperPoint neural network performance on grainy images is acceptable as it is able to detect not very visible features (figure 5.11). The SuperPoint was able to detect over 2000 keypoints on the image which was resized to 1500×1200.

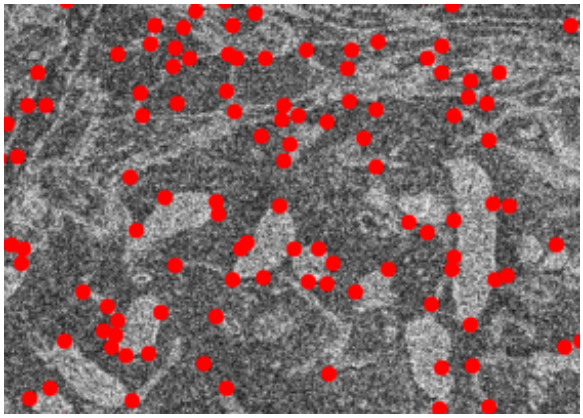


Figure 5.10: SuperPoint's feature detection on a structured image with keypoints represented as red dot markers. Light features are visibly outlined by SuperPoint.

After a solid performance of SuperPoint, the keypoints and descriptors are matched with SuperGlue. SuperGlue was not able to detect keypoints properly. A total of 24 matches have been found. In the overlap area, which is about 10% of images, keypoints are matched incorrectly.

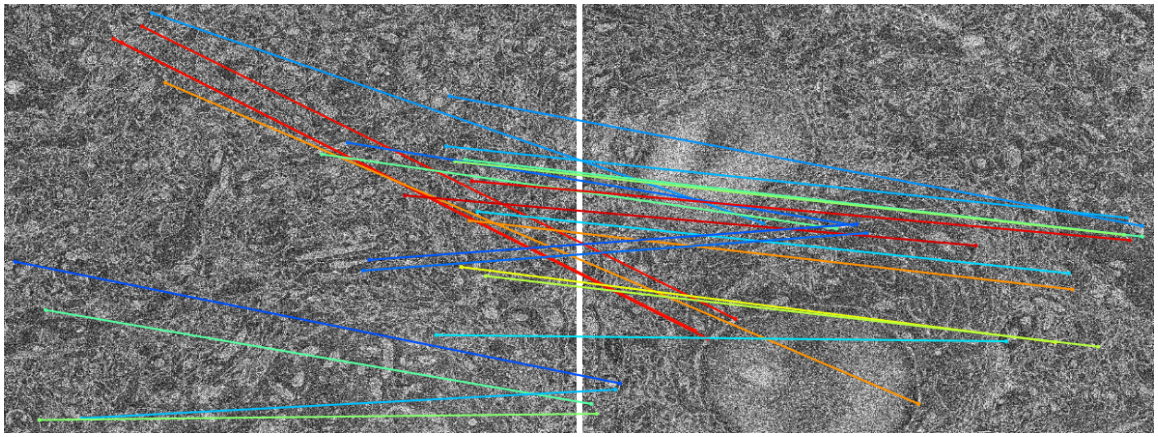


Figure 5.11: SuperGlue's matches on the grainy structured image. The colours of the line represent the confidence of a match. Going from red to blue, red represents a confident match and blue represents not so confident match.

The evaluation metric is the same as it was in the previous section. The homography was computed with provided matches and the correctness of the transformation was evaluated. The accuracy differs very highly from the previous result. The SuperGlue was not able to provide satisfactory accuracy even after 100 pixel dispersion (table 5.4).

Threshold	**0**	**1**	**2**	**10**	**100**
SuperGlue Accuracy	0%	16.6%	16.6%	25%	41.6%

Table 5.4: The accuracy rate for transforming coordinates from one image to another image.

The models were tested on dataset A to determine how well the models perform on augmented data. The images were split with 30% overlap space. The labels generated from non-augmented data are used as ground truth. Because the rotation crops the images in order to evade misleading keypoints, the keypoints which are no longer in the image will not be taken into consideration in the evaluation. The baseline accuracy of predicting correct matches on own dataset is presented in the following table:

Threshold	**1**	**2**	**3**
Baseline Accuracy	76.3%	95.99%	99.2%
Noise +15% Accuracy	34.8%	71.15%	88%
Brightness 1.16x Accuracy	62.4%	89.7%	97%
Rotation Accuracy	64.4%	90.7%	96.96%
All techniques Accuracy	37.3%	64.2%	79.24%

Table 5.5: The accuracy rate for transforming coordinates from one image to another image.

