Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 867

Lin Chen

# Deep Learning for Feature based Image Matching

München 2021

Verlag der Bayerischen Akademie der Wissenschaften

ISSN 0065-5325 | ISBN 978-3-7696-5279-6

Diese Arbeit ist gleichzeitig veröffentlicht in:

Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover

ISSN 0174-1454, Nr. 369, Hannover 2021

Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 867

# Deep Learning for Feature based Image Matching

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

Vorgelegt von

M. Sc. Lin Chen

Geboren am 06.12.1987 in Hubei, China

München 2021

Verlag der Bayerischen Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5279-6

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 369, Hannover 2021

**Adresse der DGK:**




Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften (DGK)

Alfons-Goppel-Straße 11 ● D – 80 539 München

Telefon +49 – 331 – 288 1685 ● Telefax +49 – 331 – 288 1759

E-Mail post@dgk.badw.de ● http://www.dgk.badw.de

Prüfungskommission:

Vorsitzender: Prof. Dr.-Ing. habil. Jürgen Müller

Referent: Prof. Dr.-Ing. habil. Christian Heipke

Korreferenten: Prof. Pascal Fua (EPFL)
Prof. Dr.-Ing. habil. Monika Sester
Prof. Dr. techn. Franz Rottensteiner

Tag der mündlichen Prüfung: 19.03.2021

© 2021 Bayerische Akademie der Wissenschaften, München

Alle Rechte vorbehalten. Ohne Genehmigung der Herausgeber ist es auch nicht gestattet,
die Veröffentlichung oder Teile daraus auf photomechanischem Wege (Photokopie, Mikrokopie) zu vervielfältigen

ISSN 0065-5325 ISBN 978-3-7696-5279-6

# Abstract

Feature based image matching aims at finding matched features between two or more images. It is one of the most fundamental research topics in photogrammetry and computer vision. The matching features are a prerequisite for applications such as image orientation, Simultaneous Localization and Mapping (SLAM) and robot vision. A typical feature based matching algorithm is composed of five steps: feature detection, affine shape estimation, orientation, description and descriptor matching. Today, the employment of deep neural network has framed those different steps as machine learning problems and the matching performance has been improved significantly.

One of the main reasons why feature based image matching may still prove difficult is the complex change between different images, including geometric and radiometric transformations. If the change between images exceeds a certain level, it will also exceed the tolerance of those aforementioned separate steps and, in turn, cause feature based image matching to fail. This thesis focuses on improving feature based image matching against large viewpoint and viewing direction change between images. In order to improve the feature based image matching performance under these circumstances, affine shape estimation, orientation and description are solved with deep learning architectures. In particular, Convolutional Neural Networks (CNN) are used.

For the affine shape and orientation learning, the main contribution of this thesis is twofold. First, instead of a Siamese CNN, only one branch is needed and the loss is built based on the geometric measures calculated from the mean gradient or second moment matrix. Therefore, for each of the input patches, a global minimum, namely the canonical feature, exists. Second, both the affine shape and orientation are solved simultaneously within one network by combining the loss used for affine shape and orientation learning. To the best of the author's knowledge, this is the first time these two modules are reported to have been successfully trained simultaneously.

For the descriptor learning part, a new weak match is defined. For any input feature patch, a slightly transformed patch that lies far from the input feature patch in descriptor space is defined as a weak match feature. A weak match finder network is proposed to actively find these weak match features. In a following step, the found weak matches are used in the standard descriptor learning framework. In this way, the intra-variance of the appearance of matched feature patch pairs is explored in depth and, accordingly, the invariance of feature descriptors against viewpoint and viewing direction change is improved.

The proposed feature based image matching method is evaluated on standard benchmarks and is used to solve for the parameters of image orientation. For the image orientation task, aerial oblique images are taken

into account. Through analysis of the experiments conducted for small image blocks, it is shown that deep learning feature based image matching leads to more registered images, more reconstructed 3D points and a more stable block connection.

**Keywords** feature-based image matching, image orientation, descriptor learning, feature orientation, affine shape estimation, deep learning, CNN, oblique aerial images

# Kurzfassung

Die merkmalsbasierte Bildzuordnung zielt darauf ab, übereinstimmende Merkmale zwischen zwei oder mehr Bildern zu finden. Es ist eines der grundlegendsten Forschungsthemen in Photogrammetrie und Computer Vision. Übereinstimmende Merkmale sind eine Voraussetzung für Anwendungen wie Bildorientierung, Simultaneous Localization and Mapping (SLAM) und maschinelles Sehen. Ein typischer merkmalsbasierter Zuordnungsalgorithmus besteht aus fünf Schritten: Merkmalserkennung, Schätzung der affinen Form, Orientierung, Beschreibung und Deskriptorzuordnung. Heutzutage bilden tiefe neuronale Netze den Rahmen, um die verschiedenen Schritte als Probleme des maschinellen Lernens zu verstehen, dabei wurde die Qualität der Bildzuordnung erheblich verbessert.

Einer der Hauptgründe, warum sich die merkmalsbasierte Bildzuordnung immer noch als schwierig erweisen kann, ist der komplexe Unterschied zwischen verschiedenen Bildern, einschließlich geometrischer und radiometrischer Transformationen. Wenn der Unterschied zwischen Bildern bestimmte Größen annimmt, überschreitet er die Grenzen der in den genannten separaten Schritten verwendeten Lösungen und führt dazu, dass die merkmalsbasierte Bildzuordnung fehlschlägt. Diese Arbeit konzentriert sich auf die Verbesserung der merkmalsbasierten Bildzuordnung bei großen Basislängen und unterschiedlichen Blickrichtungen zwischen Bildern. Um die merkmalsbasierte Bildzuordnungsleistung unter diesen Umständen zu verbessern, werden affine Formschätzung, Orientierung und Beschreibung mit Deep Learning Architekturen genutzt. Insbesondere werden Convolutional Neural Networks (CNN) verwendet.

Für das Lernen von affiner Form und Orientierung liegt der Hauptbeitrag dieser Arbeit in zwei Bereichen. Einerseits wird anstelle eines siamesischen CNN nur ein Zweig benötigt, und die Verlustfunktion wird basierend auf den geometrischen Maßen aufgebaut, die aus dem mittleren Gradienten oder der Matrix der zweiten Momente berechnet werden. Daher existiert für jedes der Eingabefenster ein globales Minimum, nämlich das kanonische Merkmal. Andererseits werden sowohl die affine Form als auch die Orientierung gleichzeitig in einem Netzwerk bestimmt, indem die Verlustfunktionen beiden Teiloptimierungen kombiniert werden. Nach Kenntnis des Autors ist es das erste Mal, dass diese beiden Module gleichzeitig erfolgreich trainiert wurden.

Für das Anlernen des Deskriptors wird eine neue schwache Übereinstimmung definiert. Für jedes Eingabefenster wird ein leicht transformiertes Fenster, das im Deskriptorraum weit vom Eingabefenster entfernt liegt, als schwaches Übereinstimmungsmerkmal definiert. Zum Auffinden dieser schwachen Übereinstimmungen wird ein eigenes Netz entwickelt. In einem folgenden Schritt werden die gefundenen schwachen Übereinstimmungen im Standard-Deskriptor-Lernframework verwendet. Das Erscheinungsbild übereinstimmender

Fensterpaare wird eingehend untersucht und die Invarianz von Merkmalsdeskriptoren gegenüber Blickwinkel und -richtungsänderungen verbessert.

Das vorgelegte merkmalsbasierte Bildzuordnungsverfahren wird an Standardbenchmarks evaluiert und für die Bildorientierung genutzt. Für die Aufgabe der Bildorientierung werden Schrägluftbilder verwendet. Durch die Analyse der für kleine Bildblöcke durchgeführten Experimente wird gezeigt, dass eine auf Deep Learning Merkmalen basierende Bildzuordnung zu mehr registrierten Bildern, mehr rekonstruierten 3D-Punkten und einem stabileren Blockverband führt.

**Schlagworte** merkmalsbasierte Bildzuordnung, Bildorientierung, Deskriptorlernen, Merkmalsorientierung, Schätzung der affinen Form, Deep Learning, CNN, Schrägluftbilder

# Contents

**1. Introduction** **9**
1.1. Motivation . . . 9
1.2. Main Contributions . . . 11
1.3. Thesis Outline . . . 12

**2. Basics** **15**
2.1. Feature based Image Matching . . . 15
2.1.1. Overview: What is Feature based Image Matching? . . . 15
2.1.2. Desired Properties for Detected Features and Descriptors . . . 16
2.1.3. Scale-Invariant Feature Detection . . . 19
2.1.4. Feature Affine Shape Estimation . . . 20
2.1.5. Feature Orientation Assignment . . . 20
2.1.6. Feature Description . . . 21
2.1.7. Descriptor Matching . . . 21
2.2. Convolutional Neural Network (CNN) . . . 22
2.2.1. Architecture of CNN . . . 22
2.2.2. Training of CNN . . . 23
2.3. Siamese Convolutional Neural Network . . . 25

**3. Related Work** **27**
3.1. Local Feature Detection . . . 27
3.1.1. Translation and Rotation Invariant Features . . . 27
3.1.2. Scale Invariant Features . . . 28
3.1.3. Detectors based on a Comparison of Grey Values or Saliency . . . 29
3.1.4. Detectors based on Machine Learning . . . 29
3.2. Feature Orientation and Affine Shape Estimation . . . 30
3.2.1. Orientation Assignment . . . 30
3.2.2. Affine Shape Estimation . . . 31
3.3. Local Feature Description . . . 32
3.3.1. Hand Crafted Descriptors . . . 34
3.3.2. Machine Learning based Descriptors . . . 35
3.4. An Application: Orientation of Oblique Aerial Images . . . 38

3.5. Discussion . . . 40
3.5.1. Orientation Assignment and Affine Shape Estimation . . . 40
3.5.2. Descriptor Learning . . . 41
3.5.3. An Aerial Photogrammetric Benchmark . . . 42
3.5.4. Ability to Transfer Learned Modules . . . 42

**4. Deep Learning Feature Representation** **43**
4.1. Overview of the Methodology . . . 43
4.2. Descriptor Learning using Active Weak Match Finder - WeMNet . . . 45
4.2.1. Descriptor Learning Architecture . . . 46
4.2.2. Generation of Training Pairs . . . 46
4.2.3. Loss Function . . . 48
4.2.4. Weak Match Branch . . . 49
4.3. Self Supervised Feature Affine Shape Learning - MoNet . . . 54
4.3.1. Affine Transformation Decomposition . . . 54
4.3.2. Self Supervised Affine Shape Estimation Module . . . 57
4.4. Self Supervised Orientation Assignment Module - MGNet . . . 61
4.5. Full Affine Estimation Network - Full-AffNet . . . 64
4.5.1. Full Affine Network . . . 64
4.5.2. Training Loss . . . 65
4.5.3. Data Augmentation . . . 65
4.6. Inference based on the Trained Networks . . . 65
4.7. Discussion . . . 67
4.7.1. Descriptor Learning . . . 67
4.7.2. Affine Shape Estimation . . . 68
4.7.3. Orientation Assignment Learning . . . 68
4.7.4. The Inference Pipeline . . . 69

**5. Experiments and Results** **71**
5.1. Datasets . . . 74
5.1.1. Datasets for Training . . . 74
5.1.2. Datasets for Testing . . . 76
5.2. Evaluation and Analysis Criteria . . . 84
5.2.1. Task A: Patch based Image Matching . . . 84
5.2.2. Task B: Descriptor Distance Analysis . . . 84
5.2.3. Task C: Feature based Image Matching . . . 85
5.2.4. Task D: Image Orientation . . . 85
5.2.5. Summary of Tasks and Involved Datasets . . . 87
5.3. Descriptor Learning and Patch Based Image Matching . . . 89
5.3.1. Parameter Study for WeMNet . . . 89

5.3.2. Comparison to Related Work . . . 91
5.4. Descriptor Distance Analysis . . . 95
5.4.1. Translation . . . 95
5.4.2. Rotation . . . 96
5.4.3. Affine Shape Transformation . . . 97
5.5. Image Matching Analysis . . . 101
5.5.1. Parameter Study for Affine Shape Learning . . . 102
5.5.2. Image Matching for Rotation Dataset . . . 104
5.5.3. Image Matching for Hpatches Affine Dataset . . . 107
5.6. Image Orientation . . . 109
5.6.1. Determination of Image Orientation . . . 109
5.6.2. Experiment Setup Details . . . 111
5.6.3. Orientation Result of Different Blocks . . . 111
5.6.4. Matching Quality Analysis . . . 116

**6. Discussion** **127**
6.1. Descriptor Learning and Patch Based Image Matching . . . 127
6.1.1. Parameter Study . . . 127
6.1.2. Comparison to Related Works . . . 128
6.2. Descriptor Distance Analysis . . . 128
6.2.1. Translation . . . 129
6.2.2. Rotation . . . 129
6.2.3. Affine Shape Transformation . . . 129
6.3. Feature based Image Matching . . . 130
6.3.1. Parameter Study . . . 130
6.3.2. Rotation Set . . . 130
6.3.3. Affine Set . . . 131
6.4. Image Orientation . . . 131

**7. Conclusion and Outlook** **135**

**Bibliography** **136**

**A. Affine Shape Adaptation Theory** **147**
A.1. transformation of affine Gaussian scale-space . . . 147
A.2. Local affine distortion measurement . . . 148
A.3. More affine transformation . . . 149

# 1. Introduction

## 1.1. Motivation

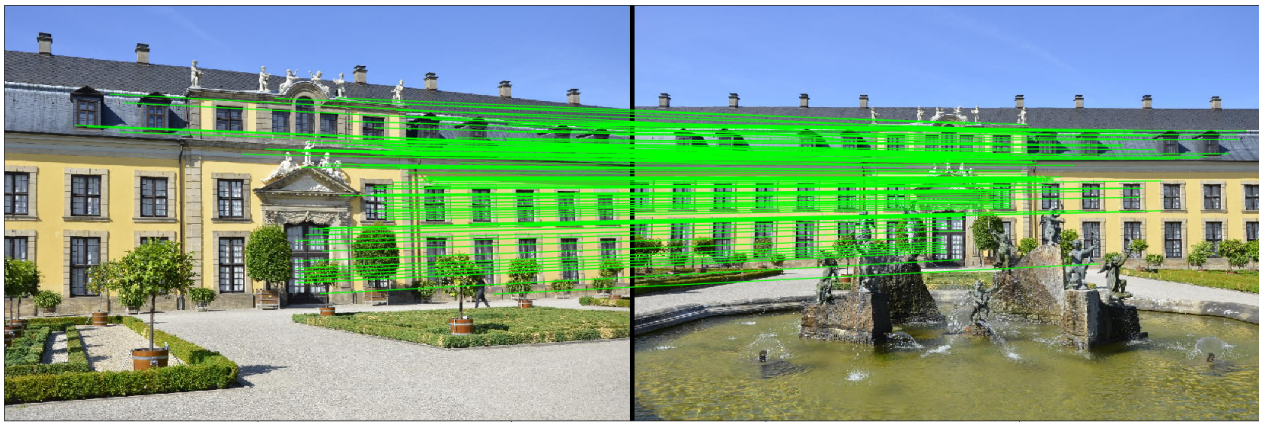
Feature based image matching, also called feature matching, is a method to solve the correspondence problem between two or more images. Feature matches are a requirement for the estimation of image orientation parameters (also called pose parameters), which, in turn, are a prerequisite of all geometric appplications in photogrammetry, robotics and computer vision which involve three dimensions. 3D reconstruction from multiple images, Simultanesous Localisation And Mappig (SLAM), Structure-from-Motion (SfM) and the generation of image mosaics all rely on image coordinates of matched conjugate features. Therefore, the quality of matching algorithms is vital for the stability and quality of the solution to those problems.

For applications such as 3D reconstruction from multiple images, a large enough number of well-distributed matching points is critical to the orientation of involved images. First, the number of correct matching points between two images must be as large as the minimum required number of matches to solve the relative orientation between images. To ensure that enough matches can be derived, hundreds or sometime thousands of features are detected for each image, mainly relying on the image size, the level of texture contained in the images and the required number of matching points for the specific application involved. Second, the matching points should be well distributed in the image and object space. In view of using the coordinates of matching features as observations to solve the parameters of image geometry, the localization accuracy of the detected features is vital to the precision of reconstructed 3D points. For applications such as a vision-based robot navigation system, delivering matching points in real time is required.

For some of the aforementioned applications, there are already several well-engineered feature matching techniques in use. For instance, SIFT (Scale Invariant Feature Transform) [Lowe, 2004] is used to match images containing a certain scale and a slight viewpoint and viewing direction change and only few cases of failure have been reported for matching these types of images. An example for the matching two images for this case is illustrated in figure 1.1. On the other hand, large relative changes between images, i.e., illumination change or large viewpoint and viewing direction change, still render feature matching a challenging task. In addition, strong repetitive patterns and texture-less regions contained in images also pose a challenge to feature matching. Although feature matching has been studied for several decades, the question of how to solve these challenging cases still remains.



(a) The input image pair



(b) Verified matches for the input image pair

Figure 1.1.: An example of matching a pair of images. The first row (a) shows the two input images containing slight scale as well viewpoint and viewing direction change, while the second row (b) shows the matches obtained after running SIFT (with affine shape estimation) and two-view geometry (fundamental matrix) verification. The matches are represented by the green lines linking the features detected in the first image to the matched features detected in the second image.

As is widely known, feature based image matching is composed of five steps: feature detection, affine shape estimation, orientation, description and descriptor matching. Among those steps, affine shape estimation is optional, it is used to decrease the appearance change caused by large relative geometric transformations, which in turn, is a result of considerable viewpoint and viewing direction change between images. Therefore, the affine shape estimation of features plays a key role when the viewpoint and viewing direction change between images are large. The goal of this thesis is to improve the performance of feature matching for images containing large viewpoint and viewing direction change, so affine shape estimation forms one of the research focuses.

In essence, detecting and describing features is an alternative way to represent an image. Compared to the number of pixels contained in an image, only a fraction of features and descriptors are extracted as a compressed representation of the input image. The most essential requirements for a feature based image

matching algorithm are the following two aspects: Firstly, the detected features must be distinctive and invariant to changes between different images may occur. Secondly, with application of a similarity measure (e.g., the Euclidean distance between vectors), the final representation of features (descriptors) should be capable of differentiating features despite complex changes. As the appearance of local features obtained from different images can be very different, each of the components in feature based image matching has a tolerance for change between images. If changes exceed the tolerance of one or more modules, the feature matching algorithm tends to fail. This is the starting point of this work.

Among the five steps of feature based image matching, this thesis will take a closer look at feature affine shape estimation, orientation assignment and description. By casting the design of those three modules as a machine learning problem using deep neural networks, those three modules are trained and then applied to real image matching tasks.

First, the feature affine shape estimation, a step after feature detection, is explored in this thesis. By estimating the features' affine shape, the estimated transformation parameters are used to resample the image patch surrounding detected features. In this way, the appearance of resampled feature patches is less influenced by the viewpoint and viewing direction change of images. To avoid over-parametrization, for a feature containing specific contents, the estimated affine shape parameters should be unique. This thesis utilizes some constraints for learning the feature affine shape. Consequently, the whole network strives to reach a canonical feature patch by optimizing the training loss built based on the utilized constraints.

Similarly, to estimate the orientation of local features, a unique solution should exist for each input feature, so that the estimated rotation can be used to resample the input patch to its canonical form. This, too, is explored in this study and a corresponding loss in the feature orientation stage is proposed so that the final predicted orientation can be used to resample the patch to its canonical form. In addition, the affine shape and orientation of features should be jointly solvable. However, the existing works failed to jointly estimate those two modules. In this thesis, the issue of jointly learning those two parts is solved by using a combination of the loss used in the orientation and feature affine shape estimation network learning.

Feature description is a problem of learning a feature embedding. By using the learned feature embedding, the output features are more discriminative and robust against viewpoint and viewing direction change. This thesis uses the Siamese CNN architecture as its starting point and explores ways to use the training data in a better way. With regard to application, this thesis uses oblique aerial image blocks as test datasets. The learning based modules are combined and tested on the orientation of those image blocks. Also, the proposed method is compared to more traditional algorithms and other deep learning based variants.

## 1.2. Main Contributions

In this thesis, feature description, feature orientation and affine shape estimation are studied. Each of those modules are first solved separately, by training corresponding neural networks. Then, the trained networks

are combined to detect and describe features from input images. Also, the trained networks are assessed with a real image orientation task using images containing large viewpoint and viewing direction change. The main contributions of this thesis are as summarized here:

- To train the feature descriptor, a Siamese CNN is employed. A novel weak match branch is proposed to actively find feature patches which have undergone a slight geometric transformation with regard to a feature patch, but which lie far away from the feature patch in the descriptor space. Then, the found patches are integrated into the descriptor learning framework. Consequently, the intra-variance of the appearance of matched features is explored, strengthening the invariance of feature descriptors against viewpoint and viewing direction change.

- For feature orientation and affine shape estimation, three major contributions are provided. First, to train the orientation of a local feature, a novel feature orientation learning architecture is proposed. Instead of using Siamese CNN branches to train the feature orientation, the architecture proposed here uses only a single branch network. The rotation of a predicted patch is calculated using its mean gradient in x and y direction. In this way, a unique global minimum solution is present as long as the included feature patch does not contain symmetric patterns. Second, also to train the affine shape, only a single branch network is needed instead of using a Siamese CNN. The affine shape of a patch predicted by the affine shape network is measured by the skew and stretch calculated using the second moment matrix. Similar to feature orientation, a unique global minimum which corresponds to the canonical feature patch is derived. Third, affine shape and orientation are learned simultaneously in one network. To the best of the author's knowledge, this is the first time that those two modules have been reported to be successfully learned simultaneously.

- Concerning applications this thesis makes use of oblique aerial image blocks to test the real performance of the feature matching method composed by the proposed learned modules. In these oblique aerial image blocks, challenging viewpoint and viewing direction change are included. Additionally, different types of landscape are studied and compared. First, the experiments based on image orientation tasks serve to check the performance of the proposed image matching method. Second, the image orientation tasks conducted in the experiment are seen as a pilot study which allows the extension of the proposed method to larger and more complete aerial image blocks in which challenging viewpoint and viewing direction change are included, e.g., oblique aerial image blocks.

## 1.3. Thesis Outline

The thesis is structured as follows. Chapter 2 introduces the theoretical groundwork the proposed feature matching method relies on, including the steps in a feature based image matching algorithm, the general and concise concept of Convolutional Neural Network (CNN) as well as Siamese CNN. Chapter 3 contains the state of the art. The present work on feature detection, orientation, affine shape estimation and description is

reviewed in Sections 3.1 through 3.3. A typical application that deals with the orientation of oblique aerial images is reviewed in section 3.4. The research issues are identified in section 3.5. In the next chapter, the main contribution of this thesis is explained in detail. In particular, section 4.1 provides an overview of the proposed feature matching method, followed by a demonstration of the proposed feature description, feature affine shape and orientation as well as the full affine shape estimation module from section 4.2 to 4.5. Section 4.6 explains how the trained networks are combined into an inference pipeline that can detect and describe features of an input image. The model assumptions, theoretical limitations as well as other closely related issues of the method proposed in this thesis are discussed in Section 4.7. The results for four different experimental tasks designed in this thesis are shown in Chapter 5. The implications of the results as well as the advantages and limitations of the methodology are discussed in Chapter 6. Finally, conclusions are drawn and future work directions are sketched in Chapter 7.

# 2. Basics

In this chapter, necessary basic knowledge for understanding the classical feature based image matching framework and deep learning feature based image matching are discussed. First, the concept of a classical feature based image matching pipeline is explained. Second, convolutional neural network (CNN) [LeCun et al., 1989] and Siamese CNN [Bromley et al., 1994] are illustrated, for both parts are used in descriptor learning and other related modules in this thesis to improve the local feature based image matching performance.

## 2.1. Feature based Image Matching

This section first provides an overview of the feature based image matching algorithm, followed by the desired properties for features and descriptors. Afterwards, the different components of a feature based image matching algorithm are explained.

### 2.1.1. Overview: What is Feature based Image Matching?

Feature based image matching is composed of five steps: feature detection, affine shape estimation, orientation assignment, feature description, and feature matching [Szeliski, 2010]. The basic pipeline of a feature based image matching algorithm is shown in figure 2.1. Before discussing each step in detail, a brief overview describing the process of feature based image matching is necessary.

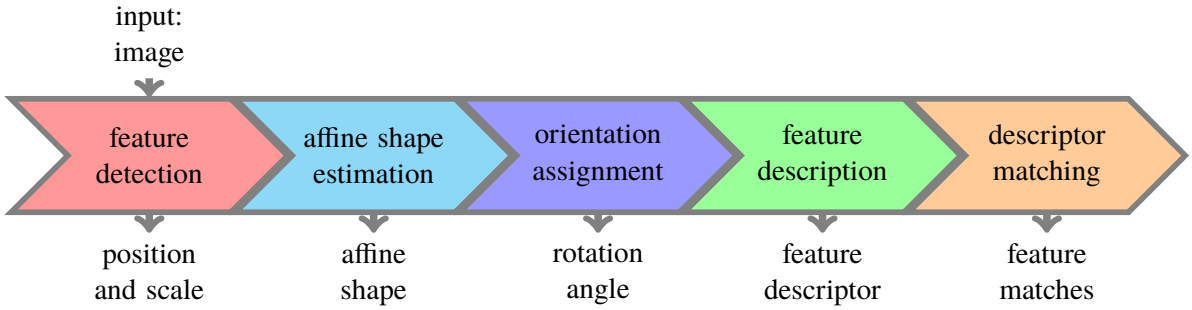



Figure 2.1.: Feature detection and description pipeline.

Local features are corners or blobs appearing in a distinctive position in the image, robust to small transformations between images. They are not necessarily salient image corners to the human eye, but they are distinctive based on some mathematical model. Since scale differences between overlapping images are common, in particular for close-range imagery, features are normally detected in scale space to retrieve their characteristic scale. Once those features are determined, their position and scale (size) are known.

In the next step, an affine shape correction of the feature is used to decrease the influence of skew and unequal scale in the two image coordinate axes. In this context, it is important to note that the affine transformation models perspective distortions and is typically required for large-baseline image pairs with convergent viewing directions. It is valid as an approximation of central perspective for small image windows. Subsequently, the principal orientation of the detected feature is estimated, taking into account different rotations of the two images around the viewing direction. Through those steps, the position, scale, affine shape and rotation of features are determined. According to these geometric elements, a (usually square) window around the detected feature is resampled to remove the geometric distortion, yielding a canonical description of the grey value neighbourhood. The size of the resulting feature support window is normally set to be several times its characteristic scale. This window is then used as input for feature description. During feature description, a high-dimensional feature vector is derived from the feature support window. This vector is used to represent the detected feature. Descriptors are normally designed to be invariant against a limited level of geometric and illumination changes. One example showing the application of these aforementioned steps is illustrated in figure 2.2.