The baseline images underwent a transformation and were augmented. With three-pixel dispersion, the models can collectively produce highly accurate predictions on matches. The most tricky augmentation technique for models is a blur which leads to over 40% accuracy decrease from baseline. The division between images was same within the all augmented and non-augmented images.

The sum of missed keypoints was evaluated for each augmentation technique. As the ground truth keypoints, the non-augmented keypoints were chosen. The number of detected keypoints in all of the images is $10,424$. The points were compared by coordinates if they match with the baseline dataset.

The blur dataset resulted in missing 180 keypoints which is about a 1.7% miss rate. The brightness dataset resulted in detecting 11 more keypoints from the baseline set.

In this chapter, all of the experiments were explained and evaluated. The training of neural network for feature detection ended up in success as it was able to detect blobs, corners and edges of distinguishable features in microscopic images. The neural network was compared to the performance of SuperPoint and visually examined.

The second proposed experiment which included training the neural network on homography estimation resulted in unsuccess. Two approaches were tried and both of them did not succeed. The first intention was to train the neural network to estimate homography between two input images. The error rate was too high, and the result could not be used for further work. The second method was to train the neural network to determine the homography matrix based on provided matches by SuperGlue. The matched descriptors were put together into the neural network, and the training ended up the same way as for two images input.

Later on, an image stitching pipeline for grid stitching was proposed and explained. The pipeline uses **SuperPoint** and **SuperGlue** as deep learning techniques for feature detection, description and matching. For estimating homography and warping the images, traditional approaches were applied. For the traditional approach, part of a pipeline functions from OpenCV library **findHomography** and **warpPerspective** were used. Due to the unsuccess of an experiment of a neural network for homography estimation, the pipeline could not be enriched with another deep learning element.

Finally, the comparison between the traditional pipeline and the deep learning pipeline was evaluated. The representants of traditional approaches for this comparison were **SIFT**, **BFmatcher**. The deep learning approach outperformed the traditional approach in feature detection, description and matching on microscopic images.

The limitations of deep learning approaches were discussed and evaluated. The SuperPoint was able to detect and describe distinguishable features from simple to highly structured images. The SuperGlue was able to properly match keypoints on images, but with more structure of an image, its performance decreased. The evaluation of correct matches was provided by transforming keypoints with a homography matrix using **RANSAC**.

Overall, it is possible to say, that the deep learning approaches outperform traditional ones.

In the future, the constructed stitching pipeline could be enhanced with a homography matrix which was proposed. Due to unsuccessful training, the model could not be adapted to the final pipeline. Another suggestion is to create an application which will be user-friendly and the matching would not be generated by terminal commands.

Chapter 6

Conclusion

This thesis introduced a process of stitching the digital images as well as approaches to achieve the panoramic image. The baseline of image stitching was explained and discussed. This process contains steps such as feature detection, description and matching.

There are various handcrafted approaches which can solve this problem accurately. The most famous representatives of traditional methods in stitching pipelines are SIFT, SURF, or ORB. Using these methods, features can be matched. But despite that, they have their limitations too.

The neural networks are highly capable approaches which can with good and robust data learn nearly anything. The main representatives in this thesis which were used to experiment with are SuperPoint and SuperGlue. These two deep neural networks can produce high-quality predictions on feature locations, description and matching.

This thesis explains experiments which were done to achieve knowledge of neural networks and image processing. The first essential step in those experiments was to create datasets which are suited for specific tasks. The augmentation techniques in the process of creating a dataset were explained.

The first experiment, in order to understand feature detection, was to train a neural network to be able to detect features. The model was trained on a large dataset which was created with help of SuperPoint. Validating the neural network resulted in 0.16 Silhouette score for microscopic images and 0.67 for images with primitive object for 10 clusters.

This neural network was further used to fine-tune the Superpoint model, `superpoint_v1`. The SuperPoint's layers which involved prediction for descriptors were frozen and only layers which produce score maps were trained. The training was done with 20,000 images. The loss function during the training was very minimal, so it did not alter or enhanced the predictions of the baseline.

The second experiment was to train a neural network on homography estimation. Two approaches were tried, with the first one being the approach proposed in the original paper by authors of SuperPoint and SuperGlue. The approach proposes to predict the homography matrix from two input images. The labels were generated by obtaining matches from SuperGlue neural network and computing the homography matrix with the help of RANSAC. The model could not learn how to properly predict the matrix. The second approach was to directly use the matches of SuperGlue to estimate homography. The neural network after training could not predict the homography as well but resulted in a lower mean square error rate than the first approach. This could be further examined.