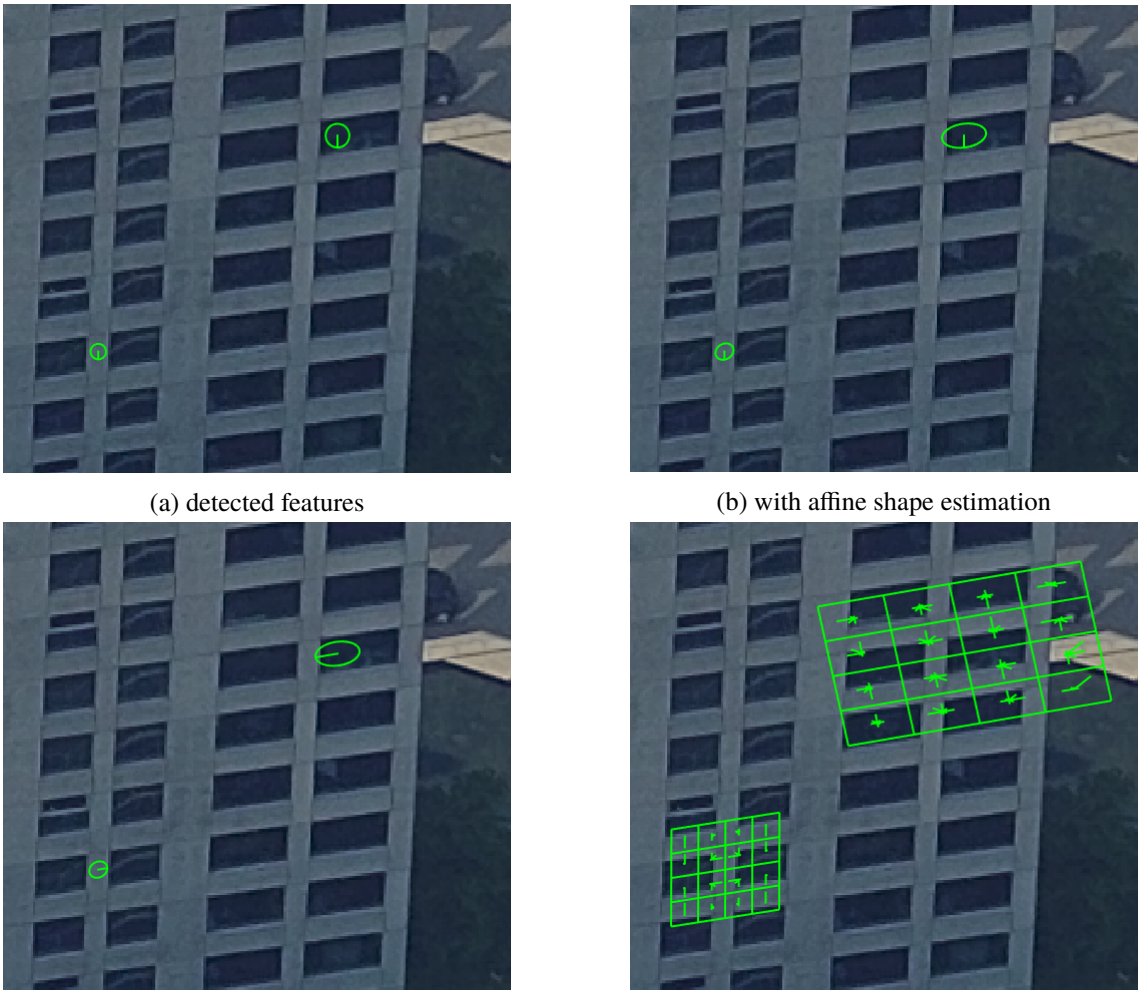
After descriptors are derived independently for each image, the correspondence problem can be cast as a neighbourhood search in the high dimensional feature space of the feature descriptors. Two related issues are essential: the similarity between potentially conjugate vectors and the computational complexity of finding these conjugate features. Based on the similarity measure, e.g., the Euclidean distance of the vectors, strategies like the search for the nearest neighbour, the ratio between the distance to the nearest and the second nearest neighbour [Lowe, 2004], and a distance threshold are employed to find matches. The problem of efficiently finding nearest neighbours in descriptor space is normally solved by indexing high-dimensional data, as seen e.g. in Beis and Lowe [1997].

### 2.1.2. Desired Properties for Detected Features and Descriptors

In order to understand how features are detected, it is necessary to first discuss the properties required of good local features.

#### What is a Good Local Feature?

In general, desired properties for a good local feature are distinctiveness, seldomness, repeatability, invariance, precise localization, and speed, as discussed in Barnard and Thompson [1980]; Förstner and Gülch [1987];



(a) detected features

(b) with affine shape estimation

(c) step b with orientation

(d) feature support window

Figure 2.2.: An example of features. (a) illustrates two selected detected features; green circles are used to show their positions and scales (radius of circle); (b) shows the detected features with affine shape estimation and the final features are then ellipses; (c) shows the assigned orientation for features after affine shape is estimated; (d) shows the feature support window extract with the correction from estimated affine shape and orientation with a range of six times the detected scale.

Shi et al. [1994].

A good local feature extracts points or blobs that are different from surrounding pixels. To yield useful results, the algorithm needs to avoid focusing on image areas containing pixels of homogeneous grey values. This property is called the distinctiveness of features and it forms the centre measure for most mathematical models of feature detection. Distinctiveness of features ensures that only features whose surrounding intensity patterns show high variance are selected. Also, distinctive features tend to be more easily matched, as they can be distinguished from other pixels in an image.

Another important factor that should be taken into consideration is the number of detected features. The complexity of feature matching is closely related to the number of features involved. When more features are

used for matching, more discriminative descriptors are preferable, largely owing to the fact that the descriptor space is more crowded as more features are described. However, several thousands of matched features may be too many to be adequate for many applications related to feature based image matching, e.g., estimating two-view geometry. Correspondingly, the number of detected features should not exceed a certain range; this is called seldomness of detected features. Compared to the number of pixels in an image, the number of detected features in this very image should be only a fraction. Therefore, detected features are considered compressed representations of an image. Normally, for an image of 1000 x 1000 pixels, only several hundred distinctive features are extracted.

When two images containing the same scene or objects are taken under different viewing conditions, e.g., from different viewpoints and with different illumination, in many cases the same detected features show up in the part both images have in common: They are repeated. This is called repeatability, which is a prerequisite for feature matching, on account of the fact that no matches between images can be retrieved when no corresponding features are detected in those images in the first place. In order to guarantee repeatability, feature detectors need to be designed invariant against geometric and radiometric deformations such as a wide baseline, rotation, etc., and robust to image noise, blur, and other factors that might conceal repetitive features. To achieve the necessary degree of invariance, mathematical models describing the underlying transformations are built. These are then used to design methods that are insensitive to the mathematical transformations [Tuytelaars and Mikolajczyk, 2008].

If the detected features are to be applied in further applications, such as calibrating cameras and 3D reconstruction, the location of these features should be determined as precisely as possible. Estimating the camera parameters and calculating the orientation between images relies on the precision of feature coordinates. In general, detecting features on edges, in this case, should be avoided. Along the edge, the variance of local image window is too low, making the localization of those features infeasible.

Another important aspect for feature detectors is their speed, especially for applications like real-time mapping, which are meant to be run with high processing speed. Improving the speed normally decreases the performance of features in one or several aspects, for example repeatability, robustness, or localization accuracy. Designing feature detectors for maximum efficiency therefore is a highly specific task that must take into consideration which properties of features may be sacrificed under which circumstances.

#### What is a Good Feature Descriptor?

A good descriptor should have discriminability, invariance and an adequate level of locality. However, discriminability and invariance can be competing properties when invariance requires descriptors to be stable facing deformations, whereas discriminability requires descriptors to be sensitive to the change of patch contents surrounding features. Invariance of descriptors is usually achieved by proper aggregation or integration, e.g, through histograms of feature response. Discriminability, in turn, is often acquired by amplification of the changes, i.e., by application of differential methods such as calculating gradients.

Another pair of competing properties is the distinctiveness and locality. A larger support window improves distinctiveness. However, using larger windows also increases the probability of introducing content with occluded areas, which counteract the increasing discriminability of larger context windows. Moreover, the increasing range of feature support windows also decreases the locality of features.

### 2.1.3. Scale-Invariant Feature Detection

#### Feature Detection in Scale Space

The determinant of the Hessian matrix computed in image plane can be used to measure the distinctiveness of pixels under a specific scale [Mikolajczyk et al., 2005]. The Hessian matrix is defined as

$$H = \begin{bmatrix} L_{xx}(x, \sigma_D) & L_{xy}(x, \sigma_D) \\ L_{xy}(x, \sigma_D) & L_{yy}(x, \sigma_D) \end{bmatrix} \tag{2.1}$$

where $L_{ij}(x, \sigma_D)$ is the second order Gaussian smoothed image derivative in $i$ and $j$ direction with scale $\sigma_D$.

The determinant of the Hessian $DH$ measures the extent of the change in local image content with respect to scale $\sigma_D$ in two orthogonal directions. It is extended to multiple scales by changing scale factor $\sigma_D$, creating a scale-space. Within this scale space, the response of features is calculated on multiple scales. The response extrema in both image dimensions and in the scale dimension are collected as candidate features. The scale on which they achieve these extreme values are their characteristic scales. The localization is further refined by Taylor expansion of the scale space.

The trace of $H$ equals $L_{xx}(x, \sigma_D) + L_{yy}(x, \sigma_D)$, which is also known as Laplacian. Alternatively to the determinant of the Hessian, Laplacian is used to measure the distinctiveness of features. This is approximated by subtracting adjacent layers in Gaussian scale space [Lindeberg, 1994], which is known as Difference of Gaussian (DoG), used e.g. in Lowe [2004] as the detector of SIFT (Scale Invariant Features Transform). The Laplacian of Gaussian (LoG) responds when the sum of $L_{xx}(x, \sigma_D)$ and $L_{yy}(x, \sigma_D)$ is large. In this way, the danger of locating features on the edges where only one of the two second derivatives is large still exists. This is why an extra edge response elimination step is applied after DoG extreme extraction in SIFT [Lowe, 2004], and also in the Förstner operator [Förstner and Gülch, 1987].

Compared to the Laplacian of Gaussian, the determinant of the Hessian only responds strongly where large variations occur along any two orthogonal directions. It also penalizes the elongated structure where the second derivate along one direction is small [Mikolajczyk et al., 2005]. The determinant of the Hessian creates a more restricting condition and performs more reliably than the Laplacian when facing affine transformations [Lindeberg, 2015; Mikolajczyk et al., 2005] between images.

In order to be invariant against scale change, a scale space representation [Lindeberg, 1994] with different feature responses is constructed. The characteristic scale, as stated already, is selected for a point where its

feature response attains a local extremum over scales, namely larger or smaller than responses at adjacent finer and coarser scales. To achieve this, the Laplacian of Gaussian is used as response since it yields the best experimental result compared to other choices, such as Mikolajczyk and Schmid [2001]. In SIFT [Lowe, 2004], the scale selection uses Difference of Gaussian (DoG) response directly in DoG scale space, as DoG is an approximated version of LoG.

### 2.1.4. Feature Affine Shape Estimation

In order to cope with affine transformations in images, the affine shape of local features can be estimated to correct affine distortion of local features. The second moment matrix is defined as follows:

$$M = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} L_x^2(u,\sigma_D) & L_x(u,\sigma_D)L_y(u,\sigma_D) \\ L_x(u,\sigma_D)L_y(u,\sigma_D) & L_y^2(u,\sigma_D) \end{bmatrix} \tag{2.2}$$

where $L_x(u,\sigma_D)$ is image first order derivative in $x$ direction computed with Gaussian kernels on scale $\sigma_D$ in position $u.g(\sigma_I)$ is the Gaussian window function with standard deviation $\sigma_I$. The ratio of the two eigenvalues of $M$ measures the level of isotropy. As a result, $M$ is used to create a correction matrix which is then applied to normalize a local image pattern. A practical way of achieving this affine shape correction is to apply the inverse of the second moment matrix in an iterative way until the smaller and larger eigenvalues of $M$ are close to each other. This normalization brings the anisotropic scale change to a uniform change. A more detailed description of affine shape adaption theory is to be found in the appendix A.

### 2.1.5. Feature Orientation Assignment

Up to this point, the feature's position, scale and affine shape have been identified. This already determines the geometric frame of a feature, but only in rotation-free cases. To achieve the invariance against rotation for detected features, the next parameter to be detected is the orientation (rotation) of a local feature.

To solve this, a sampling pattern in a range proportional to the characteristic scale around the detected features is applied. One possible strategy for assigning the feature orientation using the sampling pattern is suggested in Lowe [2004]: the image gradients are quantized and a gradient histogram is obtained. Then, the peak of the gradient bin is chosen as the principal direction.

Completing all of the above steps, a feature's position $x$, $y$, scale $\delta$, two affine parameters $a_1$, $a_2$, and orientation $\theta$ are obtained. With the help of position and by using the inverse transformation determined by the scale, affine parameters and rotation, a patch surrounding each detected feature is resampled to a canonical patch representing a local feature. This canonical patch is then defined as the feature support window for calculating a descriptor, which, in turn, is a representation of the feature.

### 2.1.6. Feature Description

Calculating a descriptor for a feature is in essence a mapping process that uses an input feature support window to create a vector which represents the feature. The plot line for a descriptor design can be summarized as follows: transformation, aggregation, optional normalization and dimension reduction [Brown et al., 2011]. Transformation refers to the magnification of a signal, i.e., gradients in the SIFT descriptor [Lowe, 2004] or simple pattern comparisons. The result of this transformation is then aggregated by some statistical measures, e.g., a histogram of gradients (HoG) for the SIFT descriptor [Lowe, 2004] or a histogram of simple Haar wavelet response for the SURF descriptor [Bay et al., 2008]. Moreover, the calculated output each aggregation step creates can be normalized as well, and the dimension can be further compressed through algorithms like Principle Component Analysis [Ke et al., 2004].

Alternatively, this mapping process can be also conducted by a deep neural network, i.e., using a CNN which takes images as input and outputs a compressed feature vector through a series of linear and non-linear transformations. The advantage of using CNN is that the descriptor design problem can be transformed into a machine learning problem, as shall be explained in later sections.

### 2.1.7. Descriptor Matching

When descriptors are obtained, matching is converted into finding correspondences in a high dimensional feature space. The next question is how to measure the similarity of two descriptors. The distance, e.g., the Euclidean distance, between any two features in the space is often taken as a measure of their similarity. Shorter distance indicate higher similarity and higher distance, in turn, mean lower similarity. A k-d tree is normally built for space partition and thus data can be efficiently structured and indexed. Algorithms like best-bin-first [Beis and Lowe, 1997] are used to speed up the search for nearest neighbours in higher-dimensional space. There are, however, more ways to measure similarity, e.g., the use of the Hamming distance for binary descriptors or the learned similarity measure of two features.

After the descriptors of two images and the similarity measure for features are given, corresponding pairs, i.e., matches, can be found. For image 1 and 2 containing $N_1$ and $N_2$ descriptors, $Desc_{1i}, Desc_{2j}$ [$i \in \{1, 2, ..N_1\}, j \in \{1, 2, .., N_2\}$] represent their descriptors. Three common strategies for descriptor matching are:

- **Distance Threshold:** Descriptor pairs calculated from different images that lie below a certain distance are judged as matching pairs.
- **Nearest Neighbour:** Each descriptor's nearest neighbour, i.e., the closest descriptor within the descriptor space of different images is identified as a matched feature.
- **Nearest Neighbour Ratio:** The descriptor's nearest neighbour as identified in different images must be significantly closer than the second nearest neighbour. Considering the distance between the

feature and its nearest neighbour $d_{nearest}$ and that distance between the features to its second nearest neighbour $d_{2ndnearest}$, this means that if the ratio between $d_{nearest}$ and $d_{2ndnearest}$ is lower than a certain threshold, e.g., 0.8, the feature and its nearest neighbour are judged as a matching pair.

This thesis will not address the descriptor matching algorithm, but concentrates on the feature description, orientation and affine shape estimation.

## 2.2. Convolutional Neural Network (CNN)

A Convolutional Neural Network [LeCun et al., 1989] is a kind of neural network which contains multiple layers of convolution, activation, pooling and, optionally, fully connected layers. In fully connected neural networks, a transformation connects each input and output unit, whereas a CNN shares the transformation parameters across the whole input image or feature maps by sliding the convolution kernel. This parameter sharing strategy decreases the number of parameters significantly.

### 2.2.1. Architecture of CNN

In CNN, convolution, point-wise non-linear activation, pooling and batch normalization form one layer. Before exploring how this layer is used in feature based image matching, the three aforementioned basic modules it contains are shortly explained.

**Convolution:** The input is first convolved with one or more convolution kernels, including bias. As more than one convolution kernel can be used, a corresponding number of feature maps can be obtained. Training parameters are the elements of the convolution kernel matrix and bias; super parameters are the size of the convolution kernel, as well as stride and padding. Stride indicates how far each step of sliding moves the kernel. Padding indicates if zeroes need to be added at the borders of input images before convolution.

**Activation:** Point-wise non-linear activation, e.g., Sigmoid function or Rectified Linear Units (ReLU) [Nair and Hinton, 2010] are applied to convolution outputs. This step brings non-linearity to the network and acts as the key to increasing the modelling complexity of CNN.

**Pooling:** Afterwards, a statistical operation, i.e., applying the average or maximum function in a local neighbourhood, may be used to down-sample the feature maps. This optional procedure is called pooling. It compresses signals from lower level convolutions and decreases the spatial resolution of feature maps in both height and width. Alternatively, increasing the stride of the convolution operation can also decrease the spatial resolution of feature maps in height and width.

**Batch Normalization:** Batch Normalization [Ioffe and Szegedy, 2015] is another optional step widely used after each convolution operation to speed up the training process. The basic idea is to normalize the feature maps with mean and standard deviation, thereby preventing the feature maps from being influenced by covariate shift [Ioffe and Szegedy, 2015], which means that the distribution of variables differs from one dataset to another.

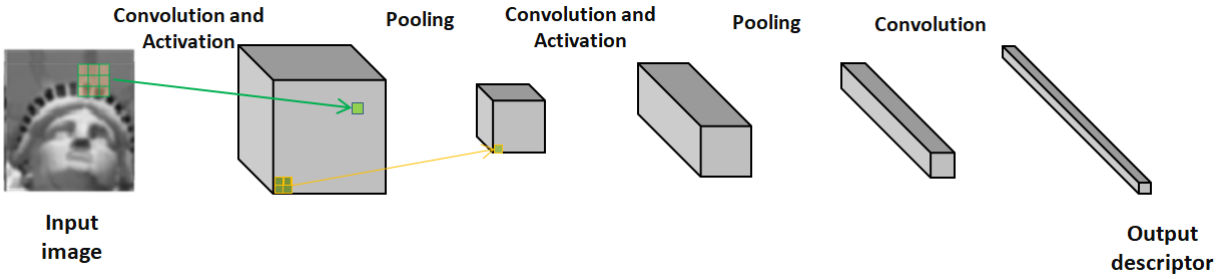



Figure 2.3.: Typical structure of a Convolutional Neural Network (CNN) for descriptor learning. The input images are first convolved (shown in green input and output in the first feature map) and activated by the point-wise non-linear activation function, followed by a pooling operation (as shown by the orange input grid in the first feature map, and the output grid in the second feature map) to decrease spatial resolution. Another layer of successive convolution-activation-pooling operations is applied, and after the third convolution layer, the output feature map is compressed into a high-dimensional vector where the width and height of the output feature map decrease to one.

An example of this architecture used in descriptor learning is shown in figure 2.3. More than one layer is applied to the input image, each sequence decreasing spatial resolution while increasing the depth of the feature maps involved as the network moves from lower layers to higher ones. The training parameters of the entire CNN are the convolution kernels in each convolution layer.

Finally, a loss function is defined. This function is often related to the task, e.g., it can be based on cross entropy loss for image classification problems, or on the norm distance of descriptors for feature descriptor learning. The learning objective for the entire network is to minimize the loss function through training.

### 2.2.2. Training of CNN

There are two basic forms of data flow in CNN: forward propagation and backward propagation. In between, the gradients of parameters are calculated and employed to update the training parameters.

**Forward Propagation:** The input image is fed into the network, and the output of each layer is computed in a successive way. The output of one layer is then taken as the input for its successive layer. The forward propagation ends when the loss is obtained. In supervised CNN learning problems, the loss is usually calculated with the ground truth labels of training samples, which are part of the training data. In forward

propagation, the data flows from layers, which are closer to input data, to those closer to the loss function.

**Backward Propagation:** After loss calculation, the partial derivatives of loss with regard to the training parameters of each layer are calculated. As the whole network is active in a nested way, the gradient calculation follows a chain rule. Specifically, it starts with calculating the gradients of loss with regard to the output of the final layer, then moves on to the gradients of loss concerning the training parameters of the second-to-last layer until it reaches the gradients of loss connected to the training parameters in the first layer. In this way, the gradients of all unknowns (training parameters in all layers) are calculated. As the gradient computation propagates from the output of a network to the input layer of the network, this process is called backward propagation. Once backward propagation is complete, the partial derivatives of loss with regard to all of the training parameters in the network are all completed. Those gradients are further used in the parameter updating step.

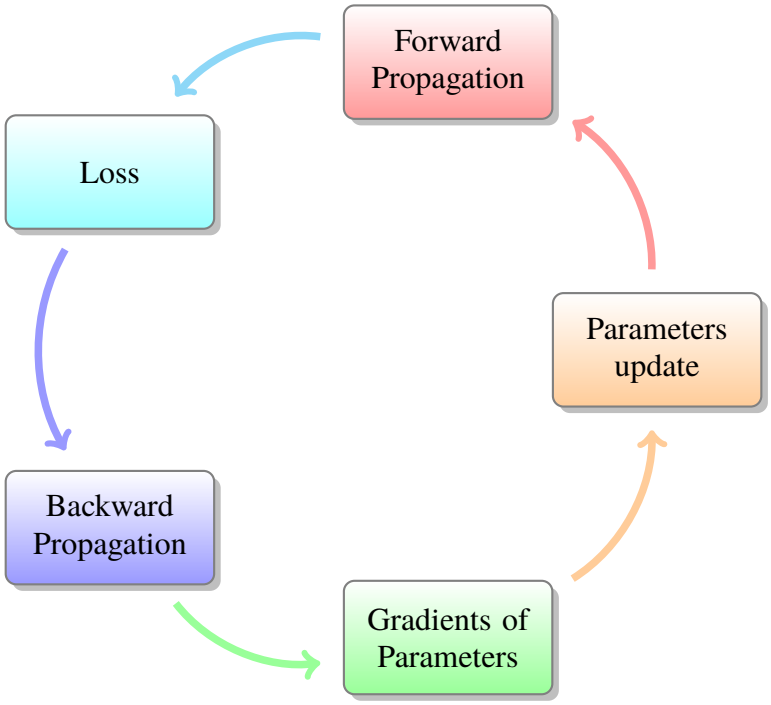



Figure 2.4.: Running circle of the CNN training algorithm. Through forward propagation, loss is calculated, followed by backward propagation to derive the gradients of training parameters, which are then used to update parameters according to the updating rules of the selected optimizer. A new iteration starts again, once the training parameters have been updated. This ceates the characteristic circular process of training loops shown in the figure, until the network reaches its stop criteria, e.g., the maximum amount of iteration steps.

**Parameter Update:** After obtaining the derivatives of loss with regard to training parameters, those training parameters are updated following the rules of optimizers. One standard among these is the so-called standard gradient descent, which subtracts the gradient of corresponding parameters from the current parameter by

application of a small learning rate. This updating aims at "moving" parameters to a place where the loss decreases. However, this is not always guaranteed because of the highly non-convex shape of the loss function. The updating rules contain many variants, e.g., gradients with momentum, adaptive momentum, or Nesterov gradient descent [Nesterov, 2013].

The whole training process is circular, as shown in 2.4. This circle can run individually for each training sample. On the one hand, the gradients computed from only one training sample are highly variable and sensitive to noise. On the other hand, averaging gradients from all training samples (full-batch) in one training iteration is not realistic because memory is always a limited resource in computation. Instead, mini-batches containing a smaller number of training samples, i.e. 256 or 512 training samples per batch, are used for each iteration. Apart from that, the learning rate is an important hyper parameter to set, as it basically tells the optimization how far each step should "move" the parameters. Setting a too large learning rate renders the approximation of the loss function curve and the gradients at current estimated value unreliable, whereas a too low learning rate significantly increases the computation time to reach a feasible solution.

In order to reduce the risk of overfitting, a fraction of units (activation) are randomly assigned to be zero and their connections (linear combination) to the units in the next layer are dropped in the training stage. This is called dropout. When the trained model is applied during test phase, all the connections are used, but reduced by the fraction factor that has been used for switching off neurons in training stage. This strategy is later used in all the networks related to the methods proposed in this thesis.

## 2.3. Siamese Convolutional Neural Network

For learning descriptors based on matching and non-matching pairs, Siamese Convolutional Neural Networks have proven to be very suitable. "Siamese" refers to the fact that two branches of CNN feature extractors, one for each input sample containing complex differences, share the same parameters. Already in Bromley et al. [1994], Siamese CNN were proposed to extract descriptors for the verification of human signatures. Their structure aims at learning a function by mapping from an input image patch to a feature vector, In this context, Siamese CNN are able to recognize the writing style of individuals and make the output feature vectors be invariant against the differences between signatures provided by a same person. By completing both branches, two descriptors are obtained. The loss function is constructed based on the similarity of output descriptors. A typical Siamese network is shown in figure 2.5.

By calculating the descriptors for both input patches, namely $Desc_1$ and $Desc_2$, the distance metric of them is obtained. Further loss is built to ensure that the descriptors of matched (same class) patches can move close to each other, and the descriptors of unmatched patches to be far from each other. Since complex transformations can be obtained in the pairs for matched features, high-level invariance is expected to be achieved through training within this framework. This framework is widely used in applications to include complex transformations in the data, e.g., pedestrian re-identification [Chung et al., 2017; Blott et al., 2019],

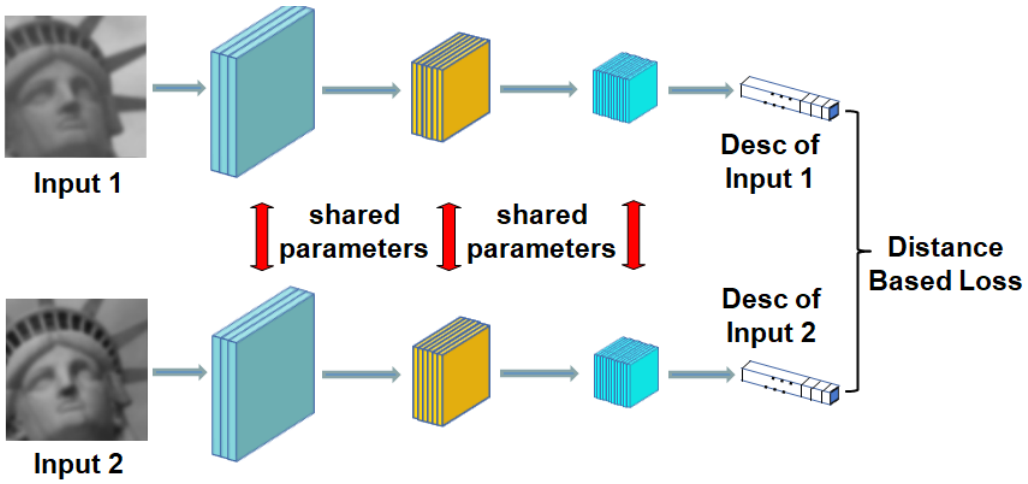



Figure 2.5.: A typical Siamese Convolutional Neural Network (CNN). The input images normally contain the transformation the network is learning and will become invariant against after training. Through two branches of CNN with shared parameters, the obtained descriptors $Desc_1, Desc_2$ are constrained based on their distance. This causes the learned descriptors to be invariant against the transformations contained in the input data.

perdestrain tracking [Nguyen and Heipke, 2020] and feature based image matching [Chen et al., 2016; Zagoruyko and Komodakis, 2015].

In this thesis, two types of input image pairs are used for training the descriptor. One is a matched image pair containing transformations, as shown in figure 2.5. The other type of input used are unmatched pairs of images containing dissimilar features, for which descriptors correspond to different coordinates in high-dimensional space and should be far away from each other.

# 3. Related Work

As introduced in section 2.1, the feature based image matching framework contains feature detection, affine shape estimation, feature orientation, feature description and descriptor matching. With the exception of feature matching, these different parts are reviewed in this chapter. Descriptor matching algorithms are not reviewed for two reasons: Firstly, compared to other modules, the descriptor matching solution used in this thesis is relatively standard. Secondly, focussing on descriptor matching would only distract from the main focus of the research documented in this thesis. This chapter, therefore, offers an overview of state-of-the-art local image feature detection in section 3.1, and a review of feature orientation and affine shape estimation in 3.2, while description is discussed in section 3.3. A brief review of the applications of feature based image matching in the orientation of oblique aerial images is presented in section 3.4. The final section of this chapter, 3.5, revisits the findings presented in previous research and summarizes open questions. Part of this chapter have been published in [Chen et al., 2021].

## 3.1. Local Feature Detection

### 3.1.1. Translation and Rotation Invariant Features

The development of so-called interest operators to detect the position of features can be traced back to the Morevac detector [Moravec, 1979] and the Dreschler operator [Dreschler and Nagel, 1981]. Moravec [1979] assesses average quadratic gradients in the four main directions (horizontal, vertical and both diagonals) of a local window. If the minimum of these four values is larger than a given threshold, the window centre is chosen as an interest point. This reflects the simple idea that a local feature should differ from its surroundings. Dreschler and Nagel [1981] on the other hand determine pairs of maximum and minimum curvature of the grey value function in the vicinity of corners. The interest point is then defined as the zero crossing of the curvature between the two points.

The Moravec detector is not rotation invariant because gradients are estimated in four pre-defined directions. A better idea is to analyse the auto-correlation matrix (also known as second moment matrix) $M$ [Lucas et al., 1981]. The two eigenvalues of $M$ contain information on the curvature of the grey value function. If both of the eigenvalues are small, the region does not show much grey value variation. If one eigenvalue is large and the other one small, a strong change in one direction and thus an edge is present. If, however,

both eigenvalues are large, then the local correlation function peaks sharply and represents either a corner or a certain circular signal [Föstner, 1991]. Förstner calculates the eigenvalues based on the inverse of the auto-correlation matrix and suggests two indicators related to the size and roundness of the two eigenvalues [Förstner and Gülch, 1987]. Harris and Stephens [1988] propose a cornerness value, which is computed as $Det(M) - \alpha Trace(M)^2$, where $\alpha$ is a variable balancing determinant and trace. Interest points are found by comparing the computed cornerness with a threshold. According to Rodehorst and Koschan [2006], the Förstner operator behaves slightly better than the Harris operator in terms of localization accuracy, detection and repeatability rate. Instead of the auto-correlation matrix, the Hessian matrix can also be used to detect features [Lindeberg, 1998]. Based on the determinant and trace of the Hessian matrix, feature selection criteria similar to the auto-correlation matrix are derived.

The detectors discussed so far are all based on local shift-invariant windows and therefore invariant to translation of the images. In addition to this, the use of eigenvalues instead of grey value changes in the direction of the image coordinate axes x and y makes both the Förstner and the Harris operator invariant against rotations. Detectors based on auto-correlation and the Hessian matrix are also robust against small scale change. However, with increasing scale change the performance considerably drops [Rodehorst and Koschan, 2006; Aanaes et al., 2012]. As is widely known, however, scale differences are common, especially in close-range photogrammetric applications or photo community collections, e.g., downloaded from the internet [Agarwal et al., 2011].

### 3.1.2. Scale Invariant Features

Multi-scale interest operators detect features on multiple scales, and then match them across scales. An example for this process is shown in Brown et al. [2005]. However, this approach only works if the scale difference between images is not too large or the scale ratio is approximately known *a priori*. A more advanced method is to analyse the features using scale-space theory [Lindeberg, 1998] which describes the scale space at some scale $t$ as a convolution of the original image with the two-dimensional Gaussian function with a variance of $t$. When changing $t$ continuously rather than in discrete steps, scale becomes a variable of a function that maps the original image to scale space. The sum of the second-order derivatives of the Gaussian function in x and y direction, i.e., the Laplacian of Gaussian (LoG) is used to compute local extrema in scale space and those extrema are selected as features, which are now scale-invariant.

In the scale invariant feature transform (SIFT) [Lowe, 2004], the LoG is approximated by the Difference of Gaussian (DoG). Sub-pixel localisation is obtained by fitting a local 3D quadratic to the surroundings of the extrema in scale space. Today, SIFT is one of the most well-known operators for feature detection (and description, see below) and performs well in matching images with scale change. Furthermore, SIFT can also tolerate a certain amount of affine transformation between images.

In Mikolajczyk and Schmid [2004], a scale selection mechanism is added to the Harris corner detector; the LoG over scale is evaluated at each detected Harris point and those points for which the LoG is an extremum

are preserved, followed by an optional iterative refinement for both scale and position. The Hessian Laplace detector [Mikolajczyk, 2002; Mikolajczyk and Schmid, 2004] is similar in spirit to this work, but starts from points detected using the Hessian matrix.

### 3.1.3. Detectors based on a Comparison of Grey Values or Saliency

In this category, the grey value of the central pixel is compared with that of the pixels in its neighbourhood. If the difference is lower than a given threshold, the pixels are considered to be similar. In SUSAN (Smallest Univalue Segment Assimilating Nucleus) [Smith and Brady, 1997], if the proportion of similar pixels in a local neighbourhood is a local minimum and below the threshold, the central pixel is selected as a feature. Another approach, FAST (Features from Accelerated Segment Test) [Rosten et al., 2010], uses machine learning to accelerate the comparison process. Since comparisons are only run on discrete pixels, the localization accuracy cannot be refined to sub-pixel level. This category of operators is primarily employed in applications for which speed is essential, but high localization accuracy is not required.

### 3.1.4. Detectors based on Machine Learning

Due to different illumination conditions, the 3D shape of the object surface and potentially complex reflection functions, analysing grey value differences between images using explicit mathematical transformations or designing features in an intuitive manner may become infeasible. An alternative is to consider feature detection as a machine learning task.

To take into account changes in illumination, a regressor is trained to predict a score map whose maxima are points with high repeatability in spite of challenging illumination changes, as suggested in Verdie et al. [2015]. Afterwards, the features that have proven stable against illumination changes are extracted by non-maximum suppression. LIFT (Learned Invariant Feature Transform) [Yi et al., 2016b] contains similar ideas for detectors, but motivates the detection to have good global discrimination between matched and non-matched feature pairs.

The core idea of the covariant detector is that features detected in the original image patch and then transformed using some geometric transformation, should be identical to features detected after applying the geometric transformation to the original image patch. This is used as the covariance constraint of Lenc and Vedaldi [2016], in which a regressor is employed as a detector to map an image patch to a feature.

An input image can be converted to a response map through trainable models. Then, features are detected as the top/bottom quantiles in those response maps. As considered in Savinov et al. [2017], the order of those top/bottom quantiles should be kept constant before and after the input image is geometrically transformed. A quad-network, composed of two original and two transformed image patches, is used to learn an order-preserving feature detection network. As mentioned above, Rosten et al. [Rosten et al., 2010] use machine

learning to improve the speed and repeatability of feature detection based on the grey value comparison of pixels in the neighbourhood.

## 3.2. Feature Orientation and Affine Shape Estimation

After a feature is detected with location and scale on the image plane, an image patch surrounding the detected feature is typically extracted with a size proportional to its scale. This patch reflects the appearance of the underlying feature and is used as input for descriptor computation. However, due to potentially different rotations around the viewing direction and also different viewing directions, patches of conjugate features can be distorted with respect to each other. As mentioned above, these distortions can be modeled as rotation and affine distortion, i.e., as skew and scale differences between the two axes of the image coordinate system.

Simply requiring the descriptor to be invariant against these relative distortions of the feature support windows decreases the discrimination power of the descriptors. Alternatively, the rotation difference can be determined by finding a principal orientation for each image patch surrounding a detected feature, and computing the descriptor based on that principal direction. Similarly, the affine shape can be estimated. The patches are then resampled to compensate for rotation and affine distortion between image patches.

### 3.2.1. Orientation Assignment

Once features have been detected, the orientation of a feature can be estimated by calculating a principal direction using the gradients calculated in a local window surrounding the detected feature. In SIFT [Lowe, 2004], a histogram of oriented gradients is calculated; the bin with the maximum count is then chosen, and the corresponding direction is refined by fitting a parabola to the peak and adjacent bins. Other bins with high values, i.e., larger than 80% of the maximum bin, are retained as secondary principal orientations. Features with multiple peaks in the support window can be better matched in this way. In SURF (Speeded-Up Robust Feature) [Bay et al., 2008], Haar wavelet responses in horizontal and vertical direction inside a circular window surrounding the detected feature are computed and plotted as points in 2D. Responses within a rotating cone of size $\pi/3$ are summed and the principal direction is assigned to the cone direction with the highest result. Also, the mean gradients in x and y direction in a small window surrounding the detected feature in the image plane have proven to be helpful in aligning features [Brown et al., 2005].

In Yi et al. [2016a] and Yi et al. [2016b], orientation is estimated by deep learning. The principal direction for an input patch surrounding a detected feature is predicted by a Siamese convolutional neural network (CNN), which maximizes the similarity of descriptors calculated for input matched feature patches. A similar idea is used in Mishkin et al. [2018] and Chen et al. [2020a] to learn the orientation of local features. This strategy achieves significantly better performance than the aforementioned methods based on handcrafted features.

### 3.2.2. Affine Shape Estimation

Detecting local features in scale space and assigning them an orientation is basically equivalent to normalizing rotation and scaling of local features before description. However, this transformation is not sufficient to model the geometric transformations between local image patches in case of large changes in viewpoint and viewing direction between images. Therefore, perspective changes, which for small windows can be compensated by an affine transformation, should also be estimated and taken into account before feature description.

Invariance against affine transformations has also been investigated. The method of Edge-Based Regions (EBR) [Tuytelaars et al., 1999] uses edges starting from detected Harris points to construct affine invariants. This method can only be applied to features surrounded by edges, limiting its range of applications. The approach called Intensity Based Region (IBR) [Tuytelaars and Van Gool, 2000, 2004] starts from one detected feature point and constructs lines in different directions. In each of these directions, the line ends at the local maximum of grey value change in a pre-defined neighbourhood. Then, an ellipse is fitted through all end points, and, in turn, used to represent the underlying feature. In Matas et al. [2004], the watershed algorithm is used to find local extrema employed for ellipse fitting. As reported in Mikolajczyk et al. [2005], the features extracted in this way are sensitive to scale change.

Affine shape estimation theory using the second moment matrix is studied in Lindeberg and Garding [1997]; Baumberg [2000]; Mikolajczyk [2002]; Mikolajczyk and Schmid [2004]. Here, affine Gaussian scale space is generated by convolution of the image patch with a non-uniform Gaussian kernel, which is represented by a $2 \times 2$ covariance matrix $\Sigma$. As indicated in Lindeberg and Garding [1997], in an affine Gaussian scale space, the response of two image patterns related by a particular affine transformation $B$ will be equal if for the underlying Gaussian kernels $\Sigma_1$ and $\Sigma_2$ the following relation holds: $\Sigma_2 = B^T \Sigma_1 B$. The second moment matrix $M$ measures a feature's level of isotropy and is thus used to describe the covariance matrices, thus $M_2 = B^T M_1 B$. Based on that, an iterative procedure is proposed to derive features for which the shape is approximately circular. In Baumberg [2000], the patch surrounding the features is normalised by multiplying the patch with $M^{-1/2}$ and then, the second moment matrix for the normalized patch is iteratively calculated and normalized with $M^{-1/2}$, until the two eigenvalues of $M$ for the normalized patch are close enough to each other. As a result, the affine transformation between two image patches is removed and only a rotation remains. In Mikolajczyk and Schmid [2004], this approach is extended to Gaussian scale space and Harris-Affine and Hessian-Affine detectors are employed to select the features.

However, a considerable amount of features is removed after the application of the iterative affine adaptation algorithm, because the ratio of the two eigenvalues remains to be too different from 1. According to Mikolajczyk and Schmid [2004], only 20-30% of the initially detected features are preserved for further feature matching. To tackle this problem, affine shape estimation based on a deep neural network is proposed in Mishkin et al. [2018], where the shape parameters are estimated by minimizing the distance between the matched descriptors. Affine shape is also learned and used to match images taken from oblique aerial

cameras in Chen et al. [2020a], in which a notable performance improvement is achieved when compared to handcrafted algorithms for removing affine distortion. ASIFT (Affine SIFT) [Morel and Yu, 2009], on the other hand, first simulates the input image with different versions of affine transformations. In a second step, the DoG features and SIFT descriptors detected in each transformed image are combined for descriptor matching, a process that makes ASIFT computationally expensive. ASIFT thus does not take algorithmic measures to estimate local affine shape for each feature, but makes matching more invariant towards affine distortions by applying a standard algorithms to different simulated views.

A literature summary graph for feature detection, orientation and affine shape estimation is shown in Figure 3.1.

## 3.3. Local Feature Description

In essence, the description of features is a problem of representation. It transforms the detected features into a new feature descriptor space, where different features can be more easily discriminated and matched. A local context window surrounding the feature is typically used to build the descriptor. The simplest descriptor is the combination of pixel grey values in this feature support window. However, this description is sensitive to photometric and geometric changes. In the new space, the description should be as invariant as possible to geometric and photometric transformations between images.

Designing descriptors is difficult because there are many factors influencing the grey values in the feature support window. A descriptor should be invariant to different types of limited change, e.g., illumination change, rotation or affine distortions. Therefore, the central challenge of descriptor design is to achieve invariance against those transformations.

Descriptors are divided into two groups: floating point descriptors and binary descriptors, based on the value type used. Floating point descriptors are designed for better discriminability, but they usually are computationally more expensive. Binary descriptors are designed for applications with sparse computational resources, e.g., real time SLAM or tracking.

The framework of descriptors can be summarized as containing several steps, as suggested by Brown et al. [2011]: transformation, aggregation, normalization, and dimension reduction. Transformation is often achieved by a basic filtering operation, such as calculating gradients or the Haar feature response. Filters are usually designed to preserve and magnify some special local patterns. They can be applied either to the whole feature support window or only to special positions. Aggregation comprises an integration procedure, e.g., maximum or mean value computation or histogram generation. As it is based on small regions in the feature support window, the arrangement of those small regions, i.e., pooling, needs to be determined a priori. Aggregation also increases the robustness against noise and provides invariance against a limited level of transformations between images. Normalization transforms a particular value to a fixed range and eliminates the influence from the absolute response. This improves the robustness against illumination change. As some

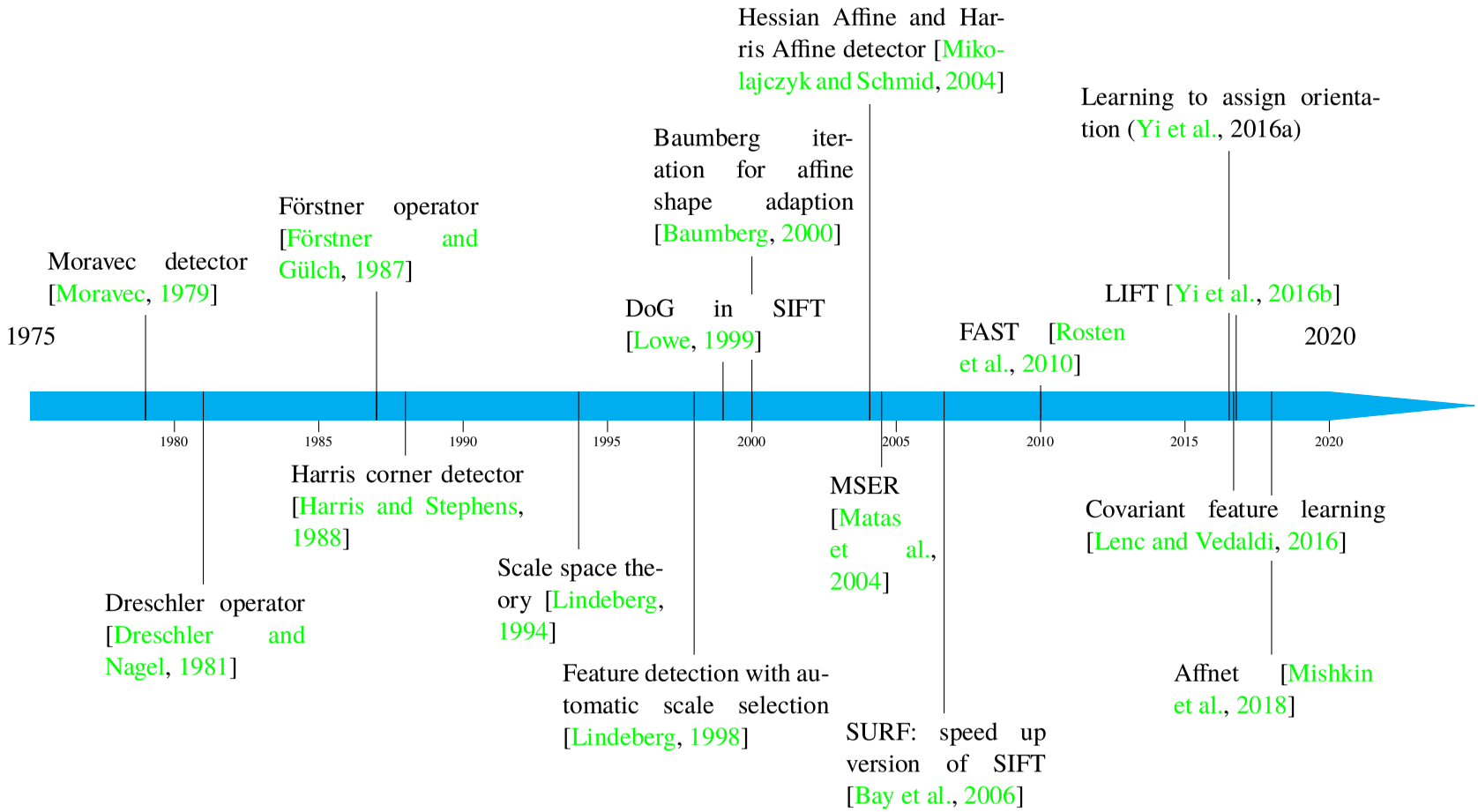



Figure 3.1.: A timeline highlighting some landmark papers for feature detection, orientation assignment and affine shape adaption.

of the descriptors are high-dimensional and different dimensions can have strong correlations, e.g., due to spatial correlation in the original feature support window, dimension reduction such as principle component analysis is applied to a number of descriptors in order to obtain a more compressed and less redundant description.

In this section, handcrafted descriptors are reviewed in subsection 3.3.1, while the machine learning based descriptors are discussed in subsection 3.3.2.

### 3.3.1. Hand Crafted Descriptors

#### Classical Feature Descriptors

This group contains the well-known descriptors SIFT (scale invariant feature transform [Lowe, 2004]) and SURF (speeded-up robust feature [Bay et al., 2008]). These descriptors integrate a large scope of knowledge concerning feature description which researchers had already accumulated before SIFT and SURF were proposed. SIFT first calculates the gradients in the feature support window, followed by Gaussian filtering to assign the central pixel a larger weight, and then aggregates the gradients in square grids. In contrast to this, SURF uses the Haar wavelet response as basic transformation, followed by a process of aggregation that also takes place in grids. Variants of these descriptors are DAISY [Tola et al., 2009], which uses steerable filters and aggregates in circular patterns, and PCA-SIFT [Ke et al., 2004], which decreases the correlation between SIFT descriptor dimensions by employing principle component analysis (PCA)where the projection basis is trained using a large number of collected SIFT descriptors.

#### Binary Descriptors based on Gray Value Comparison

Comparing pairs of pixel grey values inside the feature support window, then storing the result as a binary number forms the basic creation steps for binary descriptors. BRIEF (Binary Robust Independent Elementary Features, [Calonder et al., 2010, 2012]) employs different pixel positions in the smoothed feature support window under different distributions. ORB (ORiented BRIEF [Rublee et al., 2011]) adds orientation estimation and computes the descriptor in a greedy search selecting 256 pairs that can best discriminate homologous features in a training set.

BRISK (Binary Robust Invariant Scalable Keypoints [Leutenegger et al., 2011]) identifies the orientation of keypoints detected in scale-space and conducts the grey value comparison in concentric circles, a pattern similar to DAISY. The radius of the circles increases with the distance between the sampling point and the centre of the patch. The comparison is performed between the pairs of circles, the distance of which must be smaller than a threshold (short-distance pairing). Finally, FREAK (Fast REtinA Keypoint [Alahi et al., 2012]) selects the pairs for comparison based on knowledge of the human retina. FREAK also samples in a circular pattern, but comparisons concentrate in the region near the centre of the feature support window.

### 3.3.2. Machine Learning based Descriptors

The design of a descriptor can be defined as a machine learning problem in which the descriptor is a model with trainable parameters and the learning objective is to increase the similarity of conjugate pairs of features while decreasing the similarity of non-conjugate pairs. The way of mapping the feature support window to the descriptor space can vary widely. To provide an overview of different methods, some commonly used types of models are reviewed in the remaining part of this section.

#### Transformation-embedding-pooling Form

Descriptors trained for feature based matching were first proposed in Winder and Brown [2007]; Brown et al. [2011], in which different combinations of transformation, embedding and pooling are learned jointly to achieve a discriminative descriptor. The experiments of these authors indicate promising performance improvements compared to hand-crafted features. However, for each of the four aforementioned parts, a separate loss is designed, which leads to difficulties in the optimization of the model. Consequently, rather complex optimizing strategies are needed to find a feasible solution. To tackle this problem, a convex version objective function which notably improves performance was proposed in Simonyan et al. [2014].

Another important contribution to descriptor learning by Winder and Brown [2007]; Brown et al. [2011] is a training dataset, the so-called Brown dataset, which is widely used in later works. Within the community dealing with feature description, the term "Photo Tourism" dataset [Snavely et al., 2008] is sometimes used as a synonym for the Brown dataset, although the former contains many more images that the latter, which was derived from the original set. The Brown dataset relies on 3D reconstruction from multiple view images. For each image in the dataset, dense stereo matching and image orientation results are utilized to retrieve ground-truth matches[1]. Therefore, the training data contains realistic uncertainties for the conjugate features.

#### Comparison based Feature Descriptors

In Lepetit and Fua [2006], multiple random trees are trained to recognize matched features, using grey value comparisons at different positions of the feature support window as node tests. The so-called random fern [Ozuysal et al., 2010] uses a naive Bayesian combination of classifiers to achieve even better performance. Another category of descriptors is based on boosting. In Trzcinski et al. [2013, 2015], boosting is used to select carefully designed weak features which rely on grey value gradients over rectangular image regions. Then, each of those features is compared to a trainable threshold and converted into a binary value that forms one dimension of an output descriptor. Chen et al. [2014] learn Haar-like features which best classify the

[1] For a feature $f_L$ in image $I_L$, a small grid surrounding $f_L$ in $I_L$ is extracted and transferred to image $I_R$ through the depth map estimated from the stereo image pair $I_L, I_R$. The transferred grid is then used to estimated the scale and pixel localization for the transferred feature point in $I_R$. If the difference between the estimated scale and the pixel localization for the transferred features point is close to the scale and localization of a feature point $f_R$ in $I_R$, then $f_R$ and $f_L$ are considered to be a ground truth match.

matching and non-matching pairs using adaptive boosting [Viola and Jones, 2004]. Those works can also be classified as binary learned descriptors since they deliver descriptors in a binary form.

#### Deep Neural Network based Descriptors

For learning descriptors based on matching and non-matching pairs, Siamese convolutional neural networks (CNNs) have proven to be very suitable. As already mentioned, "Siamese" refers to the fact that two branches of CNN feature extractors - one for each input sample - share the same parameters. Already in Bromley et al. [1994], Siamese CNN was proposed to extract descriptors for the verification of human signatures. A Siamese CNN is also used in Hadsell et al. [2006] to extract a compact descriptor for the recognition of digits. In descriptor space, identical digits form clusters, removing the effect of appearance change caused by different writing styles and thus achieving invariance for representing those digits.

Jahrer et al. [2008] treated a Siamese CNN as a recognition network which outputs a class label for the support window of each detected feature in one image. Thus, the number of classes is determined by the amount of detected features in the image. Geometric transformations are simulated for both branches of the Siamese CNN to encourage the CNN to become invariant against geometric distortions. However, when matching images from a new scene, the class labels change and thus the Siamese CNN must be retrained.

Similar to the changes of identical digits written by different people, feature support windows of conjugate features from different views can also contain complex geometric and/or radiometric differences against which the descriptor should be invariant. Therefore, Siamese CNN are especially suited to the task of feature matching. Among the first, if not the very first, to use a Siamese CNN to train descriptors for feature matching, is Osendorfer et al. [2013], although this work concentrates on comparing four different types of loss functions. Carlevaris-Bianco and Eustice [2014] employ a Siamese CNN to achieve illumination invariance. Images with severe illumination changes are fed into the Siamese branches. The invariance obtained in this way exceeds that of hand-crafted descriptors. Siamese CNN as a way to learn descriptors are further used in Zagoruyko and Komodakis [2015]; Han et al. [2015]; Simo-Serra et al. [2015]; Chen et al. [2016].

Instead of measuring the similarity of descriptors with Euclidean distance, an additional metric network can be attached on top of a learned descriptor to directly predict the probability of a correct match for an input patch pair [Zagoruyko and Komodakis, 2015; Han et al., 2015]. Not surprisingly, the work based on metric learning performs better in discriminating feature pairs, as its similarity measure is not a simple distance measure but a more advanced trained similarity score. However, for an actual matching task, the metric network needs to be run for every possible combination of feature pairs computed from the different images. As a consequence, the computation is expensive and its usage is restricted. Therefore, most of the work concentrates only on learning the descriptors, i.e., on finding a good embedding that can discriminate features by simple distance measures. As stated in Simo-Serra et al. [2015], most of the negative pairs cannot contribute to the loss used for descriptor learning after the training process has run for a while. The solution

given in Simo-Serra et al. [2015] is hard mining, which means that only a small number of the samples that contribute a higher amount of loss are selected for parameter updating in each iteration during training.