The traditional approaches were compared with deep learning approaches on microscopic data provided by TESCAN. Both SuperPoint and SIFT performed well on detecting

the features on images, but SIFT was inclined to highlight features that were not as that prominent. On the other hand, SuperPoint outlined the most prominent features and produced a more structured set of keypoints. The feature matching with deep learning approach SuperGlue outperformed handcrafted approaches. Estimating homography and computing the accuracy of the matches by projecting the keypoints from one image to another was with the 3-pixel dispersion resulted of nearly 90%.

The TESCAN dataset was used to evaluate the transformation of the matching pipeline of SuperGlue implementation. This grid matching pipeline uses SuperPoint's model and SuperGlue's outdoor model and OpenCV library functions to produce stitched images.

The SuperPoint and SuperGlue performance was validated on an augmented dataset with ground truth labels from non-augmented images in the same dataset. The models were tested by projecting the keypoints. With the three-pixel dispersion, the accuracy did not get lower than 79%. The best accuracy of 97% was obtained on images with enhanced brightness.

To sum it all up, it is possible to state, that neural networks provide phenomenal advantages in image processing.

Bibliography

[1] Bradski, G. and Kaehler, A. *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*. 2ndth ed. O'Reilly Media, Inc., 2013. ISBN 1449314651.

[2] Bradski, G. and Kaehler, A. *Feature Detection and Description*. 2019. Available at: https://docs.opencv.org/4.x/db/d27/tutorial_py_table_of_contents_feature2d.html.

[3] Brown, M. and Lowe, D. G. Automatic Panoramic Image Stitching using Invariant Features. *Int. J. Comput. Vision*. Hingham, MA, USA: Kluwer Academic Publishers. february 2007, vol. 74, no. 1, p. 59–73. DOI: 10.1007/s11263-006-0002-3. ISSN 0920-5691. Available at: https://doi.org/10.1007/s11263-006-0002-3.

[4] Derpanis, K. G. *Overview of the RANSAC Algorithm* [http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf]. 1.2. 2013. [Online; accessed 29-April-2023].

[5] DeTone, D., Malisiewicz, T. and Rabinovich, A. Deep Image Homography Estimation. *CoRR*. 2016, abs/1606.03798. Available at: http://arxiv.org/abs/1606.03798.

[6] DeTone, D., Malisiewicz, T. and Rabinovich, A. SuperPoint: Self-Supervised Interest Point Detection and Description. *CoRR*. 2017, abs/1712.07629. Available at: http://arxiv.org/abs/1712.07629.

[7] Dusmanu, M., Rocco, I. and al. et. D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. In: *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

[8] Dusmez, S., Heydarzadeh, M. and Nourani, M. e. a. Remaining Useful Lifetime Estimation for Power MOSFETs Under Thermal Stress With RANSAC Outlier Removal. *IEEE Transactions on Industrial Informatics*. 2017, vol. 13, no. 3, p. 1271–1279. DOI: 10.1109/TII.2017.2665668.

[9] El Sawy, A.-B. M., Hazem, E. L. B. and Loey, M. CNN for handwritten arabic digits recognition based on LeNet5. In: Springer. *International Conference on Advanced Intelligent Systems and Informatics*. 2016, p. 566–575.

[10] Fergus, R., Perona, P. and Zisserman, A. Weakly Supervised Scale-Invariant Learning of Models for Visual Recognition. *International Journal of Computer Vision*. Springer. 2007, vol. 71, no. 3, p. 273–303. DOI: 10.1007/s11263-006-8707-x. ISSN 1573-1405. Available at: https://doi.org/10.1007/s11263-006-8707-x.

[11] Fischler, M. A. and Bolles, R. C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*. New York, NY, USA: Association for Computing Machinery. jun 1981, vol. 24, no. 6, p. 381–395. DOI: 10.1145/358669.358692. ISSN 0001-0782. Available at: https://doi.org/10.1145/358669.358692.

[12] Harris, C. G. and Stephens, M. J. A Combined Corner and Edge Detector. In: *Alvey Vision Conference*. 1988.

[13] Jangra, N. A Detailed Guide to the Powerful SIFT Technique for Image Matching (with Python code). *Analytics Vidhya*. October 2019. Available at: https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/.