The triplet architecture proposed by Chechik et al. [2010], was first used for training descriptors in Kumar et al. [2016]. Triplet networks are composed of three branches, namely an anchor ($a$), a positive ($p$), and a negative ($n$) branch. Anchor and positive branch correspond to a matching pair, anchor and negative branch to a non-matching pair. The distances $d(a,p)$ and $d(a,n)$ are used to build the loss function. Compared to a Siamese CNN that optimizes matched and unmatched parts independently, the triplet architecture pushes unmatched features away from similar features in descriptor space, thereby working equally for similar and dissimilar features. The triplet loss used in Hoffer and Ailon [2015] suggests that $d(a,n)$ should not be larger than $d(a,p)$. Furthermore, Balntas et al. [2016b] apply a margin between $d(a,n)$ and $d(a,p)$, which is identical to the suggestion given in Chechik et al. [2010]. A soft version of negative loss is proposed in Balntas et al. [2016a]: It is created by using the smaller one of $d(a,n)$ and $d(p,n)$ as negative loss.

Instead of checking each triplet separately, Kumar et al. [Kumar et al., 2016] build a global loss function to separate the distribution of distances for matched and unmatched pairs. In a global loss function, the variance of the distances for matched and unmatched pairs is minimized, as is the mean of the distances for matched pairs, whereas the mean for unmatched pairs is maximized. The authors test the proposed loss with different architectural details and achieve noticeable improvements compared to normal loss.

One of the major concerns in Siamese and triplet CNN for descriptor learning is that typically very few unmatched pairs are seen in training, which is in contrast to the typical application scenario for a trained descriptor, such as feature matching or retrieval, where much larger numbers of unmatched pairs need to be checked in comparison to matched pairs. To improve the situation, progressive sampling in L2-Net [Tian et al., 2017] uses the hardest unmatched pair to calculate the loss. The main loss is the ratio between $d(a,p)$ and $d(a,hardestn)$, with a smaller distance as desirable outcome for a matched pair and a larger one for an unmatched pair. Another constraint is the need to minimize the correlation between different dimensions of descriptors. Also, the similarity of feature maps in the descriptor network is encouraged to be high for matched features, but low for unmatched pairs. L2-Net achieves a remarkable performance improvement. Similar ideas of finding the hardest unmatched pair are explored in Mishchuk et al. [2017], where the closest non-matching patch to $a$ and $p$ in the triplet is found and a margin between the distance for a matched pair and the closest unmatched pair is included in the loss. In Mishchuk et al. [2017], a slightly better result than the one yielded by the application of L2-Net is reported, while the aforementioned additional constraints used by L2-Net [Tian et al., 2017] are ignored.

Although the distance of a matched pair and the hardest unmatched pair in a triplet is restricted by either setting a relative ratio or a margin between them, it might result in a larger cluster radius for matched features in descriptor space. As reported in Keller et al. [2018], the distance for both matched and hardest unmatched pairs and the "soft" margin or ratio between the distance for matched and hardest unmatched pairs in triplets must be balanced in order to avoid a too-large cluster radius for matched features. The result of their method shows a consistent improvement compared with Tian et al. [2017] and Mishchuk et al. [2017]. Alternatively,

SOSNet (Second Order Similarity Network) [Tian et al., 2019] suggests that the distance of different matched features should be as identical as possible, which is defined as the second order similarity (SOS). By applying this constraint, the clusters for different matched features should have a similar clustering radius within the descriptor space. SOSNet is used to regularise the standard margin based triplet loss. An improvement on Tian et al. [2017]; Mishchuk et al. [2017] is also reported in Tian et al. [2019].

A recent work called GeoDesc [Luo et al., 2018] concentrates on generating more realistic yet challenging matched pairs for L2-Net. The angle between two intersecting rays pointing at the same 3D point from different views (matching features) and the incident angle difference between each ray (angle between local normal and the ray) are used to model the difficulty of matching homologous features. The authors discard easier pairs and only use more difficult matching pairs for sampling. This method achieves better performance in matching and retrieval benchmarks.

A literature summary graph for the presented research regarding feature description is shown in Figure 3.2.

## 3.4. An Application: Orientation of Oblique Aerial Images

In this section, the application of feature based image matching is reviewed with focus on the orientation of imagery taken from oblique aerial camera systems. For nadir or near nadir images, classical feature based image matching algorithms such as SIFT and SURF work well. The focus on image orientation for oblique imagery stems from the fact that this is a more challenging task due to extensive changes in viewing direction and viewpoint between the different images, exceeding the invariance threshold of classical feature based image matching algorithm as reported in Verykokou and Ioannidis [2018, 2016]; Jacobsen and Gerke [2016].

Several different methods for matching oblique images have been published. In Smith et al. [2008], conjugate points between the oblique and vertical images are collected interactively. The reason, as stated by the authors, is that the differences in illumination and viewing directions led to the failure of automatic tie point generation. An attempt to automatically match oblique and nadir images reported in Verykokou and Ioannidis [2016] used SURF; most of the resulting conjugate points lie on a planar surface within a limited degree of viewpoint change only. The method is bound to fail when the imaged area contains larger elevation differences. The idea of running multiple homographies is used in Onyango et al. [2017] to obtain tie points between UAV and oblique camera images. In this strategy, a first homography is computed and outliers of the result are then iteratively used to compute further homographies.

Kim et al. [2019] propose a deep neural network to estimate the rotation and affine transformation between a pair of images. A qualitative performance using nadir and oblique images from the ISPRS multi-platform photogrammetry dataset given in Nex et al. [2015] shows that the method can roughly align those images from different view points. However, the estimated transformation angles are discretized into large steps, e.g., 45 degrees, which restricts its practical usage. Moreover, from the theoretical standpoint, the method can yield correct results only for scenes containing small changes in height.

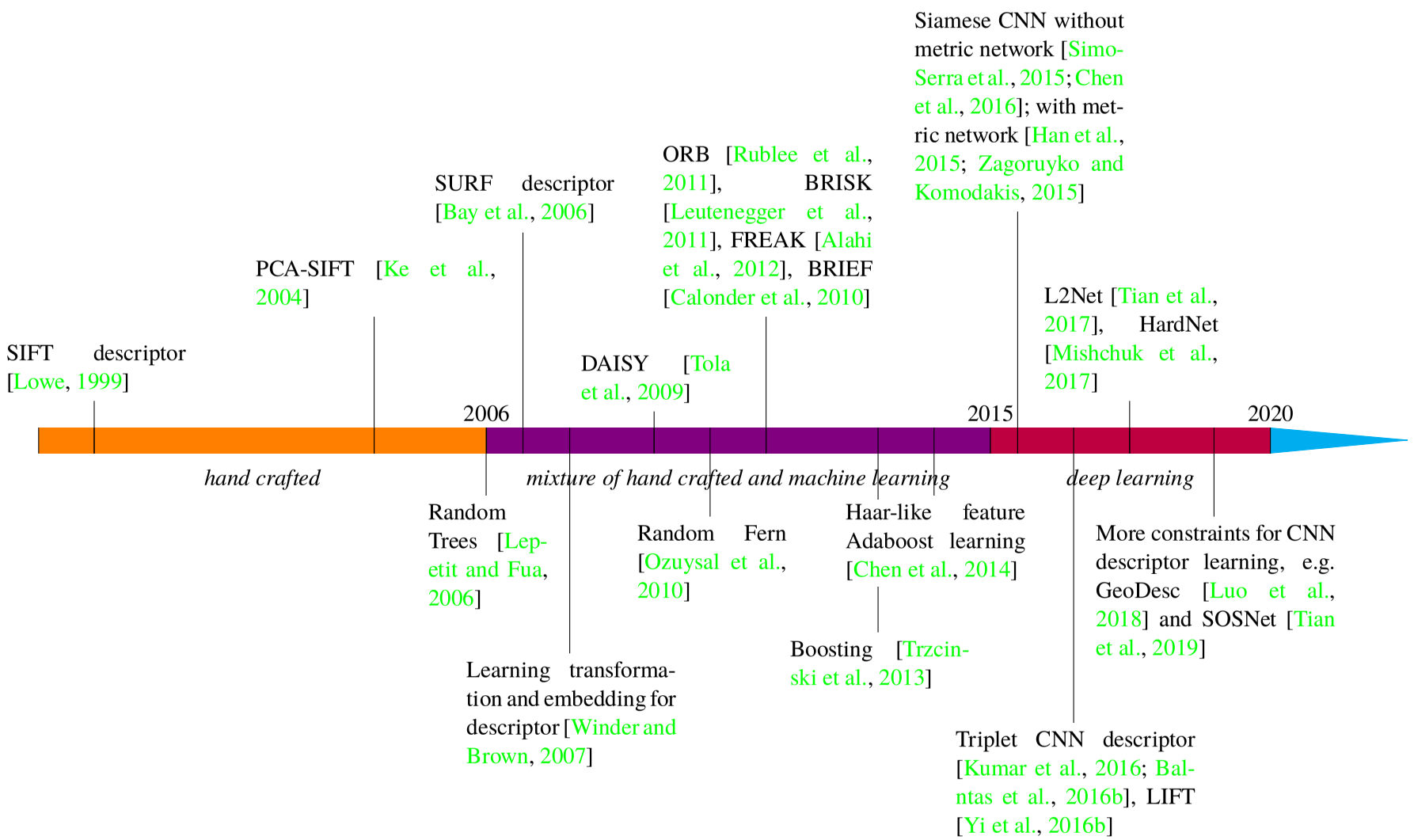



Figure 3.2.: The literature timeline for feature description. For better viewing of the whole graph, some closely linked papers are combined into one block with an +1 year difference, e.g. all the binary descriptors including ORB, BRISK, BRIEF and FREAK are combined into one block.

Matching of nadir to oblique camera images can also rely on view sphere simulation of images using ASIFT, as seen in Wang et al. [2018]. The idea of view sphere simulation is to simulate a lot of images with different levels of affine transformation between two images which need to be matched. Feature detection, orientation and description are then run based on all of the transformed images. Not surprisingly, for some pairs the geometric distortions are decreased to a level which can be handled by classical matching algorithms. Once those pairs are successfully matched, their conjugate points are transformed back to the original images, thus matching relationships are obtained.

Finally, a recently published approach successfully matches images taken from different cameras in an oblique penta camera system through optimizing the affine shape estimation, orientation assignment and feature description directly by using a deep neural network [Chen et al., 2020a].

Unfortunately, only limited information is reported for the matching algorithms used in commercial software, e.g., Pix4D or Photoscan. For open source software of 3D image reconstruction from multiple images, such as Bundler[2] [Snavely et al., 2006], VisualSFM[3] [Wu et al., 2011; Wu, 2013] and COLMAP[4] [Schönberger and Frahm, 2016], SIFT is the main algorithm used for feature based image matching.

## 3.5. Discussion

Before presenting this thesis' main contribution in the next chapter, this section summarizes some open questions in the related work that arise when working with descriptor, affine shape estimation and orientation assignment learning. Targeted solutions developed in this thesis will be provided in the next chapter in order to tackle the identified research issues.

### 3.5.1. Orientation Assignment and Affine Shape Estimation

When assigning orientation to a feature support window for image matching, only the relative orientation between the two images is relevant, as they can be matched in any absolute orientation. The same is true for the two parameters of affine distortion (different scale on the two axes of the image coordinate system and skew). Therefore, there is no single correct solution for orientation assignment and affine shape estimation for a single patch . In Yi et al. [2016a] and Yi et al. [2016b], it is claimed that orientation parameters optimized only by using descriptor distance loss contribute to finding the best possible solution for image matching. However, an underlying assumption that ensures this idea works is that there must be a distinctive descriptor distance minimum whenever two feature support windows are aligned. This means there is a unique solution for the problem. Of course, as far as the computation of image coordinates of tie points for image orientation

[2] http://www.cs.cornell.edu/ snavely/bundler/
[3] http://ccwu.me/vsfm/
[4] https://demuc.de/colmap/

is concerned, any of the solutions provided by the network is as good as any other. Nonetheless, the numeric stability and convergence properties of the computations should be investigated.

In current work on orientation and affine shape estimation, descriptors are needed to build a descriptor distance based loss. However, in classical feature based image matching, e.g., SIFT [Lowe, 1999, 2004], orientation assignment and affine shape estimation [Baumberg, 2000; Mikolajczyk and Schmid, 2004] depend only on geometric measures calculated on individual patches surrounding features. Thus, patches are transformed in some canonical form in which matching can be performed unambiguously and independent of descriptor distance. An open question is whether this idea can be transferred to deep learning based approaches as well. This issue will be explored and answered in sections 4.3 and 4.4.

### 3.5.2. Descriptor Learning

To learn descriptors from data, matched and unmatched pairs or triplets are fed into the learning framework. The loss function is designed to make the similarity, often measured by the inverse Euclidean distance between descriptors, a maximum for matched pairs and a minimum for unmatched ones. An identical number of positive and negative pairs are typically used in the training procedure to update the parameters of the descriptor models. However, in real applications, e.g., image matching or image retrieval, far more negative than positive pairs must be compared, as the process of finding correspondences by comparison uses a "one against many others" approach. Taking this imbalance into consideration, the unmatched pairs are mined by comparing and selecting difficult ones from a pool containing large numbers of unmatched pairs. Therefore, the descriptor sees a significantly larger number of negative samples (unmatched pairs) than positive samples (matched pairs) during training. Typical works concerning the search for more unmatched pairs are Mishchuk et al. [2017]; Simo-Serra et al. [2015]; the descriptors reported in these papers show notable improvements for discriminability through seeking more unmatched pairs during training.

On the other hand, the appearance of matched patches has not been explored in-depth in descriptor learning. Compared to "seeing" unmatched pairs, the descriptor has a limited chance to explore the intra-variation of matched pairs. In other words, for each patch the descriptor has a much lower chance of seeing the possible patches which match the patch but contain a limited level of distortion. In current research, only a couple of matched patches are contained for each feature and during training the matched pairs are sampled from those matched patches. Naturally, those patches can only cover a small part of the space formed by possible patches where a descriptor should be built close to the one for reference. To improve the coverage of possible matched features, a weak match network is proposed in section 4.2 in order to actively find the matched patch which is most distant from a reference patch during descriptor learning. By inserting this module in descriptor learning, the invariance of the descriptor against changes in viewpoint and viewing directions is improved.

### 3.5.3. An Aerial Photogrammetric Benchmark

To evaluate the different variants of descriptors, results from 3D reconstruction of images are used, e.g., Fan et al. [2017, 2019], and Jin et al. [2020]. The test dataset used in Fan et al. [2017, 2019] is a dataset composed of images containing consecutive change of viewpoint and viewing directions, i.e., although some overlapping images are indeed wide baseline pairs, for each image an overlapping partner can be found with only small changes in viewing direction and viewpoint position. In Jin et al. [2020], the Photo Tourism dataset is used as a wide baseline dataset for the 3D image reconstruction task. A check revealed that this dataset also contains a large number of consecutive views.

In aerial photogrammetry, matching of oblique images has been an attractive research area in the last years, partly due to its practical needs. While the images taken by different cameras belonging to an oblique camera system contain distinct and large viewpoint and viewing direction changes, consecutive images as seen in the aforementioned datasets typically do not exist. In order to evaluate the learned modules in an application involving more challenging changes in viewpoint and viewing direction, image blocks taken from oblique aerial camera systems should be used to evaluate the deep learning methods. In this thesis, image blocks taken from oblique aerial cameras are used for the performance evaluation of the methods proposed in this thesis, as illustrated in Sections 5.2 and 5.6.

### 3.5.4. Ability to Transfer Learned Modules

It has not been properly investigated yet how the learned modules for descriptors, orientation assignment and affine shape estimation might be transferred between different imaging domains. Thus, unlike for other machine learning tasks such as semantic segmentation and person re-identification, it is unclear how well the deep learning features for image matching can be transferred from the trained dataset, e.g., a close-range scene, to a test scenario, e.g., a set of aerial images.

Feature based image matching related deep learning tasks, especially affine shape estimation, orientation assignment and descriptor learning, are based on simple networks and a simple form of distance based loss functions which should be beneficial to the generalisation of learned models. In addition, all the context windows involved in those tasks are “local” and several times the size of the detected characteristic scale of features. Consequently, a limited range of context is involved, which also increases the possibility of wide generalisation. In order to obtain a better understanding of this question, the capability of deep learning based feature matching algorithms of transferring the results across imaging domains should be systematically analysed, e.g., based on a series of datasets containing significant differences such as street view images and aerial images. The experiment presented in this thesis uses distinctively different datasets for training and evaluation tasks, thereby gaining a better understanding of the ability to transfer deep learning based feature matching algorithms across different domains.

# 4. Deep Learning Feature Representation

This thesis aims at improving the performance of feature based image matching by solving the problem of feature affine shape estimation, feature orientation and feature description in deep learning frameworks. After features are detected with positions and scales in an image, a context window is extracted surrounding each detected feature. The problem of affine shape estimation, as shown in section 2.1.4, is solved by a self-supervised affine shape estimation network proposed in this thesis. In the following step, the orientation of local features is derived via an orientation network trained by a self-supervised orientation estimation network proposed in this thesis. Alternatively, the orientation and affine shape of local feature are simultaneously estimated by a full affine shape estimation network, also proposed in this thesis. To compute descriptors for detected features, principal orientation and optional affine shape are used to extract feature support windows. In those windows, the influence of rotation and affine transformations caused by changes in viewpoint and viewing direction is reduced. Using the windows as input, the features are described by applying the feature descriptor network, which, in turn, is trained using the new descriptor training framework proposed in this thesis. Through combining those trained networks, features are detected and described for any input image. This is called the inference of the learned modules, i.e., how to apply them for real applications where feature matching is required. In the next step, features detected and described in different images are matched in descriptor space.

Before diving into the details of different networks in this thesis, an overview of the proposed methodology is given in section 4.1. Sections 4.2 through 4.5 offer a closer look at the innovation created by the models original to this thesis, with particular regard to learning feature description, orientation assignment, affine shape estimation and full affine shape estimation. In section 4.6, the inference of trained modules is explained before, finally, the assumptions underlying the experiments and theoretical limitations are discussed in section 4.7.

## 4.1. Overview of the Methodology

Overall, the framework proposed in this thesis contains two parts: training and inference. In the training part, four networks, for affine shape estimation, orientation assignment, full affine shape estimation and feature description respectively, are trained. To train the descriptors, the matching relationship between local feature patches is required, whereas for the other three modules this relationship is not needed. In the inference part, the trained modules are used in a feature detection and description framework to detect features and compute

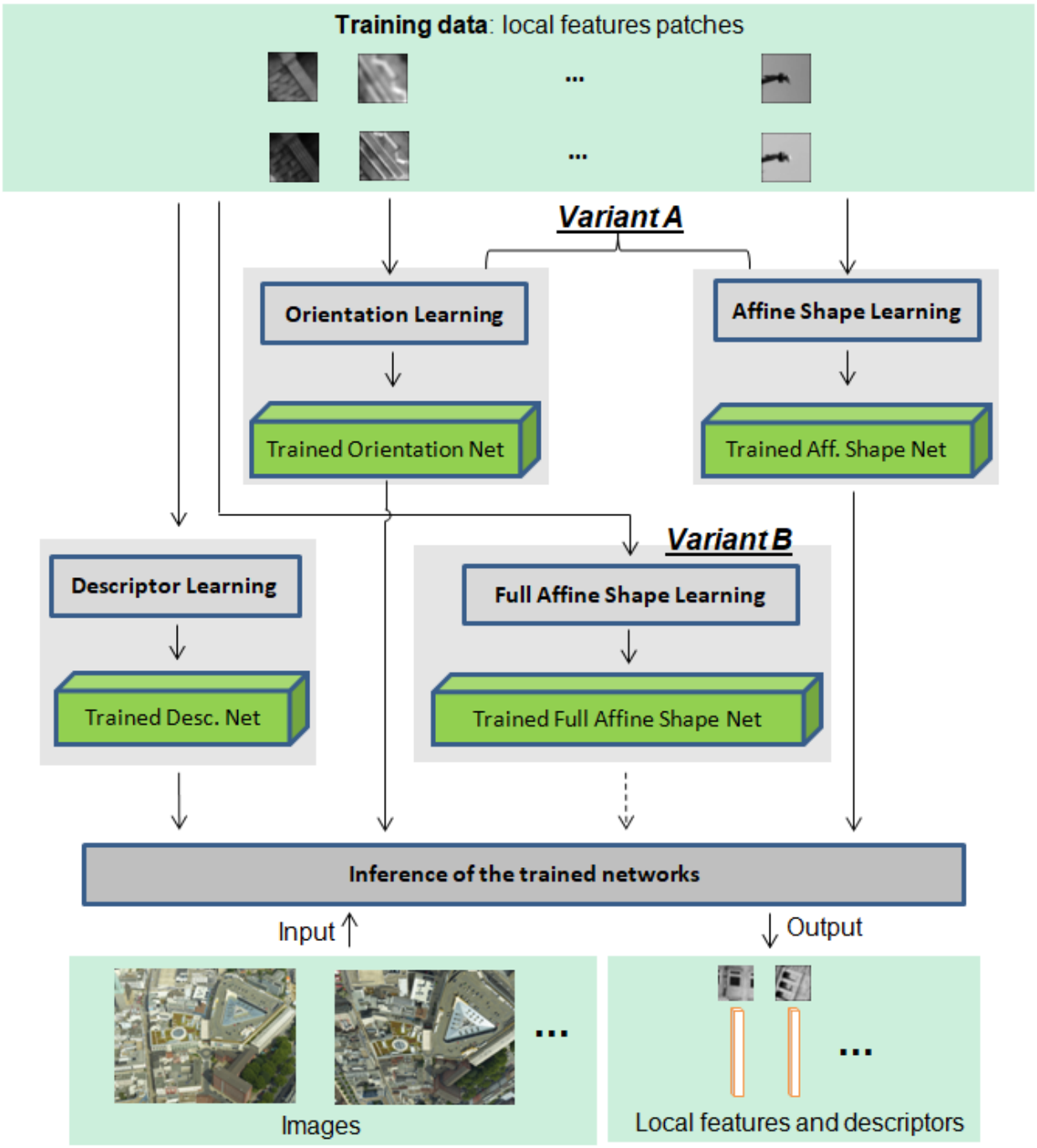



Figure 4.1.: Overview of the methodology. The training data containing local feature patches is used in descriptor, orientation, affine shape and full affine shape learning modules. The orientation and affine shape of local feature can be estimated separately (variant A) or simultaneously via the full affine shape learning (variant B). The trained descriptor, orientation and affine shape network (or Full affine shape network) are combined in the inference pipeline of the method, the input and output of which are images and corresponding detected features and descriptors.

descriptors for input images. An overview of the method used in this thesis is illustrated in figure 4.1.

The functions of each training module are as follows:

- Affine shape estimation network: given an image patch, it predicts the affine shape of the underlying patch. Only one degree of rotation remains once patches have been corrected with predicted affine shape.
- Orientation assignment network: given a patch, it predicts the principal direction of said patch, which allows image patches of arbitrary rotation to be aligned to the predicted principal rotation.
- Full affine shape network: given a patch, it predicts the principal direction and the affine shape of said patch simultaneously, which allows image patches of arbitrary affine shape and rotation to be corrected with predicted full affine shape.
- Descriptor network: given a feature support window of a feature as input, it outputs a descriptor which is the final representation of the feature.

As mentioned before, the inference pipeline combines the trained affine shape, orientation, full affine shape, and description networks. Through running the inference pipeline the local features and corresponding descriptors for input images are extracted as shown in the bottom of figure 4.1. In the chapter describing the experiment itself, the inference is applied to image blocks in order to obtain features and descriptors. In this way, image orientation and bundle adjustment results are obtained. These will be used as the main test for the performance of the proposed method.

## 4.2. Descriptor Learning using Active Weak Match Finder - WeMNet

Descriptors are computed based on feature support windows. The core idea of descriptor learning is to pull descriptors of matched features to be as close as possible while pushing descriptors of unmatched features to be far away from each other in the feature descriptor space. Also, feature descriptors should be robust to slight geometric transformations in the feature support window.

In this section, the descriptor learning architecture is first introduced. Then the generation of training pairs, augmentation rules and the loss function used in training are explained. This is followed by the introduction of an optional weak match branch which actively finds hard matched samples that can be used in the descriptor learning as well. Finally, the analysis of descriptor distance for matched feature pairs undergoing different types of geometric transformations is proposed.

### 4.2.1. Descriptor Learning Architecture

To train descriptors, matched and unmatched patch pairs are used in a Siamese CNN which is composed of two branches, $P_1$ and $P_2$. The architecture is illustrated in figure 4.2. The patches are support windows of features. After applying a descriptor network to those patches, the related descriptors $d_1$ and $d_2$ are obtained. As the pair of patches is either matched or unmatched, the corresponding descriptor distances are used to build the loss function using the matching relationship of the pair of patches.

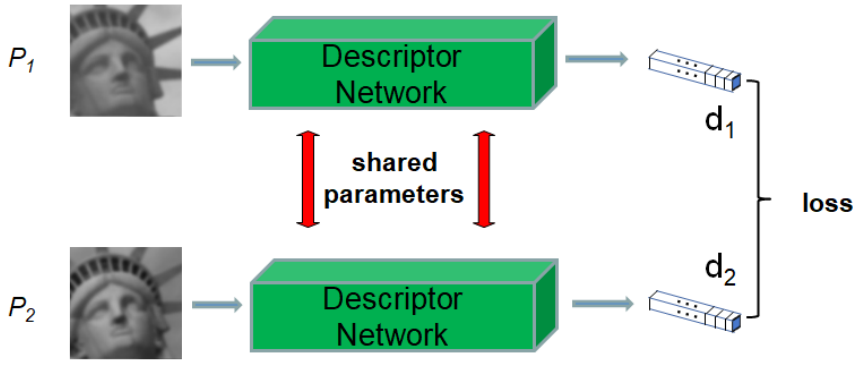



Figure 4.2.: Descriptor Learning Architecture. The Siamese CNN is composed of two branches, $P_1$ and $P_2$. Each branch takes one patch (feature support window) as input and runs it through the descriptor network to obtain the descriptors $d_1$ and $d_2$. This process is repeated for each input patch. The descriptor networks in the two branches share the same weights. The input pairs of patches are matched or unmatched pairs of features. Loss based on the calculated descriptors is then constructed.

**Descriptor Network:** Details on the descriptor network used to generate descriptors are provided in table 4.1 and illustrated in figure 4.3. This network is identical to the one used in Mishchuk et al. [2017] and Mishkin et al. [2018], and was originally proposed by Tian et al. [2017]. Through a series of convolution layers, a 32×32 pixel single channel image patch is transformed and compressed into a 128-dimensional descriptor which is then scaled to unit length. This rescaling, in turn, results in the fact that the maximum Euclidean distance between any two descriptors is 2.

### 4.2.2. Generation of Training Pairs

Following Mishchuk et al. [2017], the training data is composed of image patches. These patches are derived from image blocks with known interior and exterior orientation, depicting a 3D scene in which a dense description surface is given as well. For each patch it is already known which other patches are correct matches. This is realised via a 3D point index for each patch, thus patch tuples are represented as 3D point indices. Working with this data, a set of N 3D points is first sampled without replacement from the training data. Then, two different patches associated with the same 3D point index are randomly selected to form a

| Layer | Filter | #In-Out | Stride | Size of feature map | Activation | BN |
|---|---|---|---|---|---|---|
| 1 | 3x3 | 1-32 | 1 | $32 \times 32 \times 32$ | ReLU | Yes |
| 2 | 3x3 | 32-32 | 1 | $32 \times 32 \times 32$ | ReLU | Yes |
| 3 | 3x3 | 32-64 | 2 | $16 \times 16 \times 64$ | ReLU | Yes |
| 4 | 3x3 | 64-64 | 1 | $16 \times 16 \times 64$ | ReLU | Yes |
| 5 | 3x3 | 64-128 | 2 | $8 \times 8 \times 128$ | ReLU | Yes |
| 6 | 3x3 | 128-128 | 1 | $8 \times 8 \times 128$ | ReLU | Yes |
| | | | Dropout with rate=0.1 | | | |
| 7 | 8x8 | 128-128 | 1 | $1 \times 1 \times 128$ | ReLU | Yes |

Table 4.1.: Descriptor Network architecture. #In-Out: number of input and output channels. ReLU: Rectified Linear Unit, which works as $g(z) = max(0, z)$ for an input $z$. BN: batch normalization.