[14] Krizhevsky, A., Sutskever, I. and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In: Curran Associates, Inc. *NIPS*. 2012, p. 1097–1105.

[15] Liu, W., Wang, S., Deng, Z. et al. A Review of Image Feature Descriptors in Visual Positioning. *IPIN-WiP*. 2021.

[16] Lowe, D. G. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*. Kluwer Academic Publishers. november 2004, vol. 60, no. 2, p. 91–110. DOI: 10.1023/B:VISI.0000029664.99615.94. ISSN 0920-5691. Available at: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[17] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*. Springer. 2004, vol. 60, no. 2, p. 91–110.

[18] Nie, L., Lin, C. and al., K. L. et. Deep Rectangling for Image Stitching: A Learning Baseline. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun 2022, p. 5730–5738. DOI: 10.1109/cvpr52688.2022.00565. Available at: https://ieeexplore.ieee.org/document/9878669.

[19] OpenCV. *Corner Detection with Harris Corner Detection Method* [https://docs.opencv.org/4.x/dc/d0d/tutorial_py_features_harris.html]. 2021. Accessed on April 5, 2023.

[20] Palacio-Niño, J. and Berzal, F. Evaluation Metrics for Unsupervised Learning Algorithms. *CoRR*. 2019, abs/1905.05667. Available at: http://arxiv.org/abs/1905.05667.

[21] Pellikka, M. and Lahtinen, V. A robust method for image stitching. *Pattern Analysis and Applications*. Springer. 2021, vol. 24, p. 1847–1858. DOI: 10.1007/s10044-021-01005-8. Available at: https://doi.org/10.1007/s10044-021-01005-8.

[22] Prechelt, L. Early Stopping — But When? In: Montavon, G., Orr, G. B. and Müller, K.-R., ed. *Neural Networks: Tricks of the Trade: Second Edition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 53–67. DOI: 10.1007/978-3-642-35289-8_5. ISBN 978-3-642-35289-8. Available at: https://doi.org/10.1007/978-3-642-35289-8_5.

[23] Sarlin, P.-E., DeTone, D. and al., T. M. et. SuperGlue: Learning Feature Matching with Graph Neural Networks. In: *CVPR*. 2020. Available at: https://arxiv.org/abs/1911.11763.

[24] Schaffalitzky, F. and Zisserman, A. Multi-view Matching for Unordered Image Sets, or "How Do I Organize My Holiday Snaps". In: *Computer Vision — ECCV 2002*. Berlin, Heidelberg: Springer Berlin Heidelberg, April 2002, p. 414–431. DOI: 10.1007/3-540-47969-4_28. ISBN 978-3-540-47969-7. Available at: https://link.springer.com/chapter/10.1007/3-540-47969-4_28.

[25] Shorten, C. and Khoshgoftaar, T. M. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*. Springer. 2019, vol. 6, no. 1, p. 60. DOI: 10.1186/s40537-019-0197-0.

[26] Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *ArXiv preprint arXiv:1409.1556*. 2014.

[27] Szeliski, R. Image Alignment and Stitching: A Tutorial. *Foundations and Trends® in Computer Graphics and Vision*. 2007, vol. 2, no. 1, p. 1–104. DOI: 10.1561/0600000009. ISSN 1572-2740. Available at: http://dx.doi.org/10.1561/0600000009.

[28] Szeliski, R. *Computer Vision: Algorithms and Applications*. 2nd ed. Springer Cham, january 2022. ISBN 978-3-030-34371-2. Available at: https://doi.org/10.1007/978-3-030-34372-9.

[29] Tyagi, D. Introduction to Feature Detection and Matching. *Medium*. January 2019. Available at: https://medium.com/data-breach/introduction-to-feature-detection-and-matching-65e27179885d.

[30] Williams, E., Moore, J. and Li, S. W. e. a. The Image Data Resource: A Bioimage Data Integration and Publication Platform. *Nat Methods*. Nature Publishing Group. 2017, vol. 14, no. 8, p. 775–781. DOI: 10.1038/nmeth.4326.

Appendix A

Contents of the included storage media

- `thesis.pdf` Thesis report file.
- `poster.pdf` Poster for the thesis.
- `scripts/` Folder with source files.
- `pretrained_models/` Folder with pre-trained model.
- `dataset/` Folder with used datasets in the thesis.
- `requirements.txt` List of Python libraries dependencies
- `README.md` README manual for the project.
- `latex/` Folder with LaTeXsource files.