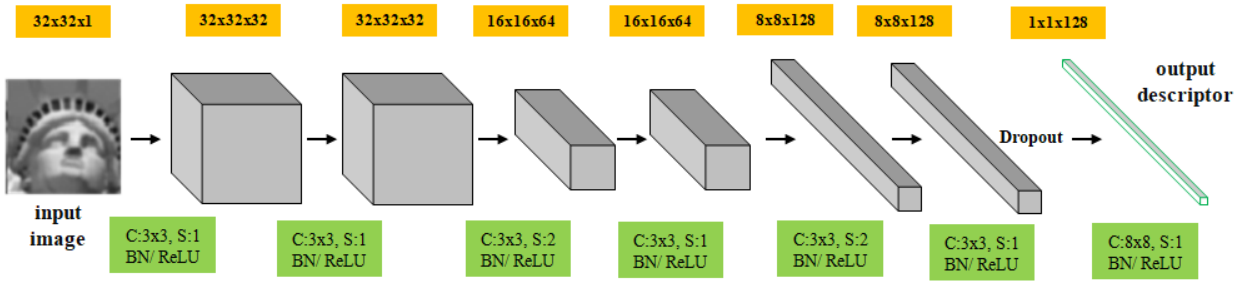



Figure 4.3.: Architecture of the Descriptor CNN. An input $32 \times 32$ pixel feature support window is convolved and processed according to the operation indicated in the green boxes, in which C means the size of convolution kernel, S stands for stride, BN means batch normalization and ReLU indicates the rectified linear unit. The feature map size in each layer is indicated in the orange boxes on the top of each layer. Finally, a 128-dimensional descriptor is obtained.

pair of patches. Then, a mini batch (with size N) of training data is generated. This process continues to generate pairs until the number of samples reaches the required size for training.

The training process also needs counter examples, i.e., non-matching pairs. For a patch $p_1$, patches associated with a different 3D point index belong to the unmatched patches (see Figure 4.4). The use of the 3D index thus ensures that these pairs, when sampled randomly from the training data, do not by chance contain correct matches. Obviously, the number of possible unmatched pairs is much higher than that of matched pairs, due to the fact that any combination of pairs that have a different 3D index within a mini-batch can be used as a unmatched pair. To properly train the network, however, an equal number of matched and unmatched pairs is needed. Therefore, unmatched pairs used in training have to be selected from the larger set. For this selection step, the hardest mining strategy [Mishchuk et al., 2017] is employed.

#### Hardest Mining of Negative Pairs

The hardest unmatched pair is the unmatched pair with the smallest distance between two corresponding descriptors. It is selected ("mined") during training. For the $i^{th}$ patch in the branch $P_1$ ($p_1^i$), its distance to

the hardest unmatched patch is defined as $A_i^{hardest}$, which is calculated as follows:

$$A_i^{hardest} = min(min_{j \neq i} A(d_1^i, d_1^j), min_{j \neq i} A(d_1^i, d_2^j)) \tag{4.1}$$

where $d_1^i$ = the $i^{th}$ descriptor of the branch $P_1$
$d_2^j$ = the $j^{th}$ descriptor of the branch $P_2$
$A(:,:)$ = Euclidean distance of two descriptors
$i, j \in [1, 2, ..., N]$

$min_{j \neq i} A(d_1^i, d_1^j)$ and $min_{j \neq i} A(d_1^i, d_2^j)$ compute the hardest unmatched pairs with the smallest distance within the same branch $P_1$, and between the two branches $P_1$ and $P_2$, respectively. The selection of the hardest unmatched pair is illustrated in figure 4.4. Through seeking the hardest samples, the network "sees" far more unmatched training pairs than matched ones, which corresponds to the fact that matching means "searching for a needle in a haystack," due to the fact that a lot more unmatched pairs than matched pairs need to be compared for real image matching applications. After mining, a triplet containing a pair of matched patches as well as the most difficult unmatched patch is obtained for each training patch within a mini-batch passed to the first branch of the network.

**Augmentation of Patches:** In order to increase the number of matched pairs and make the network to see more samples during training, the matched pairs are augmented by flipping or rotating them with a value randomly chosen from the set $[90°, 180°, 270°]$. The square feature support window rotated by those three angles does not introduce zero grey value in the border. In addition, a random brightness change of 10% is applied, which means the pixel grey values in the original patch are scaled by a value randomly selected in the range of $[0.9, 1, 1]$. All the aforementioned simulations are applied online.

### 4.2.3. Loss Function

To calculate the loss, the triplet margin-based loss function [Chechik et al., 2010; Hoffer and Ailon, 2015] is used.

**Triplet Margin based Loss:** In a batch of N sampled triplets, the loss is defined as:

$$L_{triplet} = \sum_{i=1}^{N} max(0, A_i(d_1^i, d_2^i) + \beta - A_i^{hardest}) \tag{4.2}$$

where $A_i(d_1^i, d_2^i)$ is the distance between the $ith$ matched pairs and $A_i^{hardest}$ is the hardest negative distance computed for the $ith$ triplet inside the batch. $\beta$ is a margin formed between the distance of matched and unmatched pair. In this thesis, $\beta$ is set to be 1, as suggested in Mishchuk et al. [2017]. This loss concentrates

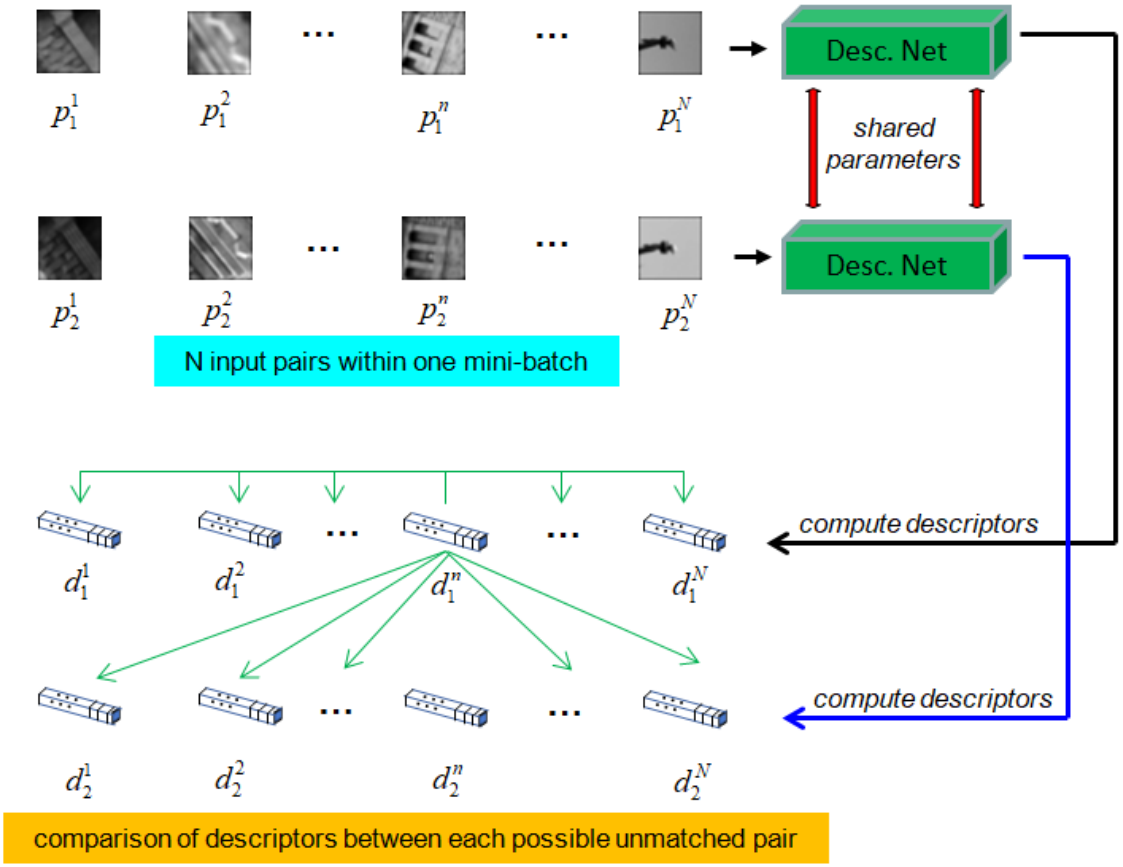



Figure 4.4.: Hardest mini batch mining. The two patch sets $\{p_1^1, p_1^2 ... p_1^n, ..., p_1^N\}$ and $\{p_2^1, p_2^2 ... p_2^n, ..., p_2^N\}$, as indicated in the top of the figure, are fed into the two branches Siamese descriptor network to compute descriptors $\{d_1^1, d_1^2 ... d_1^n, ..., d_1^N\}$ and $\{d_2^1, d_2^2 ... d_2^n, ..., d_2^N\}$, both of which are indicated in the bottom part of the figure, separately. Take patch $p_1^n$ as example, its descriptor, $d_1^n$, is compared with all the descriptor of patches within branch $P_1$, between branch $P_1$ and $P_2$. Each green arrow indicates a comparison. The pair with the smallest descriptor distance is selected ("mined") for calculating the loss for unmatched pairs.

on forming a margin and does not care how large the distance of matched pair is. Considering that all the descriptors are normalized, the maximum distance between any two descriptors is 2.

### 4.2.4. Weak Match Branch

During the training process, the patches of matched features are aligned. However, it is useful to involve possible matched patches containing a limited range of geometric transformations, so that the descriptor network can see enough intra-class variance, i.e., various possible appearances of matched patches caused by geometric transformations. The idea of involving weak matches is illustrated in figure 4.5. The feature patch used for descriptor training in a normal configuration is called a reference patch here. In descriptor space, patches containing a certain level of geometric transformation when compared to a reference patch -

while still being close to the reference patch - are defined as "weak matches" (see the patch $d^i_{weak}$ in figure 4.5). When care is not taken for those patches, they may be located far from the reference patch in descriptor space. Through finding weak matches and feeding them into the descriptor network, the descriptor distance between the descriptor of a reference patch and its weak match is pulled to be closer, therefore the trained descriptor is then made more invariant against a certain range of geometric transformation.

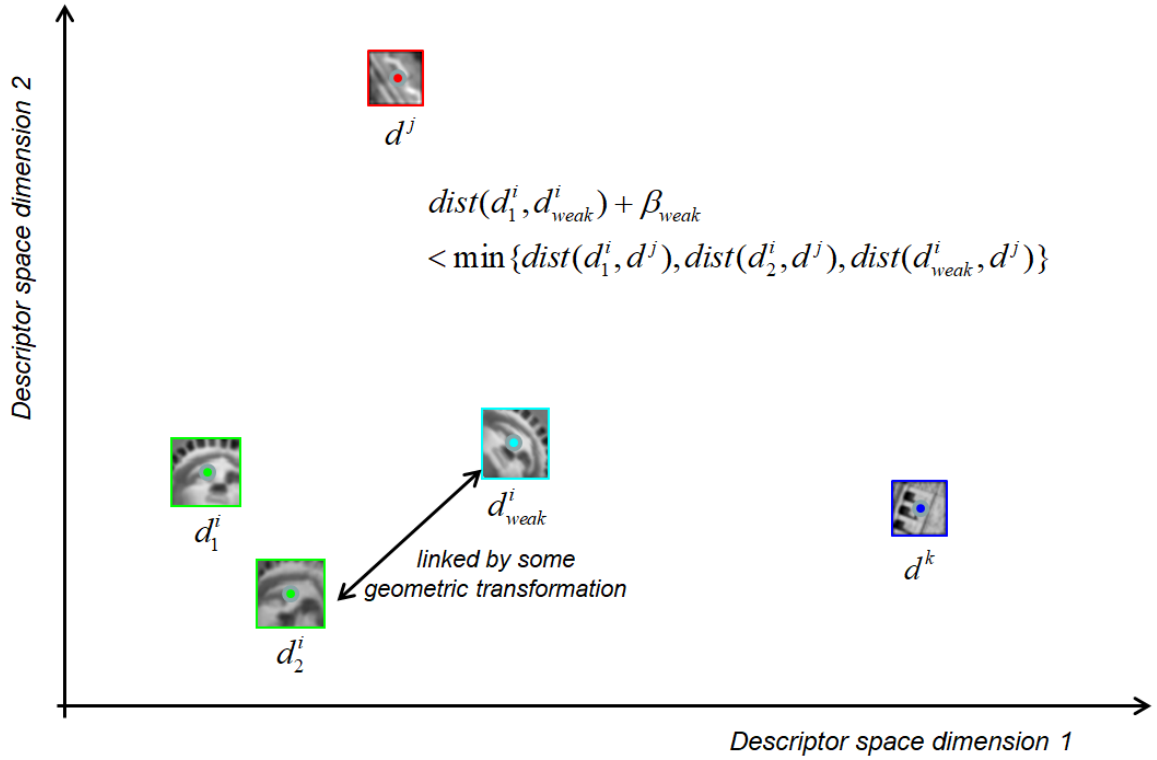



Figure 4.5.: The idea of weak matches. Here the descriptor space is simplified as a two dimensional space. In descriptor space, the descriptors $d^i_1$ and $d^i_2$ represent a pair of matched features, the feature support window of which is shown as two image patches with green borders. The other two features represented by $d^j$ and $d^k$ are unmatched features to $d^i_1$ and $d^i_2$. Through geometric transformation of the feature support window of $d^i_2$, another patch - the descriptor of which is $d^i_{weak}$ - is found with the condition that the distance between $d^i_2$ and $d^i_{weak}$ is maximized. Although the feature represented by $d^i_{weak}$ is a weak match, it should still be closer to $d^i_1$ when compared to any unmatched features. Therefore, the border of the descriptor space dominated by $d^i_1$ is $d^i_{weak}$, and a threshold $\beta_{weak}$ is proposed to form a margin between the feature space of $d^i_1$ and the space of any other unmatched features. This means that the minimum distance between $d_j$ and each one of $d^i_1$, $d^i_2$ and $d^i_{weak}$ is made to be larger than the sum of $dist(d^i_1, d^i_{weak})$ and the threshold margin $\beta_{weak}$.

This work proposes an additional branch to actively find weak matches. To illustrate the weak matched patch finding process, refer to figure 4.6. The two patches $p^i_1, p^i_2$ form a pair of matched feature patches. After fed into the network, their descriptors $d^i_1, d^i_2$ are obtained and used to compute the descriptor loss, as explained in equation 4.2. This corresponds to the normal configuration of descriptor learning. In the weak match branch, patches $p^i_{1\bullet crop}$ and $p^i_{2\bullet crop}$ are cropped from the centring part of $p^i_1, p^i_2$. Then $p^i_{2\bullet crop}$ is fed into a weak

match network to predict geometric transformation parameters, which are used to sample $p^i_{weak}$ from $p^i_2$. The geometric transformation parameters are generated through the weak match net, which is explained in the next paragraph. The patch $p^i_{weak}$ is the most distant patch from $p^i_{1\bullet crop}$ in terms of descriptor distance, found by maximizing the descriptor distance between $p^i_{1\bullet crop}$ and $p^i_{weak}$, namely $loss_{weak_match_finder}$. For this maximum distance optimizing problem, only the network parameters in the weak match network are trainable. In the next step, the descriptors of $p^i_{weak}$ and $p^i_{1\bullet crop}$ are used to build $loss_{weak_match}$, in which weakly matched patches are made to lie close to its reference patch in descriptor space.

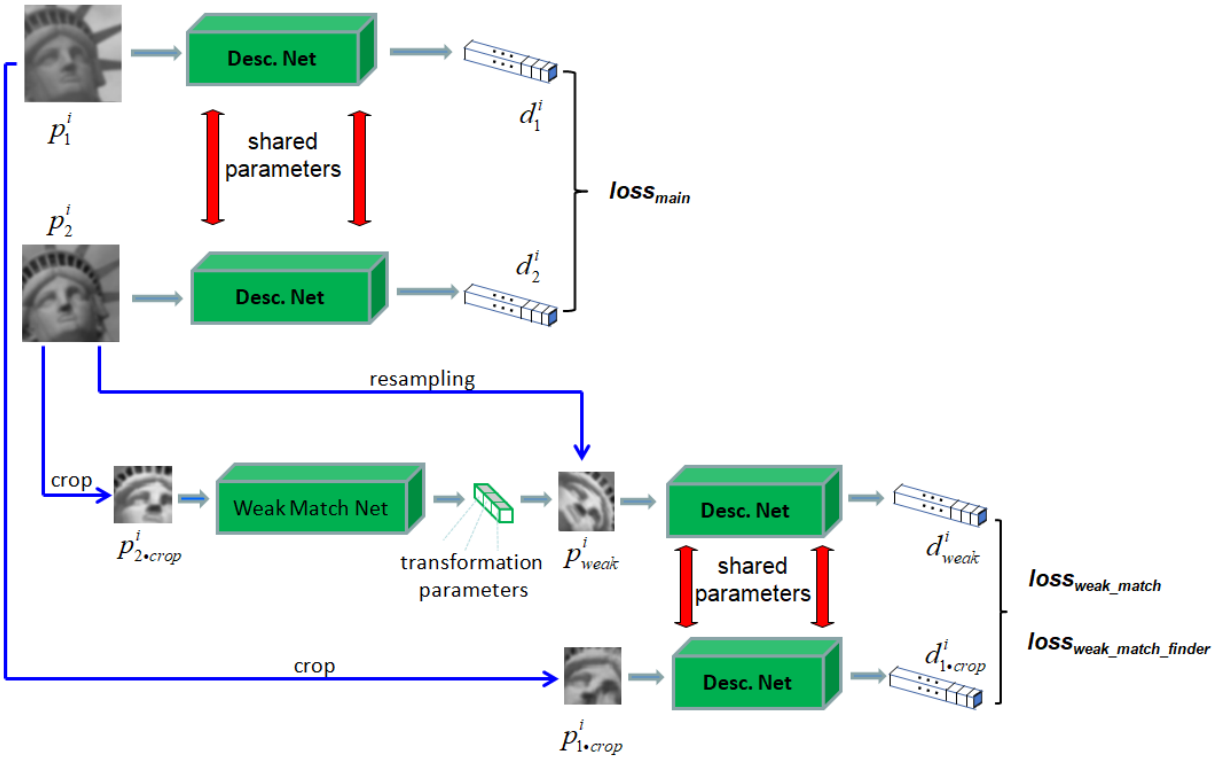



Figure 4.6.: Process of finding weakly matched patches and applying them into descriptor learning.

**Weak Match Network (WeMNet):** For each anchor patch, a small network is built to find the most difficult match and then this patch is used to train the model. This network is designed as illustrated in figure 4.7, and the details are also shown in table 4.2. The output of the weak match net are affine geometric transformation parameters. For the meaning of different parameters, refer to formula 4.7. The three parameters predicted from weak match network are only given in a relatively small range (this is why the arctan values in Table 4.2 are dived by the empirically found denominators to compute the angles of affine transformation $\psi, \theta, \phi$). Thus the found weak match patch is guaranteed to be only a transformed version in a certain range. As a consequence, the main purpose of this weak match network is to actively increase the invariance of descriptors against a limited range of affine transformations.

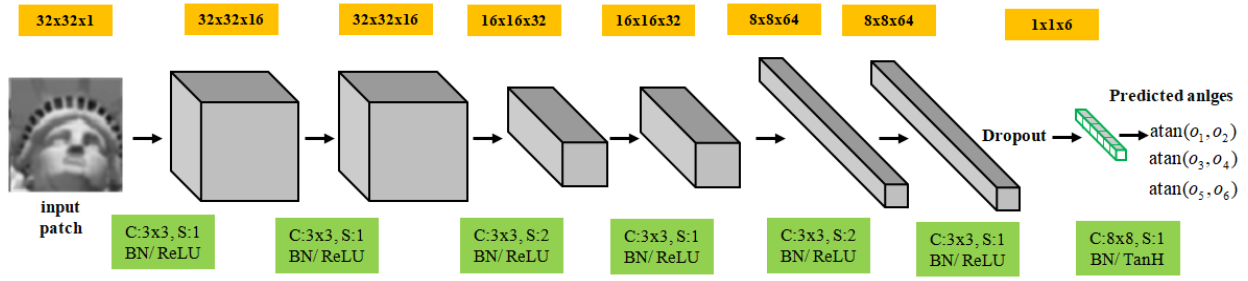



Figure 4.7.: Weak match network. The input cropped feature support window and the output is a set of predicted affine transformation parameters, which is composed of three angles (translation and scale are not considered).

| Layer | Filter | #In-Out | Stride | Size of feature map | Activation | BN |
|---|---|---|---|---|---|---|
| 1 | 3x3 | 1-16 | 1 | $32 \times 32 \times 16$ | ReLU | Yes |
| 2 | 3x3 | 16-16 | 1 | $32 \times 32 \times 16$ | ReLU | Yes |
| 3 | 3x3 | 16-32 | 2 | $16 \times 16 \times 32$ | ReLU | Yes |
| 4 | 3x3 | 32-32 | 1 | $16 \times 16 \times 32$ | ReLU | Yes |
| 5 | 3x3 | 32-64 | 2 | $8 \times 8 \times 64$ | ReLU | Yes |
| 6 | 3x3 | 64-64 | 1 | $8 \times 8 \times 64$ | ReLU | Yes |
| | | | Dropout with rate=0.25 | | | |
| 7 | 8x8 | 64-6 | 1 | $1 \times 1 \times 6$ | TanH | Yes |
| 8 | $\psi = atan(O_1, O_2)/6, \theta = atan(O_3, O_4)/8, \phi = atan(O_5, O_6)/8$ | | | | | |

Table 4.2.: Weak match network architecture. #In-Out: number of input and output channels. ReLU: Rectified Linear Unit, which works as $g(z) = max(0, z)$ for an input $z$. BN: batch normalization. TanH: hyperbolic tangent defined as $TanH(x) = (e^x - e^{-x})/(e^x + e^{-x})$.

**Loss:** There are two different losses related to the usage of weak match networks. The first one is $L_{weak_match_finder}$, which accounts for finding the weak match; the parameters used to optimize it are the training parameters of the weak match network. The second one is $L_{weak_match}$, which aims to form a margin between the weakly matched patch and the unmatched patch to help separate the patches, which contain only a limited amount of geometric transformation, within descriptor space.

The first loss is defined as

$$L_{weak_match_finder} = \sum_{i=1}^{N} (2 - A_i(d^i_{1\bullet crop}, d^i_{weak})) \tag{4.3}$$

where $d^i_{1\bullet crop}$ = the descriptor of the feature patch cropped from the $i^{th}$ patch in the first branch

$d^i_{weak}$ = the descriptor of the weak match for the $i^{th}$ patch in the first branch; this descriptor is resampled from $p^j_2$, using a transformation predicted by the weak match net

$A(:,:)$ = Euclidean distance between two descriptors

$i, j \in [1, 2, ..., N]$ where $N$ stands for mini-batch size

The central idea of this loss is to maximize the distance between $d_1^i$ and $d_{weak}^i$, thereby finding the most difficult weak match for $d_1^i$. Once this weak match has been found, it is used in the second loss $L_{weak_match}$, which is defined as:

$$L_{weak_match} = \sum_{i=1}^{N} max(0, A_i(d_{1\bullet crop}^i, d_{weak}^i) + \beta_{weak} - A_i^{hardest'}) \tag{4.4}$$

where $\beta_{weak}$ = the threshold for the margin between a weak match and the most difficult unmatched patch in descriptor space, which is smaller than $\beta$

$A_i^{hardest'}$ = the distance between the descriptor of the most difficult unmatched pair and the $i^{th}$ input feature of the Siamese network.

$A_i^{hardest'}$ is defined as:

$$A_i^{hardest'} = min(min_{j\neq i}A(d_{1\bullet crop}^i, d_{2\bullet crop}^j), min_{j\neq i}A(d_{1\bullet crop}^i, d_{weak}^j), min_{j\neq i}A(d_{2\bullet crop}^i, d_{weak}^j)) \tag{4.5}$$

Similar to the hardest unmatched pair mining introduced before, the most difficult unmatched patch inside a mini-batch is found with the three cases to form unmatched feature pairs:

- the cropped patches from the first and second branch - $min_{j\neq i}A(d_{1\bullet crop}^i, d_{2\bullet crop}^j)$.
- the cropped patch from the first branch and the weak matches - $min_{j\neq i}A(d_{1\bullet crop}^i, d_{weak}^j)$.
- the cropped patch from the second branch and the weak matches - $min_{j\neq i}A(d_{2\bullet crop}^i, d_{weak}^j)$.

Among the three different combinations of unmatched pairs, the $L_{weak_match}$ aims to find the most difficult one and put this found feature into the descriptor training process. Consequently, a more concentrated distribution of descriptors for features containing intra-class variance can be achieved. For descriptor training, the loss is now extended:

$$L_{descriptor} = L_{triplet} + \lambda_{wm} L_{weak_match} \tag{4.6}$$

Here $\lambda_{wm}$ controls the relative importance of the weak match loss.

**Training Strategy:** In each training iteration, the training works in the following order:

- *Finding the weakest match*: The weak match network is trained with $L_{weak_match_finder}$, then the weak match patch is found and sampled. During this step, the descriptor network is fixed.
- *Descriptor training*: The weak match network is fixed and the descriptor network is trained with $L_{descriptor}$.

Through consecutive training, $L_{weak_match_finder}$ will gradually increase as the descriptor attains more invariance against geometric transformations modelled by the weak match network, until it reaches an approximately constant value. Correspondingly, $L_{descriptor}$ will gradually decrease and then stay approximately constant once the networks are sufficiently trained.

## 4.3. Self Supervised Feature Affine Shape Learning - MoNet

In this section, the distortion parameters for affine shape are discussed. In central projection, a circle on a plane parallel to the image plane is imaged as a circle. When the image is taken from an oblique direction, however, its image turns to an ellipse. In that case, the image scale varies in different directions. The oblique direction is defined by two parameters, $\phi$ and $\theta$ (see figure 4.8). In addition to that, the in-plane rotation $\psi$ of the feature is still unknown.

This part first presents the affine transformation parametrization in section 4.3.1 and then proposes a self-supervised affine shape estimation network to estimate the affine shape for each input feature context window in section 4.3.2.

### 4.3.1. Affine Transformation Decomposition

The linear part of an affine transformation in 2D is a $2 \times 2$ matrix. It can be decomposed into the following form, as illustrated in figure 4.8, according to [Morel and Yu, 2009]:

$$\begin{aligned} B &= R(\psi)DR(\phi) = \lambda R(\psi)D'R(\phi) \\ &= \lambda \begin{bmatrix} cos\psi & -sin\psi \\ sin\psi & cos\psi \end{bmatrix} \begin{bmatrix} 1/\sqrt{cos\theta} & 0 \\ 0 & \sqrt{cos\theta} \end{bmatrix} \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix} \\ &= \lambda \begin{bmatrix} cos\psi & -sin\psi \\ sin\psi & cos\psi \end{bmatrix} \begin{bmatrix} \sqrt{t} & 0 \\ 0 & 1/\sqrt{t} \end{bmatrix} \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix} \end{aligned} \tag{4.7}$$

where $\lambda$ = detected feature scale
$R(\cdot)$ = a rotation matrix whose parameter is the rotation angle
$\theta$ = angle in the range of $[0, \pi/2)$, which controls the amount of unequal scale
$t$ = stretch, equal to $1/cos(\theta)$, indicating the level of anisotropic scale change
$\phi$ = longitude, which controls the direction in which two orthogonal directions are scaled according to stretch
$\psi$ = in plane rotation angle

The first line of the above equation stands for a SVD decomposition and then the overall scale ($\lambda$) in the diagonal matrix $D$ is separated out. $\lambda$ is determined during feature detection and is subsequently kept

constant. $\psi$ is estimated during the orientation assignment process. Thus, the two degrees of freedom of affine shape transformation are $\theta$ and $\phi$ (or, alternatively $t$ and $\phi$). Note that while $R(\psi)$ and $R(\phi)$ represent rotation matrices, $D'$ does not. However, the determinant of $D'$ equals 1.

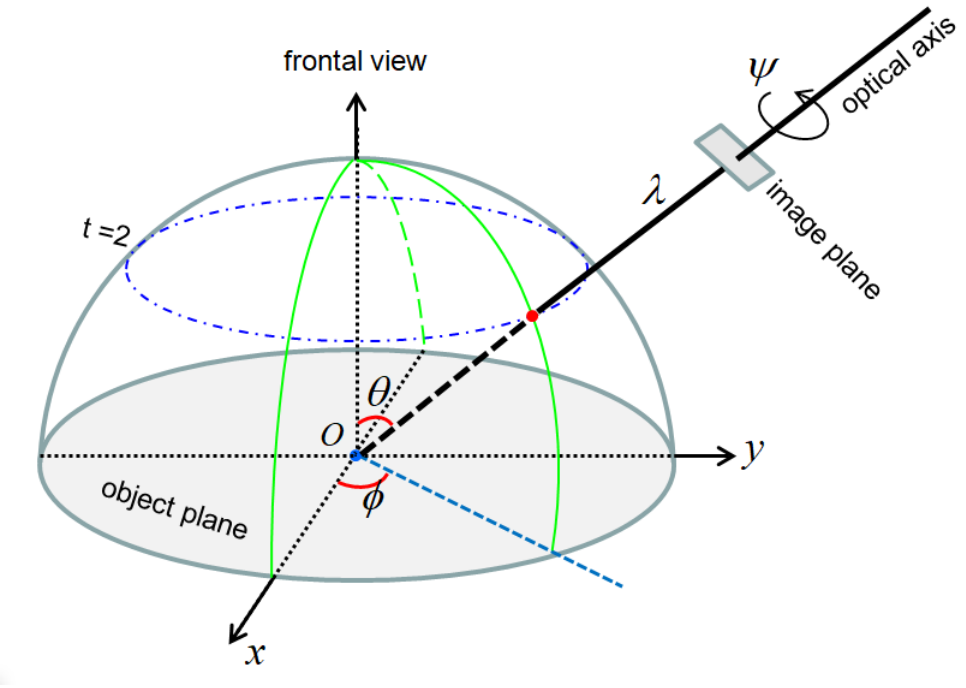
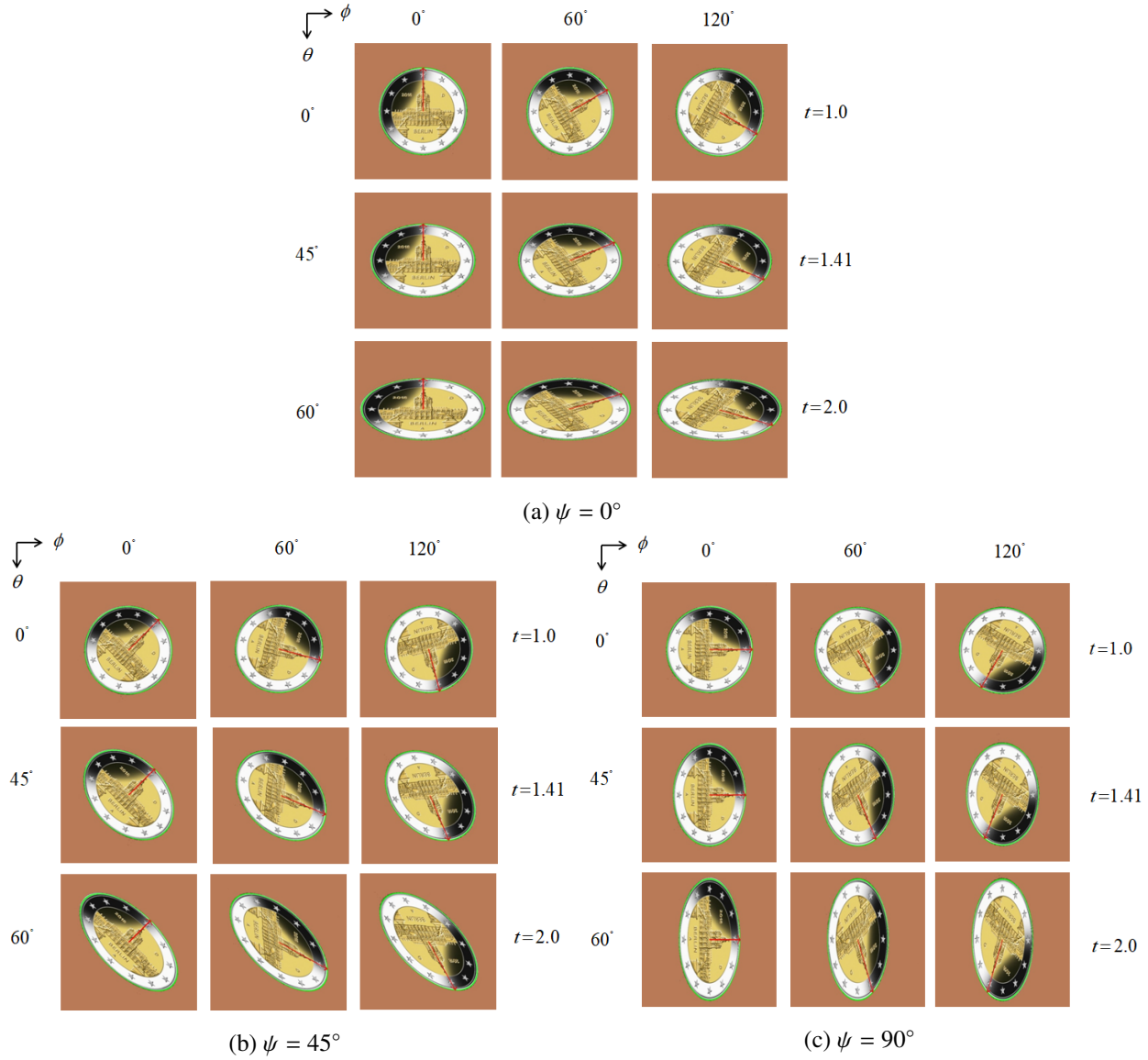



Figure 4.8.: Affine transformation decomposition employed in Morel and Yu [2009]. $\phi$ is defined as the longitude. The anisotropic scale change is applied in the direction of $\phi$ and the one orthogonal to $\phi$. $\theta$ is called "latitude" by the authors of Morel and Yu [2009] (in astronomy and geodesy, this angle is called zenith angle), which controls the amount of anisotropic scale change. $\phi$ and $\theta$ determine the viewing direction of the camera. The camera's in-plane rotation is defined as $\psi$ and the scale factor of imaging is controlled by $\lambda$.

To create an affine transformation, a canonical feature is first rotated by $\phi$ and then elongated by applying a scale change of $\sqrt{t}$ and $1/\sqrt{t}$ in the direction of $\phi$ and the one orthogonal to $\phi$, respectively. The pixels contained in the neighbourhood of the feature forms the content of that feature. This step is followed by a rotation of $\psi$ to indicate the orientation of the feature content. $\phi$ and $\theta$ control the viewing direction, $\theta$ controls the anisotropic scaling. In SIFT, $\theta$ is set to zero, thus the determination of $\psi$ and $\phi$ degenerates to the determination of one angle only, and $\lambda$ is estimated as the characteristic scale of detected features. To cope with images which show a higher level of affine transformations, $\theta$ and $\phi$ should both be estimated to compensate the affine transformations in images. An example of affine distortion in real image patches is shown in figure 4.9. As shown there, only $\theta$ and $\phi$ affect the content of transformed features. Applying a different $\psi$ only changes the orientation of the image content contained in the feature.




(a) $\psi = 0°$

(b) $\psi = 45°$

(c) $\psi = 90°$

Figure 4.9.: Example of transformed angles in affine decomposition with different $\psi$, $\theta$, $\phi$. An image of a Euro coin is used as the test image in this figure. To better visualize the result after applying different transformations, the boarder of the coin is circled in green and the upright direction is indicated with a red arrow. (a), (b) and (c) present the case for $\psi = 0°$, $\psi = 45°$ and $\psi = 90°$, respectively.

As employed by Mishkin et al. [2018] and also by Perd'och et al. [2009], an affine transformation, as applied

to each patch individually, is decomposed into the following form:

$$\begin{aligned} B &= \lambda R(\psi')A \\ &= \lambda \begin{bmatrix} cos\psi' & -sin\psi' \\ sin\psi' & cos\psi' \end{bmatrix} \begin{bmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{bmatrix} \\ &= \lambda \begin{bmatrix} cos\psi' & -sin\psi' \\ sin\psi' & cos\psi' \end{bmatrix} \begin{bmatrix} a_{11}^{'} + 1 & 0 \\ a_{21}^{'} & a_{22}^{'} + 1 \end{bmatrix} \end{aligned} \tag{4.8}$$

where $A$ = a matrix of affine shape parameters with $det(A) = 1$
$\lambda$ = the detected feature scale, kept constant during estimation of affine shape parameters
$a_{11}, a_{21}, a_{22}$ = affine shape parameters. Since $det(A) = 1$, $a_{11} = 1/a_{22}$
$a_{11}^{'}, a_{21}^{'}, a_{22}^{'}$ = the residual form of affine shape parameters, computed during affine shape estimation
$\psi'$ = feature rotation angle, also kept constant during estimation of affine shape parameters

Setting $a_{12} = 0$ enables the affine shape estimation to preserve the vertical direction for a local image patch because the affine shape matrix then always has one eigenvector equal to $(0, 1)^T$. As a consequence, the rotation matrix $R(\psi)$ of the equation 4.7 must e split into two consecutive rotation matrices $R(\psi')$ and $R(\psi^{''})$ such that $R(\psi) = R(\psi')R(\psi^{''})$. The role of $R(\psi^{''})$ is to rotate $A$ so that the vertical direction of the feature is preserved, i.e. $a_{12} = 0$. This relationship is explained by equation 4.9.

$$\begin{aligned} B &= \lambda \begin{bmatrix} cos\psi & -sin\psi \\ sin\psi & cos\psi \end{bmatrix} \begin{bmatrix} 1/\sqrt{cos\theta} & 0 \\ 0 & \sqrt{cos\theta} \end{bmatrix} \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix} \\ &= \lambda \begin{bmatrix} cos\psi' & -sin\psi' \\ sin\psi' & cos\psi' \end{bmatrix} \begin{bmatrix} cos\psi^{''} & -sin\psi^{''} \\ sin\psi^{''} & cos\psi^{''} \end{bmatrix} \begin{bmatrix} 1/\sqrt{cos\theta} & 0 \\ 0 & \sqrt{cos\theta} \end{bmatrix} \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix} \\ &= \lambda \begin{bmatrix} cos\psi' & -sin\psi' \\ sin\psi' & cos\psi' \end{bmatrix} \begin{bmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{bmatrix} \qquad = \lambda R(\psi')A \end{aligned} \tag{4.9}$$

The rotation matrix with the angle $R(\psi')$ will be discussed in the orientation assignment part. The matrix A contains the affine shape parameters. The affine shape estimation network (the same network as used in Mishkin et al. [2018]) is used to estimate the affine matrix elements for each input patch. This network has a similar structure as the descriptor network, outputting affine shape parameters $a_{11}^{'}, a_{21}^{'}, a_{22}^{'}$. Note that since $det(A) = 1$, A has two degrees of freedom, which is in accordance with equation 4.7.

### 4.3.2. Self Supervised Affine Shape Estimation Module

Instead of using a Siamese architecture where the matching relationship is needed to sample matched and unmatched pairs for training, the second moment matrix is used here to measure the shape of local features

during the training of affine shape. Stretch and skew of a local patch, derived from the second moment matrix, are used as the loss for affine shape training.

An overview of affine estimation is shown in figure 4.10. For the input image patches, different affine shapes are first simulated. Then these patches are fed into the affine shape estimation network to estimate their affine shape parameters. In the following step, the estimated affine shape parameters are used to re-sample input affine simulated patches and obtain an output patch, and the stretch and skew of the output patches are calculated based on the second moment matrix. The whole network is then trained to minimize the loss derived from the predicted stretch and skew. This is an extension of classical affine shape estimation theory as presented in Baumberg [2000]; Mikolajczyk and Schmid [2004], in which the affine shape is estimated and the root of the second moment is used to resample the patch after each iteration (see section 3.2.2). However, in this work, the measurement of stretch and skew is preserved, and the iteration process is replaced by the steps of training the affine shape estimation network with the proposed loss.

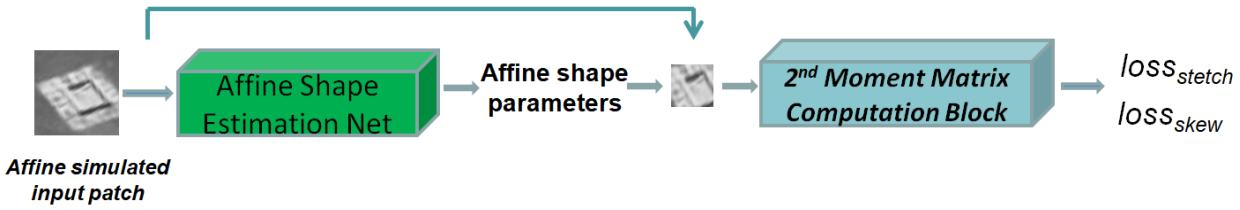



Figure 4.10.: Overview of the affine shape estimation network. The simulated patch is first fed into the affine shape estimation network, which predicts the affine shape parameters. Afterwards, the patch is resamplled using the predicted affine shape parameters. Then, the second moment computation block takes the resampled patch and computes stretch and skew, which are used to form the training loss.

**Second Moment Matrix Computation Block:** For the resampled patch, the elements of the second moment matrix are calculated by the $2^{nd}$ moment computation block, as shown in figure 4.10. The process itself is illustrated in figure 4.11. For the input patch with a size of $w \times h$, the gradients in x and y direction, $g_x$ and $g_y$, are calculated. Then, the squared terms of the gradients $g_x^2, g_y^2$ and $g_x g_y$ are obtained, which, in turn, are weighted by a Gaussian function with standard deviation $w/2$. The weighted values are accumulated by deriving the mean values of the weighted $g_x^2, g_y^2$ and $g_x g_y$. The second moment matrix is then computed as:

$$M = \begin{bmatrix} g^2_{x_mean} & g_{x_mean} g_{y_mean} \\ g_{x_mean} g_{y_mean} & g^2_{y_mean} \end{bmatrix} \tag{4.10}$$

Through eigenvalue decomposition of $M$, the stretch and skew in the local image patch are derived. All layers of the second moment computation block are differentiable so that the partial derivatives of the loss w.r.t the different elements of the estimated second moment matrix can be back-propagated to the affine shape estimation network.

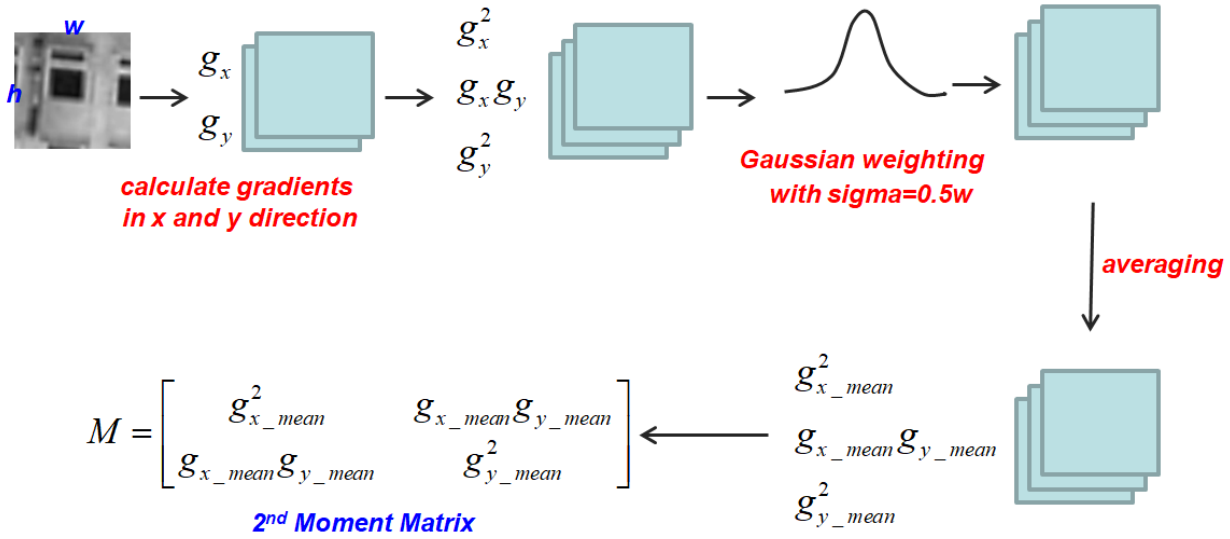



Figure 4.11.: The second moment matrix computation process. Taking an image patch of size $w \times h$ pixels as input, the final output are the elements of the second moment matrix $M$ of the input patch.

**Affine Shape Estimation Network:** A structure very similar to the descriptor network is utilized for the affine shape estimation network. It differs from the descriptor network in only two aspects: A) the number of input and output units is halved from the layer Aff-1 to Aff-6; B) the output layer is composed of 3-element units activated by a hyperbolic tangent function. The three output units correspond to the affine shape parameters $a_{11}^{'}, a_{21}^{'}, a_{22}^{'}$. To fix the overall scale of a feature, the three affine shape parameters are subsequently divided by its determinant $(a_{11}^{'}+1)(a_{22}^{'}+1)$. The affine estimation network is used to estimate the affine matrix for each input affine simulated patch, the details of this network are explained in table 4.3. The affine network is illustrated in figure 4.12. The output of ReLU (rectified linear unit) is larger or equal to zero, while the output of TanH (the hyperbolic tangent function) lies in the range of (-1,1). This range copes well with the range of affine shape parameters. Consequently, for the final layer (Aff-7), TanH (the hyperbolic tangent function) is used for non-linear activation.

| Layer | Filter | #In-Out | Stride | Activation | BN |
|---|---|---|---|---|---|
| Aff-1 | 3x3 | 1-16 | 1 | ReLU | Yes |
| Aff-2 | 3x3 | 16-16 | 1 | ReLU | Yes |
| Aff-3 | 3x3 | 16-32 | 2 | ReLU | Yes |
| Aff-4 | 3x3 | 32-32 | 1 | ReLU | Yes |
| Aff-5 | 3x3 | 32-64 | 2 | ReLU | Yes |
| Aff-6 | 3x3 | 64-64 | 1 | ReLU | Yes |
| | | Dropout with rate=0.1 | | | |
| Aff-7 | 8x8 | 64-3 | 1 | TanH | No |

Table 4.3.: Structure of the affine shape estimation network. #In-Out: number of input and output channels. TanH stands for the hyperbolic tangent function.

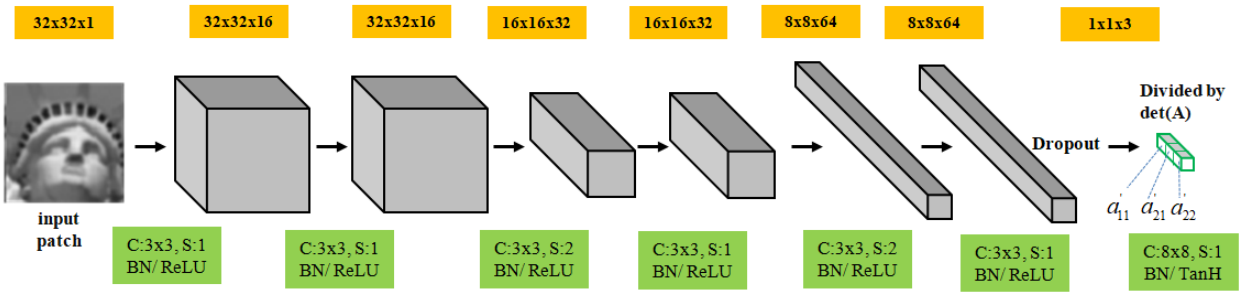



Figure 4.12.: Affine matrix estimation network. The final output are the elements of the affine matrix for each input patch. An input $32 \times 32$ pixel patch around a detected features is convolved and processed according to the operation indicated in the green boxes, in which C means Convolution, S stands for Stride, BN means Batch Normalization, ReLu stands for rectified linear unit and TanH stands for hyperbolic tangent function. The feature map size in each layer is shown in the orange boxes on top of each layer. Finally, the affine shape parameters $a'_{11}, a'_{21}, a'_{22}$ are obtained and divided by $(a'_{11}+1)(a'_{22}+1)$.

**Data Augmentation:** Affine augmentation is applied for each input patch. Both the longitude $\phi$ and rotation angle $\psi$ are randomly sampled (generated from a uniform distribution) in range of $[0, 2\pi)$, while the stretch $(t)$ is gradually increased from the first to later epochs. During sampling, only the centre part of a patch transformed in the affine simulation is cropped to avoid "black" pixels, which have a detrimental effect on the generalization ability of the network to be trained.

**Loss Function:** To measure the stretch and skew of the resampled patch using the estimated affine transformation, the result of the eigenvalue decomposition of the second moment matrix $M$ is used. Before deriving the eigenvalue of $M$, normalization by dividing through the root of the determinant of $M$ is carried out:

$$\begin{aligned} M_{norm} &= \begin{bmatrix} a/\sqrt{|M|} & b/\sqrt{|M|} \\ b/\sqrt{|M|} & c/\sqrt{|M|} \end{bmatrix} \\ &= R \begin{bmatrix} eig_{max} & 0 \\ 0 & eig_{min} \end{bmatrix} R^T \end{aligned} \tag{4.11}$$

where $eig_{min}$ and $eig_{max}$ are the smaller and larger eigenvalue, and $R$ is a $2 \times 2$ rotation matrix. The absolute value of the ratio between the two eigenvalues measures the stretch of the local affine shape. Accordingly, the stretch loss is constructed as:

$$L_{stretch} = 1 - \left| \frac{eig_{min}}{eig_{max}} \right| \tag{4.12}$$

which drives the magnitude of the two eigenvalues to be close to each other. Therefore, a shape containing no anisotropic scale change is preferred during training.

Also, $R$ measures the skew of the local affine shape, namely the direction in which the two orthogonal unequal

scale changes happen. During training, $R$ is gradually changed towards a $2 \times 2$ identity matrix. In turn, this makes $M_{norm}$ to have a diagonal form and thus the skew loss is defined as:

$$L_{skew} = \left| \frac{b}{\sqrt{|M|}} \right| \tag{4.13}$$

A lower skew loss $L_{skew}$ is obtained when $\frac{b}{\sqrt{|M|}}$ is close to zero. This causes $M_{norm}$ to become close to a diagonal matrix. In addition, when the two eignevalues are identical to each other, $L_{stretch}$ is minimized. This, in turn, brings $M_{norm}$ to an identity matrix, which is the underlying shape of the canonical feature.

For the affine shape, $L_{stretch}$ and $L_{skew}$ are combined to form the training loss and a relative importance factor $\lambda_{skew}$ is set for $L_{skew}$:

$$L_{aff} = L_{stretch} + \lambda_{skew} * L_{skew} \tag{4.14}$$

In the experimental chapter of this thesis, different magnitudes of $\lambda_{skew}$ are tested in order to identify the magnitude most useful for feature matching. Due to the fact that the second moment matrix is used to measure the shape of the resampled patch using predicted affine shape parameters, the affine shape network proposed in this thesis is named as **MoNet** (2nd Moment Network).

In [Mishkin et al., 2018], a Siamese CNN is used to train the affine shape network using descriptor distance based loss. However, the involvement of descriptor distance based loss leads to the requirement of pre-known matching relationships among patches. Compared to descriptor distance based loss, the proposed loss term does not rely on any pre-known matching relationship among patches, thus it works in a self supervised way. Also, the difficulty of generating training data is decreased and, in theory, every feature detected with location and scale can be fed into this framework for training of affine shape.

## 4.4. Self Supervised Orientation Assignment Module - MGNet

As feature pairs can not only exhibit affine distortion but also can be rotated in an arbitrary way, there still exists the (unknown) rotation angle $\psi$ for each patch. This is called feature orientation and this section describes how to estimate it. Note that as rotations of windows containing features are generated by simulation, explicit matching relationships are not required, which explains the term “self supervised” module.

**Orientation Learning Architecture:** An orientation assignment network is set up for the estimation of orientation in a local feature patch. The overview of this method is illustrated in figure 4.13. The input feature patch is simulated with different rotations, and the orientation assignment network is used to estimate its orientation, which is used to resample the input simulated patch. Subsequently, the resampled patch is fed into mean gradient computation block to calculate its mean gradients. Using all the calculated mean

gradients, the orientation loss is then calculated. In the following the orientation network, gradient layers, the augmentation strategy as well as the training loss are presented in detail.

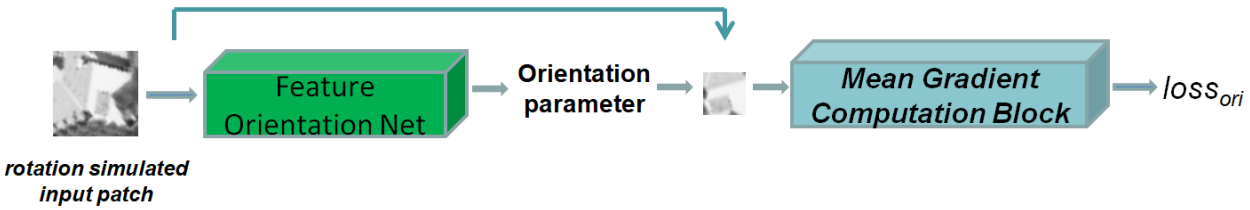



Figure 4.13.: An overview of the orientation estimation network architecture. After rotation simulation, the feature orientation of the simulated patch is estimated using the feature orientation network and then resampled with the estimated rotation angle. The resampled patch is subsequently used as input for the mean gradient computation block to calculate the orientation of the predicted feature and then the corresponding loss.

**Feature Orientation Network:** The orientation network contains blocks similar to the ones used in the affine shape estimation network. The only exception are the last layers, which use the two normalized output units to predict the rotation matrix. The details of the orientation network are explained in table 4.4 and figure 4.14. The final layer outputs a regressed rotation matrix for each input feature patch.

| Layer | Filter | #In-Out | Stride | Activation | BN |
|---|---|---|---|---|---|
| Ori-1 | 3x3 | 1-16 | 1 | ReLU | Yes |
| Ori-2 | 3x3 | 16-16 | 1 | ReLU | Yes |
| Ori-3 | 3x3 | 16-32 | 2 | ReLU | Yes |
| Ori-4 | 3x3 | 32-32 | 1 | ReLU | Yes |
| Ori-5 | 3x3 | 32-64 | 2 | ReLU | Yes |
| Ori-6 | 3x3 | 64-64 | 1 | ReLU | Yes |
| | | Dropout with rate=0.25 | | | |
| Ori-7 | 8x8 | 64-2 | 1 | Tanh | No |
| Ori-8 | rotation matrix using elements of (Ori-7) | | | | |

Table 4.4.: Structure of the orientation assignment network used in this thesis.

To calculate the output rotation matrix, 2D quaternions with the two elements $q_0$ and $q_1$ are used and normalized so that $q_0^2 + q_1^2 = 1$, producing a unit quaternion. The final rotation matrix is constructed as:

$$R = \begin{bmatrix} 1 - 2q_1^2 & -2q_0q_1 \\ 2q_0q_1 & 1 - 2q_1^2 \end{bmatrix} \tag{4.15}$$

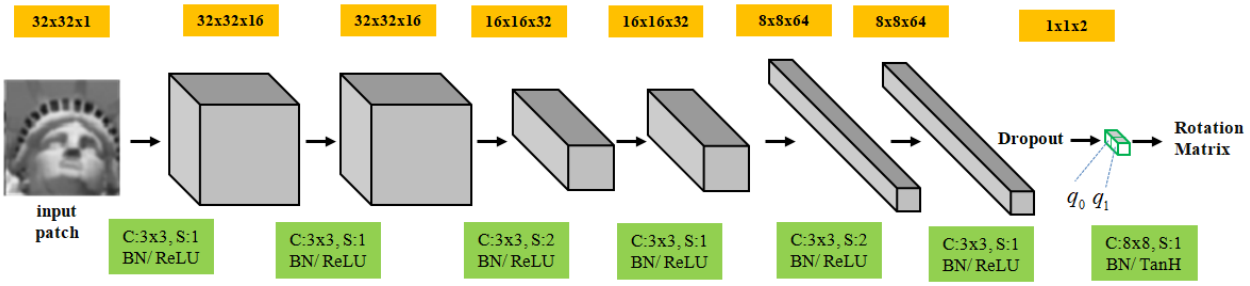



Figure 4.14.: Orientation estimation network. The final output is the predicted rotation matrix for each simulated input patch. An input $32 \times 32$ pixel patch around the detected feature is convolved and processed according to the operation indicated in the green boxes, in which C means Convolution, S stands for Stride, BN means Batch Normalization, ReLu stands for rectified linear unit and TanH stands for hyperbolic tangent function. The feature map size in each layer is indicated in the orange boxes on the top of each layer. Finally, the rotation matrix for the input feature is obtained.

**Mean Gradient Computation Block:** For each patch, similar to [Brown et al., 2005], the mean gradient is calculated and used to estimate the principal orientation of the resampled patch in figure 4.13. The calculation process of mean gradient is illustrated in figure 4.15. The gradients in x and y direction of a local feature patch are first calculated and then weighted by a Gaussian kernel with a standard deviation of half the input patch size. This weighting process assigns pixels closer to centre higher weights and, vice versa, pixels more distant to the centre a lower weights. In the next step, the mean values are computed using an average pooling operation and the results in two directions are normalized to form a unit vector, which is used as the input of arctangent function to derive the rotation angle ($\eta$) of the input patch. It is worth noting that the weights in those layers are predefined and kept constant.

As the orientation network proposed in this thesis relies on mean gradients of local patches, it is abbreviated to **MGNet** (Mean Gradient Network).

**Augmentation:** Uniformly distributed random online rotation augmentation in a range of $[0, 2\pi)$ is applied for each patch during training.

**Loss Function:** The training loss is built on the estimated principal orientation of the patch, which is resampeled using the predicted orientation from the orientation assignment network. The loss is defined as:

$$L_{ori} = |\eta| \tag{4.16}$$

where $\eta$ is the result computed by the mean gradient computation block. This loss drives the canonical feature to the one whose mean gradient in y direction is zero, which corresponds to the canonical feature, too.

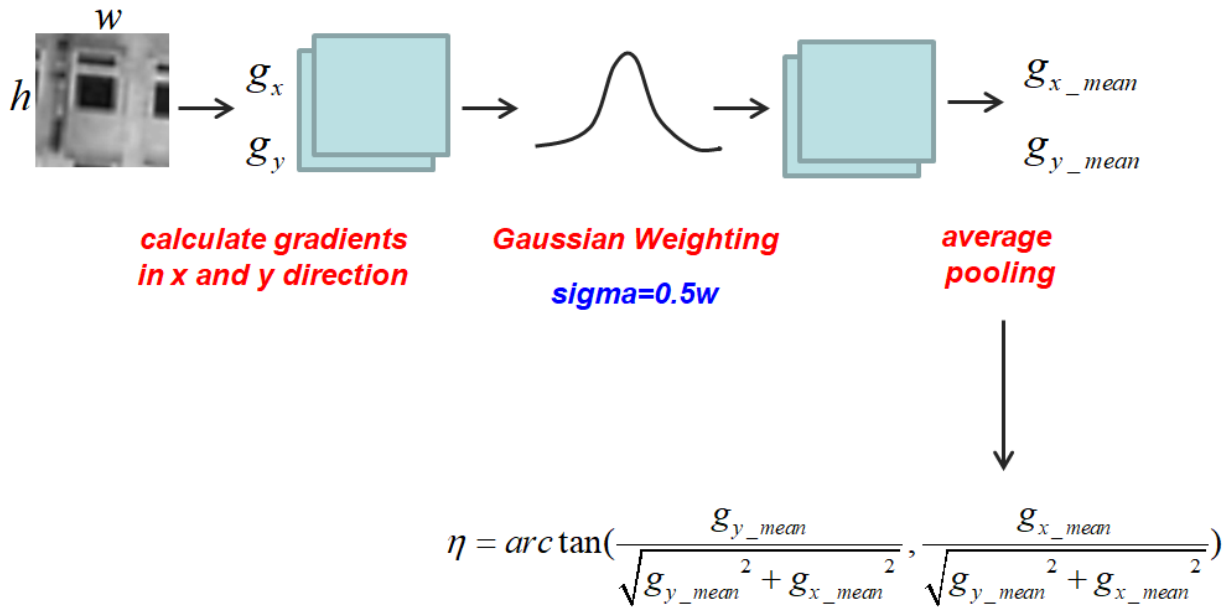



Figure 4.15.: The mean gradient computation process. For each feature patch of size $w \times h$ pixels, the final output is the predicted rotation angel of the input feature patch.

## 4.5. Full Affine Estimation Network - Full-AffNet

In the previous sections, the affine shape and orientation are trained separately. This section describes the simultaneous training of affine shape and orientation parameters, called full affine estimation, and proposes a new training loss for the corresponding network.

A full affine network (**Full-AffNet**) is used to regress the affine transformation of a local patch with three degrees of freedom, i.e., stretch, skew and rotation. As before, the input are simulated image patches transformed using different affine parameters. The full affine network is then used to predict the three parameters of the full affine shape. Afterwards, those predicted full affine shapes are used to resample the image patch. The sum of stretch loss, skew loss and rotation loss, which are proposed in the previous two sections, measures how close the current estimated patch is to the canonical form of the employed feature.

First, the overall architecture that is used to learn the stretch, skew and rotation is explained. In the next step, the training loss, as well as data augmentation are discussed.

### 4.5.1. Full Affine Network

The full affine network contains a network structure similar to the one used by the affine shape network, except for the output layer. The details of the network architecture are provided in table 4.5. After the computation,

the four output values are interpreted as (reshaped to) a 2×2 matrix and then normalized by the square of its determinant, eliminating the effect of the overall scale change. In other words, the degree of freedom for the output patch is three.

| Layer | Filter | #In-Out | Stride | Activation | BN |
|---|---|---|---|---|---|
| FullAffine-1 | 3x3 | 1-16 | 1 | ReLU | Yes |
| FullAffine-2 | 3x3 | 16-16 | 1 | ReLU | Yes |
| FullAffine-3 | 3x3 | 16-32 | 2 | ReLU | Yes |
| FullAffine-4 | 3x3 | 32-32 | 1 | ReLU | Yes |
| FullAffine-5 | 3x3 | 32-64 | 2 | ReLU | Yes |
| FullAffine-6 | 3x3 | 64-64 | 1 | ReLU | Yes |
| | | Dropout with rate=0.25 | | | |
| FullAffine-7 | 8x8 | 64-4 | 1 | No | No |

Table 4.5.: Structure of the full affine estimation network

### 4.5.2. Training Loss

As the full affine network treats the stretch, skew and rotation simultaneously, the shape of the output resampled patch is measured by the orientation and affine shape loss. In particular, the full affine shape loss is defined as:

$$L_{full-aff} = \lambda_{ori} * L_{ori} + L_{stretch} + \lambda_{skew} * L_{skew} \tag{4.17}$$

In the above formula, $\lambda_{ori}$ and $\lambda_{skew}$ are used to control the relative importance of $L_{ori}$ and $L_{skew}$, respectively. The process of computing the three different loss components is given in the previous two sections. Section 4.3.2 and 4.4 provide the details.

When optimizing $L_{full-aff}$, the desired shape for the output patches is expected to be isotropic with zero skew and zero mean gradient in y direction, which corresponds to the canonical patch.

### 4.5.3. Data Augmentation

Affine augmentation is conducted for the input pair. The rotation angle and skew are simulated within a range of $[0, 2\pi)$ with a uniform distribution sampler, whereas the stretch factor is gradually improved from the beginning to later stages of training.

## 4.6. Inference based on the Trained Networks

Once the four aforementioned modules are learned, they are integrated into a feature and descriptor extraction pipeline which outputs detected features and their descriptors for an input image. The whole process is shown

in figure 4.16 with regard to how the proposed networks in this thesis are used in real feature detection and description tasks. First, the Hessian matrix determinants of each pixel in the input image are calculated for each sample scale of the input image in scale-space. Then, the local extrema of the Hessian determinant are detected in scale-space, followed by the refinement of image coordinates and characteristic scale.

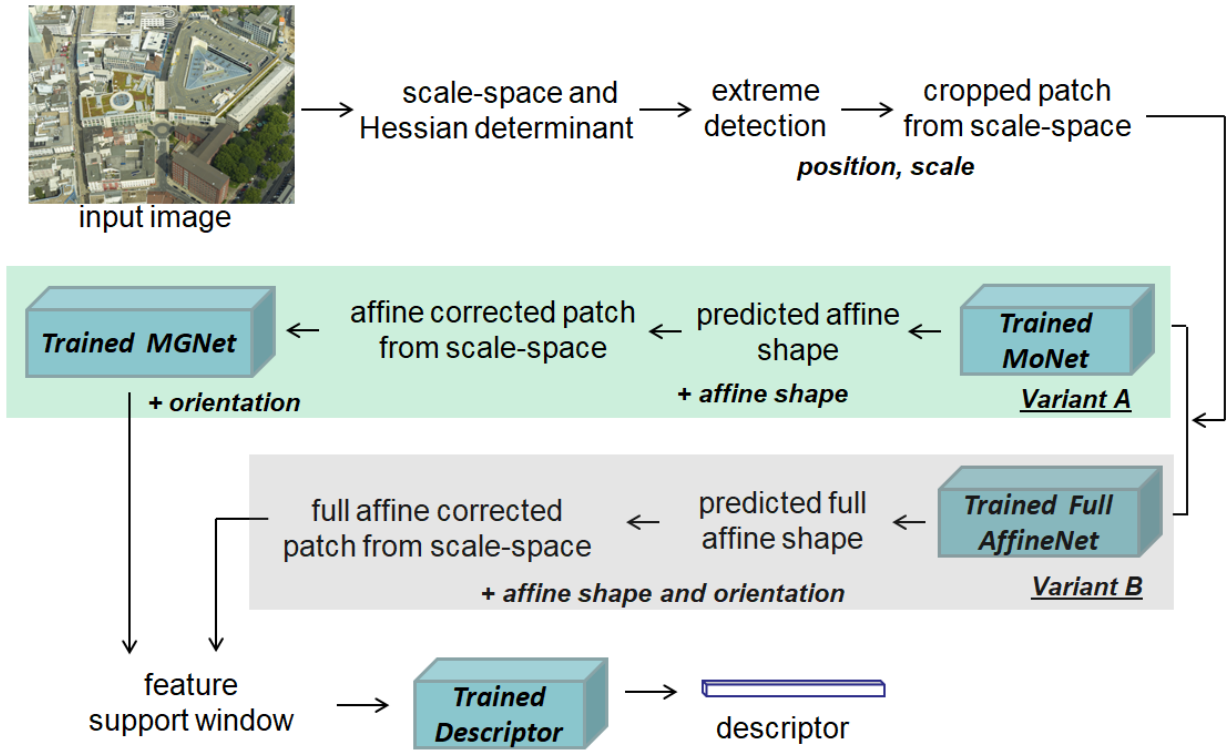



Figure 4.16.: Inference of the whole pipeline. The input image first goes through a scale-space feature detection module to obtain its features, represented by position and scale. Then, the trained affine shape network is applied to predict affine shape parameters, through which the patches around features are resampled and further applied to the trained orientation network. Then, the patches are resampled to remove the rotation ambiguity and used as the input for the trained descriptor network to obtain descriptors. Alternatively, the affine shape and orientation are estimated with the trained Full Affine network in one step, as indicated by variant B which is shown with a grey background box.

Subsequently, a patch is resampled around the detected feature position with a size proportional to the characteristic scale. This patch is regarded as input for the affine shape network to predict its estimated affine shape, which, in turn, is used in the next step to compensate for the affine distortion of local patches. In the following step, the orientation network is applied to the patch corrected for affine distortion in order to estimate rotation. The patch is then further corrected by the estimated rotation angle and forms the feature support window for the local feature. Those two steps correspond to the variant A in figure 4.16. Alternatively, the resampled patch after feature detection is fed into the trained Full AffineNet, then the full affine shape for the input feature patch is predicted and further corrected in scale-space. The corrected patch

is used as feature support window to compute descriptors.

In the next step, the trained descriptor network is applied to the obtained feature support windows and descriptors for the underlying local features are computed. It should be noted that whenever resampling is necessary, all related parameters are combined and only a single resampling step is carried out.

If the image size is too large, the image is divided into several tiles without overlap and the inference pipeline is run on each image tile, followed by a combination of all features and descriptors computed from each tile of the image.

## 4.7. Discussion

The model assumptions of each module are discussed in this section, as well as the advantages and limitations of the proposed method.

### 4.7.1. Descriptor Learning

The quality of the descriptor trained in this thesis depends on the assumption that after training the descriptor network, the trained descriptors can achieve invariance against a certain level of transformation, which is contained in the data fed into the descriptor network for training. When homologous features from different images are described and matched, certain difference are normally present, mainly owing to the fact that the detected feature locations and the estimated orientation and affine shape are not perfectly aligned since they always contain some errors. Those errors must be taken into consideration when shaping the descriptor learning process. One way of achieving this is to use data containing real feature detection noise, for instance the Brown dataset. However, obtaining this type of data is expensive, because it requires necessary data such as the ground truth three-dimensional point cloud derived through multi-view dense matching and exterior orientation parameters of involved images. In order to improve the required invariance, this thesis suggests using the weak match branch to find challenging patches containing a certain level of geometric transformation in each iteration. Those patches are used to optimize the parameters of the descriptor network in each iteration. By using this active weak match finding strategy, the dependence on the variation contained in the training patch is alleviated.

In addition to invariance, the descriptor should be capable of differentiating features imaged from different three dimensional positions, namely achieving good discriminability. To achieve this, it is crucial for the descriptor to "see" enough variance among the image patterns contained in feature support windows, i.e., in training patches. This means that the dataset used for training should contain scene with an abundance of texture and variations of local features. During the learning stage, whether a feature can be differentiated from others is reflected through the Euclidean distance based loss regarding unmatched pairs. Thus, it also depends on the model assumption according to which the used Euclidean distance in feature descriptor space

is an adequate similarity measure for feature matching. In order to be consistent with this similarity measure, all the later evaluation tasks in the experimental chapter use the Euclidean distance of descriptors to measure the similarity between features as well.

If the images which to be matched contain repetitive patterns, non-conjugate input patches look similar which in turn, leads to the fact that also the output descriptors are similar to each other. Consequently, matching those features will be a challenging issue. This is also due to the fact that the learned descriptor only considers local patterns. One possible way to cope with this situation is to incorporate constraints that ensure the consistency in the order in which matched features appear in the images. This challenging case, however, will not be addressed in this thesis and is left to future work.

### 4.7.2. Affine Shape Estimation

An important dependence is that both stretch and skew loss are well-trainable within a deep learning framework. In the classical theory dealing with affine shape estimation, iterative procedures based on applying the transformations determined by the inverse of the square root of the second moment matrix are used, e.g., Mikolajczyk and Schmid [2004]. However, as stated in Mikolajczyk and Schmid [2004], a considerable number of features is removed after the iterative affine adaptation algorithm and only 20-30% of the initially detected features are preserved for further feature matching. In this thesis, the iteration process in the above mentioned method is replaced by the optimization of the affine shape estimation network, whereas the objective function of both ways remains very similar to each other.

Another one of the most notable advantages of this method is that a clear goal for affine shape correction is defined. For any patches, the number of constraints designed in the loss term is equal to the degree of freedom contained in the corresponding transformations. Therefore, a canonical solution exists for each feature. Compared to the previous work [Mishkin et al., 2018; Chen et al., 2020a], descriptor distance is not utilized as similarity measure for the suggested method in this thesis. This contributes to a simpler, yet comparable solution. The matching relationship used for pair sampling in training is not needed. In consequence, only image patches surrounding features are required, which significantly lowers the difficulty of obtaining training data.

### 4.7.3. Orientation Assignment Learning

The core assumption of orientation assignment is that the vector built by the mean gradients in x and y direction is reliable enough to indicate the rotation of a feature and is well-trainable, too. Although the mean gradient in x and y direction is calculated in a simple way, it is a powerful indicator for most local image patches. After applying rotation transformations to patches, an angle can be derived by using the $arctan$ function with normalized mean gradients in x and y direction on the rotated patches. Through this experiment, it is found that the mean gradients can effectively and sensitively indicate the orientation of

image patches. Moreover, the calculated mean gradients change continuously with respect to different angles (except for angle $\pm\pi/2$). In comparison, the more widely used histogram of gradients based orientation assignment strategy, as suggested in Lowe [1999], is not differentiable because gradients are quantized for calculating the histogram. This differentiable property plays a key role for the learning procedure.

Similar to the affine shape estimation module, a canonical form of feature is achieved. Therefore, the optimized solution of feature orientation for different patches is unique, corresponding to the canonical feature patch. The dependence on feature descriptor distance, as employed in Yi et al. [2016b,a]; Mishkin et al. [2018]; Chen et al. [2020a], is eliminated.

If the patches contain radial symmetric patterns, the mean gradients derived in x and y direction are less sensitive to rotations of the patches. This means the training data should not be dominated by symmetric patterns.

### 4.7.4. The Inference Pipeline

The inference pipeline shares all of the aforementioned assumptions for the different modules. For the feature detection in scale space, it relies on the Hessian detector, which has been proved a better invariance against viewpoint and viewing direction change, in comparison to other hand crafted feature detectors [Mikolajczyk et al., 2005]. Extensions of the approach to learn the feature detector, similar to the work in Lenc and Vedaldi [2016]; Yi et al. [2016b], can be integrated into this approach to further improve the invariance of feature detection against viewpoint and viewing direction change. This part of feature detector learning, however, will also be left to future work.

# 5. Experiments and Results

The focus of this research is to apply deep learning methods to improve the image matching performance of local features against large viewpoint and viewing direction change. To address this topic, descriptors are designed to tolerate a limited level of affine distortion in local features, whereas a higher level of affine distortion is designed to be handled by the trainable affine shape and orientation assignment module. This chapter presents a series of experiments designed to evaluate and analyse the performance of the method proposed in this thesis.

The last chapter explained the method for training descriptor, affine shape and orientation assignment as well as the full affine shape estimation module, which aims at boosting the matching performance against large viewpoint and viewing direction change. This chapter assesses the performance of the learned modules. With this in mind, four different experimental tasks are included in this chapter and the goal for each of them are listed below:

- Task A - Patch based Image Matching: Test how the trained descriptor performs when different levels of transformation are contained in a patch based image matching task[1]. The module involved in this experiment is the learned descriptor network.
- Task B - Descriptor Distance Analysis: Find out how the descriptor distance of matched feature pairs changes when different kinds of transformations are applied to the features. The module involved in this experiment is again the learned descriptor network.
- Task C - Feature based Image Matching: Compare how well the inference pipeline can match images in two different cases: A) imagery containing a full range of in plane rotation change; B) images including large viewing direction change, resulting in large affine distortions and a limited level of in-plane rotation. In both cases, ground truth results are available. The modules involved are the learned descriptor, affine shape, orientation and full affine shape network.
- Task D - Image Orientation: Small blocks of oblique aerial images are utilized to assess the performance of the proposed method in image orientation tasks. The bundle adjustment results are used to analyse and assess the performance. The modules involved are the same as in task C.

A summary of the listed experiments is illustrated in figure 5.1. As can be seen, for the affine shape and

[1] Patch based image matching means that image patches are provided as the input for running description and descriptor matching. Those image patches are extracted surrounding features, which can be either detected or projected from other image views.

orientation of features, both options of separating them into two steps (variant A) and combining them into one full affine network (variant B) are tested.

First, the learned descriptor is evaluated to reflect its performance for patch based image matching. A descriptor, as a fundamental part of several of the tasks of this thesis, is designed to be discriminative and robust against certain amount of geometric deformation in image patch data. In order to evaluate the performance of the learned descriptor, the dataset HPatches [Balntas et al., 2017] is selected as the test benchmark for task A.

Second, the descriptor distance of patch pairs under different kinds of transformation is computed and analysed. Through this analysis, more knowledge about how the descriptor distance change with respect to different geometric transformations is gained. The dataset used in this task requires the matching relationship between feature patches, so that patch pairs are sampled according to the provided matching relationship.

Third, the proposed feature and descriptor extraction pipeline is evaluated on standard image matching benchmarks containing large viewpoint and viewing direction change with known matching relationships. In addition to the descriptor part, the influence of the affine shape estimation and orientation assignment modules is also included. Standard image matching benchmarks provided with ground truth geometric relationships are used for testing the performance. The whole pipeline is used to extract features and descriptors for images contained in the test benchmark.

Finally, as standard image benchmarks provided with ground truth geometric relationship represent a relatively ideal case, the performance of the whole pipeline in real applications remains unknown. Among all the real applications with local image features, image orientation is one of the most attractive and fundamental tasks in the photogrammetry and computer vision. Small image blocks taken from an aerial penta-camera system including nadir and oblique images with significant changes in viewpoint and viewing direction, the features and descriptors of which are extracted with the whole pipeline, are selected for assessing the proposed method. In a full block of images captured by aerial penta-camera systems, images with larger differences in viewpoint and viewing direction can often be matched, because the block contains images in between so that features can be tracked across the block. To decrease this influence, some adjacent views in the full block are removed and only small blocks of images are used to form the test blocks. Therefore, compared to the coverage of a full block of images, a smaller number of viewpoints and viewing directions is covered. The quality of image orientation parameters and the computed 3D points after bundle adjustment with the derived features as input are used to assess the results. Note that since the overlap of the images used is partly less than the standard for aerial triangulation, the precision of the orientation parameters and the 3D coordinates of the tie points is expected to be larger than in normal cases also. This part forms the main assessment of the proposed method.

Additionally, necessary hyper-parameters for the descriptor and affine shape network are studied. Specifically, the relative weight of the weak match loss in descriptor learning is studied in section 5.3 and the relative weight of skew loss is studied in section 5.5.

The following section 5.1 introduces the datasets used for descriptor, affine shape, orientation assignment, as well as full affine shape module learning. In the next section, the evaluation criteria of the aforementioned evaluation tasks (Task A, B, C and D) are explained. Sections 5.3 through 5.6 provide the evaluation results for Task A, B, C and D, as well as the training details and the result of the hyper-parameter study.

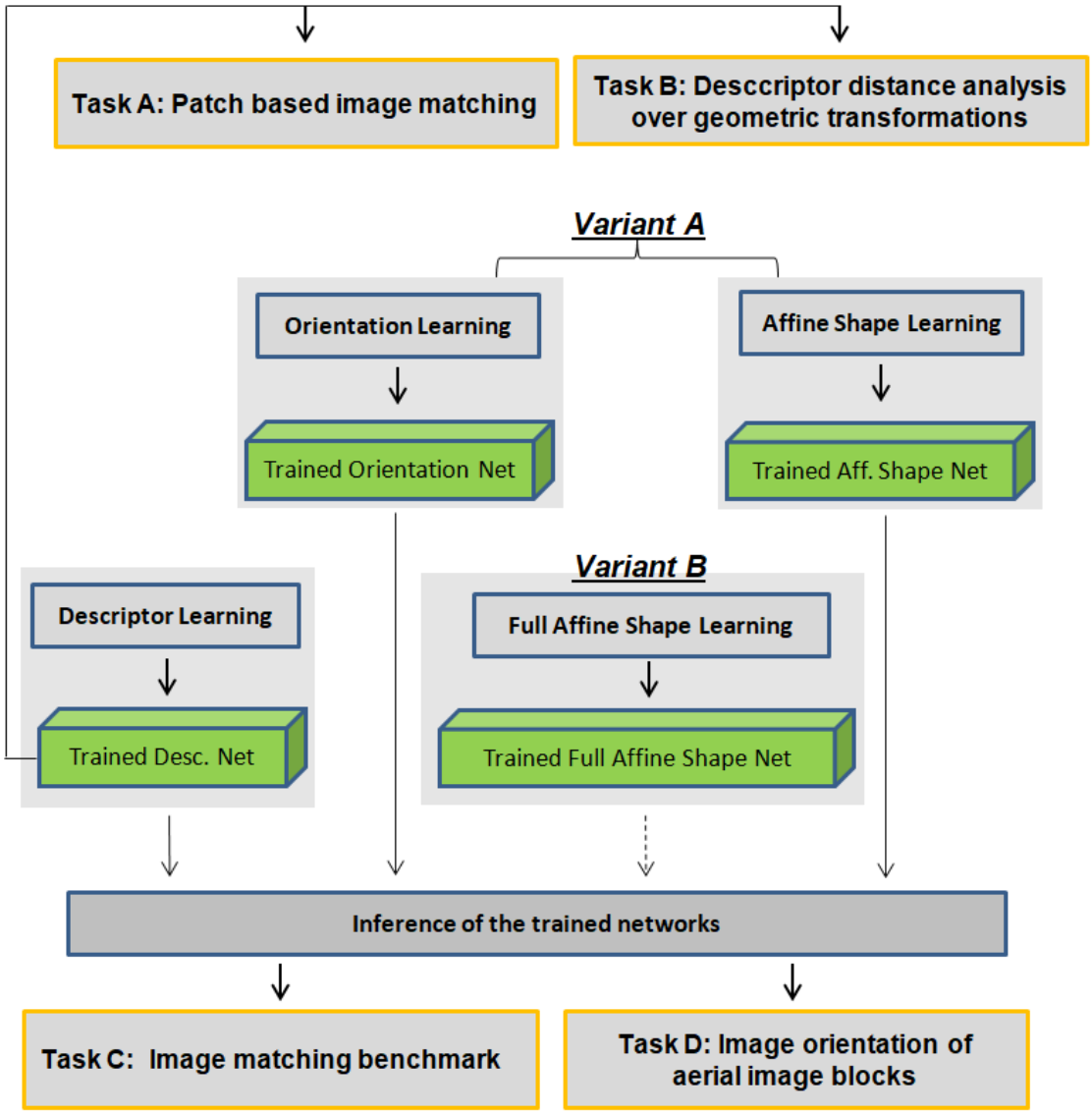



Figure 5.1.: Overview of the experiments. The four designed experiments (Task A, B, C and D) are shown in the orange boxes. Task A and B rely only on the trained descriptor while Task C and D rely on all of the trained modules. For task C and D, Variant A and Variant B are treated as alternative variations of the method.

## 5.1. Datasets

In this section, all the datasets used in this research are introduced. It contains two parts: the dataset for training and that for evaluating the result of Tasks A, B, C and D. A summary of all involved datasets in this research is provided in table 5.1.

| Main Usage | Dataset Name | Cameras and Imaging Platform | Patch or Image? | Notes |
|---|---|---|---|---|
| Train | Brown Dataset | consumer camera, street view | patch | 3 subsets |
| | Aerial-Graz | oblique camera, aerial view | patch | – |
| Test | Hpatches | consumer camera, close range view | patch | 2 subsets |
| | Aerial-Dortmund | oblique camera, aerial view | patch | – |
| | Hpatches Image Sequence | consumer camera, close range view | images | 2 subsets |
| | Oblique Aerial Image Blocks | oblique camera, aerial view | images | 4 subsets |

Table 5.1.: Datasets involved in this research.

### 5.1.1. Datasets for Training

Datasets for the training of descriptor, affine shape, orientation assignment and full affine shape network are introduced in this subsection.

#### Brown Dataset

To train the descriptor proposed in this thesis, i.e. WeMNet, the Brown Dataset [2] [Brown et al., 2011] is used. To generate the Brown Dataset, multi-view datasets that contain large numbers of images (community photo connections) [Goesele et al., 2007] are utilized. Through structure from motion and dense multi-view stereo matching for the community photos [Snavely et al., 2008], image orientation results and dense surface models are obtained. In the following step, as mentioned in section 3.3.2 and explained in Brown et al. [2011], for a feature $f_L$ on image $I_L$, a small, uniform and dense grid surrounding $f_L$ is sampled and transferred to image $I_R$ through the depth map estimated between the stereo image pair $I_L, I_R$. The transferred grid is a group of image points which represent the extent of matched features point $f_R$ on $I_R$. The localization, scale and orientation of $f_R$ are estimated by least square which minimizes the fitting error of sample grid points on $I_R$, i.e., computing the best fitted local feature frame represented by translation, rotation and scale. By running this process for each of the feature points on $I_L$, the ground truth matching feature on $I_R$ is obtained. If the estimated scale and pixel localization for the transferred features point are close to[3] the scale and localization

[2] http://matthewalunbrown.com/patchdata/patchdata.html (accessed on Jan 26,2021)
[3] Specifically, 5 pixels for position, 1/4 octaves for scale and $\pi/8$ for rotation.

of a detected feature point $f_R$ on $I_R$ , $f_R$ and $f_L$ are judged as a ground truth match pair.

By running the process mentioned above, true matches between any two connected images during stereo matching are collected and, finally, matched features from different visible images are tracked to form the training patch dataset. It contains a large number of 3D points, each of which has several visible features from different views. The patch extent is extracted with a size of 6 times the characteristic scale and rescaled to $64 \times 64$ pixels. One of the most important advantages of this method is that it models the true interest point noise in the data [Brown et al., 2011].

In total, there are three different subsets: NotreDame, Liberty and Yosemite. For each subset, there are two versions of dataset generated by using different feature detectors. Concretely, one version uses the features detected by Difference of Guassian (DoG) detector and the other one uses the Harris detector. One sample of a matched feature for a same 3D point is shown in figure 5.2. A limited level of transformation among different patches, as shown in figure 5.2, is contained in this dataset.



Figure 5.2.: Feature patches that corresponds to a 3D point in the NotreDame set of the Brown Dataset [Brown et al., 2011]. This 3D point contains 6 different visible match feature patches, i.e., 6 different views of images.

In order to better compare and analyse the performance of MGNet, MoNet and Full-AffNet trained on different datasets, the brown dataset is also used to train MGNet, MoNet and Full-AffNet in task C and D.

#### Aerial-Graz Dataset

To generate the training image patches for affine shape and orientation estimation, large blocks of images are used to generate training pairs. An image block containing 721 images was collected and provided for this research by the company Vexcel under the framework of VOLTA[4] (innoVation in geOspatiaL and 3D daTA) project. The images are taken using the Vexcel UltraCamera Osprey camera[5], which is an aerial oblique camera system with one nadir camera and four side looking cameras (forward, backward, right, left camera). The side looking cameras have oblique angles of 45 degrees. The data used was taken in Graz, Austria, and the flying height was 1000m. The ground sampling distance (GSD) for the nadir view is 6.5cm and it varies from 5.0 to 7.5cm for oblique views.

The images are matched and put through the open-source Structure-from-Motion (SfM) software COLMAP[6]

[4]https://volta.fbk.eu/
[5]https://www.vexcel-imaging.com/ultracam-osprey-4-1/
[6]https://github.com/colmap/colmap

[Schönberger and Frahm, 2016] to obtain the image orientation result. Using this SfM, 712.000 3D points are obtained. Each of those 3D points has at least two visible views and thus at least two image features from different images. The average number of visible views distributed over all 3D points is called the mean track length, which indicates the redundancy level of collected 2D feature points. For the processed 712.000 3D points, a mean track length of 3.56 is obtained. For each detected feature that is also matched and reconstructed by SfM, a surrounding context window with a radius of 12 times the detected scale is cropped and resampled to a $96 \times 96$ pixels patch. This dataset is named as Aerial-Graz in the following sections of this thesis. An exemplary feature track of this dataset is shown in figure 5.3.

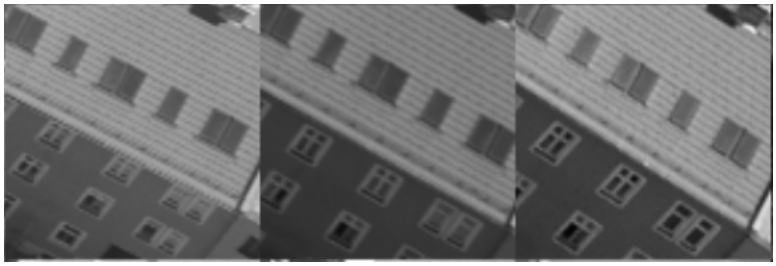


Figure 5.3.: An exemplary feature track of the Aerial-Graz dataset. The three feature patches correspond to one identical 3D world point and are taken from different views.

Once patches have been generated, mini-batches are sampled for the training of affine shape, orientation and the full affine network. As mentioned before, the sampling process does not rely on the matching relationship. In theory, the training of affine shape, orientation and the full affine shape network only requires image patches surrounding detected features, as all the relevant networks (MoNet, MGNet and Full-AffNet) are only trained with one branch and the matching relationship is not needed to compute the training loss. In contrast, in the other closely related work on AffNet [Mishkin et al., 2018], a Siamese CNN is used and only the feature patches that survive image orientation (or SfM) are used for training. To make the comparison between those two methods as fair as possible, the same type of feature patches, i.e., those which survived SfM, is used. Undoubtedly, using detected features without any further processing via SfM would do just as well for training the proposed MoNet, MGNet and Full-AffNet.

In this thesis, Aerial-Graz is also used to train the AffNet and OriNet proposed in Mishkin et al. [2018], in order to compare those two networks to the networks proposed in this thesis using the identical dataset.

### 5.1.2. Datasets for Testing

#### Dataset for Patch based Image Matching (Task A)

To evaluate the learned descriptor, the Hpatches benchmark [7] [Balntas et al., 2017] is used. This dataset contains image patches that are generated from two groups of images: illumination (57 subsets) and viewpoint

[7] https://github.com/hpatches/hpatches-benchmark

(59 subsets). In the illumination group, images are taken with the same camera parameters (related by identity homography matrix) but varying illumination. The viewpoint group of images depicts planar scenes acquired from different viewpoints. A horizontal tilt constitutes the major transformation between images. In each subset of these images, there are one reference image and five target images. The matching relationship is provided by a ground truth homograph matrix between each target image and the reference image. According to Balntas et al. [2017], once features have been detected in the reference images by a DoG feature detector, they are perturbed with different amounts of rotation, anisotropic scaling and translation to simulate the noise in feature detection facing viewpoint change. Based on the level of perturbation, the noise is classified into three categories, namely easy, hard and tough. The level of perturbation increases gradually from images in the easy category to the tough ones. Those perturbed features are then projected from the reference image to target images through ground truth homographies before being resampled. In this context, it should be noted that the features from the target images are not obtained by feature detection. Instead, they are the result of projection of perturbed features in reference image onto the target images. Then, features are scaled to five times their characteristic scale in order to extract patches which may be used for descriptor computation. The patches are all rescaled to 65 x 65 pixels and the features with a scale smaller than 1.6 px are discarded because larger scaling factors are needed for those small scale features to generate patches and thus re-sampling artefacts can be involved. Exemplary pairs are shown in figure 5.4. An increasing level of distortion can be observed in the image as the perturbation level increases from easy to tough. Based on those patches, patch verification, image matching and patch retrieval tasks are designed in order to evaluate the performance of different feature descriptors [Balntas et al., 2017]. In this thesis, only the image matching task is taken into use, as feature matching is the topic of this work.

#### Dataset for Descriptor Distance Analysis (Task B)

To analyse how the descriptor distance is changed by different kinds of geometric transformation applied to input patch pairs (Task B), two different datasets are used. The first one is the Brown Dataset, which is also used in descriptor learning. The second one is a patch dataset generated in a similar way as Aerial-Graz, but with different aerial image blocks taken by an IGI penta camera. This dataset was provided by the IGI company, Germany. The employed oblique camera system comprises one nadir camera and four side cameras (forward, backward, right, left camera). The side looking cameras have oblique angles in range of 42 to 45 degrees. In total, 181 images are contained in this dataset. The images in this block were mainly taken at a flying height of 620m. Some additional photos were taken at a flying height of 370m and 960m. For the images taken at the height of 370m, the GSD in nadir and average oblique views are 1.9cm and 2.7cm, respectively. For images with the flying height 620m, the GSD is 3.2cm for nadir views and 4.5cm on average for oblique views, and for images taken at the flying height of 960m, the GSD for nadir views is 4.9cm and the average GSD for oblique views is 6.9cm.

The same process used to generate Aerial-Graz, as mentioned in section 5.1.1, is employed to compute the image orientation result for this second dataset. 157.000 3D points with a mean track length of 4.32 are

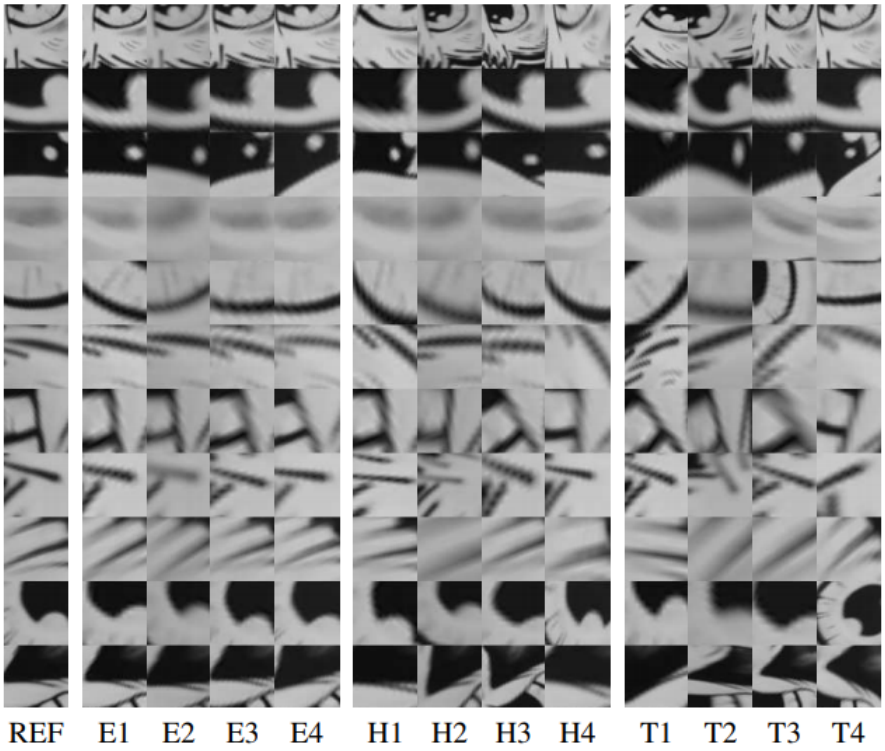



Figure 5.4.: An example of Hpatches dataset [Balntas et al., 2017]. Each row represents one patch from the reference image (in the first column) and different easy (E1-E4), hard (H1-H4), and tough (T1-T4) matched patches from other target images.

obtained. To generate pairs for task B, 3D points are first picked and then two corresponding features for each of these selected 3D points are randomly chosen to form a pair. Those pairs are further used for the descriptor distance analysis experiment (Task B).

#### Dataset for Feature-based Image Matching (Task C)

The dataset used in this task is Hpatches image sequence[8] [Balntas et al., 2017], which contains the same images as the aforementioned Hpatches dataset. Compared to Hpatches, the images, instead of feature patches, are used as input for image matching. In this task, only images from the viewpoint change subset are used. The viewpoint subset is composed of 59 small groups of images taken for planar scenes. Each of the small datasets contains homography matrices, which describe the geometric relationship between the reference image and each target image. Two groups of Hpatches image sequences from the viewpoint subset are shown in figure 5.5.

Two cases are tested in task C, i.e. the rotation and affine transformation. For the first case, the reference image in each subset is transformed with 12 different rotation degrees, therefore 12 rotated images are obtained for

[8] http://icvl.ee.ic.ac.uk/vbalnt/hpatches/hpatches-sequences-release.tar.gz (accessed on Nov 13, 2020)



(a) azzola image 1 (b) azzola image 3 (c) azzola image 6

(d) blueprint image 1 (e) blueprint image 3 (f) blueprint image 6

Figure 5.5.: Two examples of Hpatches sequence images taken from the viewpoint subset . The first column (a) (d) shows the reference image [image 1] of subset "azzola" and "blueprint"; the second column (b) (e) and the third column (c) (f) are the second [image 3] and fifth [image 6] target image in these two subsets.

each subset. This dataset is called Hpatches-Rot. To generate the rotation set for each subset in Hpatches image sequence, the full rotation range is first equally divided into 12 regions from $-180°$ to $180°$ and then a rotation angle in each region is randomly sampled with a uniform distribution. In addition, a random scale factor in the range of $[0.9 - 1.1]$ is uniformly sampled and applied to the input image. Therefore, the major transformation in this dataset is rotation, but a small range of scale change is also contained. For the second case, namely affine transformation, only the reference image and the "image6" in each subset, i.e., the one containing most challenging viewpoint and viewing direction changes, are used. This dataset is called HPatches-Aff. As the ground truth homography matrix is always available, the ground truth matches can be derived for the rotation and affine set.

#### Dataset for Image Orientation Tasks (Task D)

Apart from testing the performance of learned descriptors, feature affine estimation, orientation and the full affine shape network on patch-based benchmarks and image matching benchmarks, its performance in real image orientation tasks is assessed. This part of the assessment is defined as Task D in this thesis. Image blocks containing images taken by an oblique aerial camera system, again, are used for Task D, mainly due to the fact that the obtained images contain large viewpoint and viewing direction changes, which serve as a suitable test-bed for the method proposed in this thesis.

The datasets used in Task D are composed of four different image blocks. All of these image blocks are taken from the airborne oblique images contained in the ISPRS/EuroSDR benchmark for multi-platform

photogrammetry [9] [Nex et al., 2015]. An IGI penta camera system was used to take the images for this benchmark (note that it was an older version than the one used for acquiring the Aerial-Dortmund data mentioned before). It provides two different sets of image blocks: "Zeche Zollern" and "Dortmund". In "Zeche Zollern", terrestrial objects forest, farmland and low buildings, are depicted in this set, whereas "Dortmund" was taken in Dortmund and the images are dominated by dense city blocks. In order to investigate the performance of the method proposed in this thesis considering different types of landscape, both datasets are used. Two image blocks (block1 and block2) are taken from the "Zeche Zollern" dataset and another two (block3 and block4) from the "Dortmund" dataset. The ground sampling distance for the two blocks is identical: it is 10cm for the nadir view and varies from 8cm to 12cm for the oblique views [Nex et al., 2015]. Also, the flying height for both datasets is identical: it is 830m. The image dimension is $8176 \times 6132$ pixels for all four blocks. An example of five images of block4 is shown in figure 5.6, which highlights distinctive viewing directions and viewpoints for images in the image block.

As stated before, both city and suburban imagery are used to investigate the performance of the proposed method for different types of landscapes. Specifically, the first block covers farmland, forest and low buildings while the second block is dominated by residential areas and low buildings but also contains a few forest and some farmland. The third and fourth block contain dense building blocks, road and some trees, which are typical terrestrial objects in a city scene. For a closer look of the typical nadir image used in each block, please see the figure 5.7.

The types of content and the configuration of each block are shown in table 5.2. For each block, three to five images of each viewing direction are used, a closer look at the selected views of each block is illustrated in figure 5.8.

| block name | Dataset | #images | #nadir | #front | #right | #back | #left | main scene contents |
|---|---|---|---|---|---|---|---|---|
| block1 | Zeche Zollern | 18 | 4 | 4 | 3 | 3 | 4 | farmland, forest, low buildings |
| block2 | Zeche Zollern | 19 | 4 | 4 | 4 | 4 | 3 | residential area, low buildings, forest |
| block3 | Dortmund | 17 | 3 | 3 | 4 | 4 | 3 | buildings, city center |
| block4 | Dortmund | 20 | 4 | 4 | 4 | 4 | 4 | buildings, city center |

Table 5.2.: Details about the image blocks used for determining image orientation. #images stands for the amount of images; #nadir, #front, #right, #back, #back represent for the amount of image taken from nadir, front, right, back and back camera in a penta-camera system, respectively.

[9] http://www2.isprs.org/commissions/comm1/icwg15b/benchmark_main.html (accessed on Jan, 26, 2021)



(a) front camera



(b) left camera



(c) nadir camera



(d) right camera



(e) back camera

Figure 5.6.: Five exemplary images taken from different views in one small oblique aerial image block (block4). Images (a), (b), (c), (d), (e) are taken by front, left, nadir, right and back camera, respectively. As those five images come from different imaging stations, there is a large viewpoint and viewing direction change among any two of the images taken by different cameras.

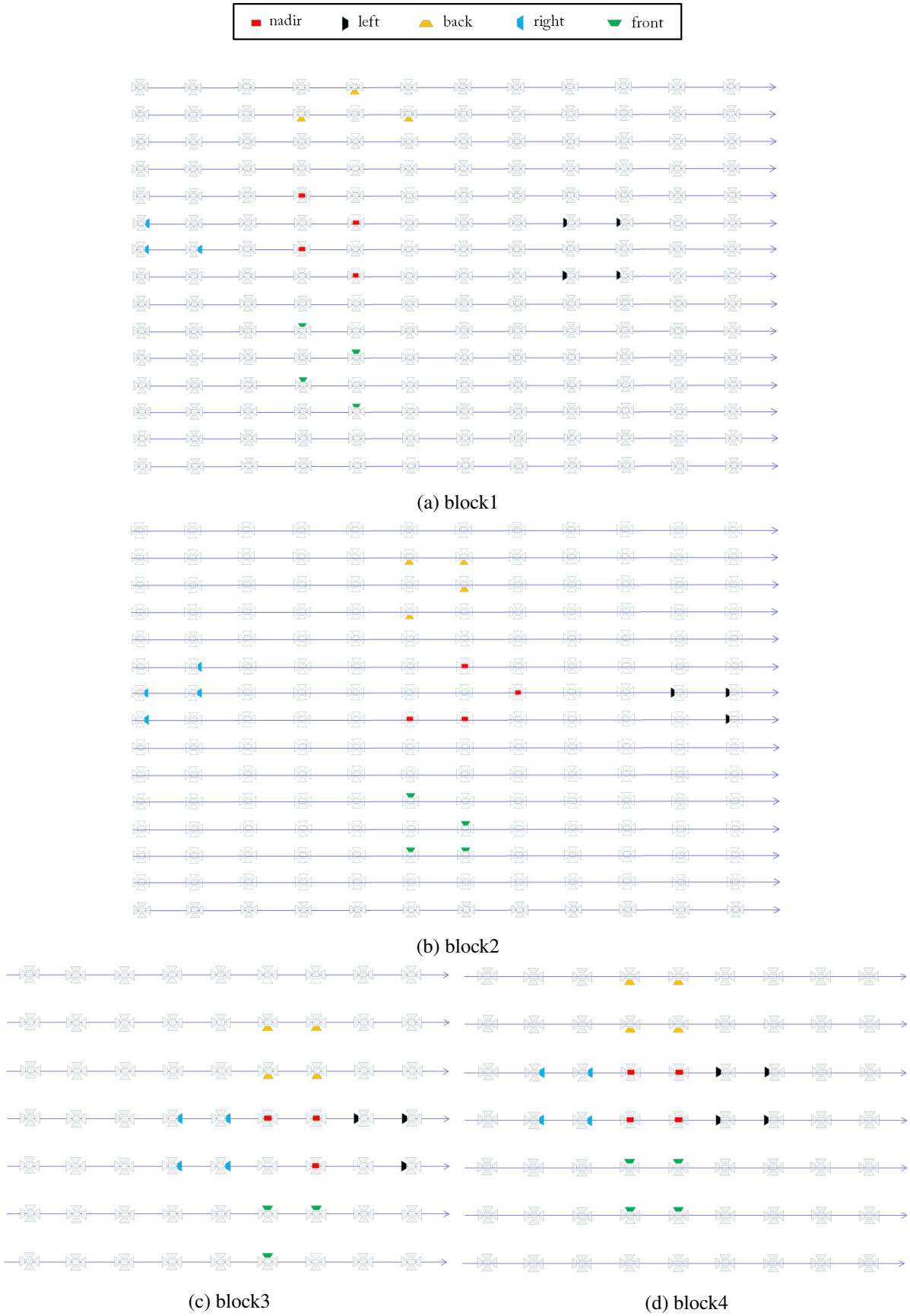
For block 1 and 2, adjacent imagery from inter and intra flight line are not included, in order to increase the difficulty of matching. For the dataset “Dortmund”, the imagery taken at every other flight line are not included for the published version. As block 3 and 4 are selected from “Dortmund”, adjacent flight lines have already been excluded. Therefore, compared to a full block, connecting images from different views of the selected 4 image blocks with matching points is more challenging.



(a) block1 (b) block2

(c) block3 (d) block4

Figure 5.7.: Typical nadir view for block1 (a), block2 (b), block3 (c) and block4 (d) used in the image orientation experiment. Block 1 contains more texture-less scene composed of farmland and forest. In comparison, block 2 includes more artificial terrestrial objects, e.g., low residential buildings and low industrial building, which result in a more abundant texture. The other two blocks, 3 and 4, contain urban buildings. A slight difference is that block 3 contains a few more higher buildings than block 4.




(a) block1

(b) block2

(c) block3

(d) block4

Figure 5.8.: The selected views used in block1 (a), block2 (b), block3 (c) and block4 (d). For each graph, the stations where images are taken using the penta-camera system are indicated by the symbol defined at the top of the figure. The blue line represents the flying line and the flight direction is shown by the arrow. The selected views for each image block are shown in colour according to the colour bar shown at the top of this figure.

## 5.2. Evaluation and Analysis Criteria

In this section, the criteria of the evaluation task A, B, C and D are explained.

### 5.2.1. Task A: Patch based Image Matching

For this task, a collection of features from pairs of images containing a reference $L_0$ and target image $L_1$ is used. Features on the reference image $L_0$ $(x_{i0}, i = 1, 2, ..., N)$ are detected and features on target images, $(x_{i1}, i = 1, 2, ..., N)$, are generated by projecting features detected on reference image $L_0$ with noise perturbation. Thus, each feature of $x_{i0}$ is certain to have a correctly matched feature $x_{gt_i 1}$ and $gt_i = i$. By varying the perturbation level from easy to tough, the matching difficulty between $(x_{i0}, i = 1, 2, ..., N)$ and $(x_{i1}, i = 1, 2, ..., N)$ increases. As suggested in the corresponding benchmark software "hpatches-benchmark"[10], the nearest neighbour based on the distance between descriptors (recorded $x_{\delta_i 1}$) is taken as the matched feature for each feature $x_{i0}$ detected in the reference image $L_0$. The matching relationship $y_i$ is labelled as +1 (correct match), if $\delta_i = i$ and as -1 (wrong match), if not. Based on the decreasing order of similarity score, the matching relationships $y_i$ are sorted into $(y_{\pi_1}, y_{\pi_2}, .., y_{\pi_N})$ and the average precision $AP(y_{\pi_1}, y_{\pi_2}, .., y_{\pi_N}; N)$ is calculated as the final performance measure. In particular, $AP$ is defined as:

$$AP(y; N) = \frac{\sum_{k:y_k=+1} y_k}{N}$$

In the above equation, $\sum_{k:y_k=+1} y_k$ represents the number of correct matches and $N$ stands for the number of returned matches. Owing to the fact that the number of ground truth matches equals the number of involved features for matching, recall and precision are identical. Therefore, only average precision is used to evaluate the performance. In an ideal case, AP is 1; a higher AP indicates better matching performance.

### 5.2.2. Task B: Descriptor Distance Analysis

After descriptors are learned, it is critical to know how the distance of a matched pair change when different kinds of geometric transformations are applied to one of the local image patches surrounding detected features. To explore this issue, the descriptor distance between matched patch pairs which have undergone different kinds of geometric transformations is analysed. This analysis provides a closer look at how descriptor distance for feature pairs change by geometric transformations, therefore it acts as an important guidance for the question: under which circumstances the descriptor distance is an effective measure for the level of geometric distortions contained in local feature patch.

Among all possible types of transformations, rotation, translation and affine shape transformation are explored,

[10] https://github.com/hpatches/hpatches-benchmark

because those are common potential geometric transformations a descriptor needs to be invariant against in real applications. To explore this, a large number of matched pairs are sampled for this study. For each matched patch pair containing an anchor ($a$) and a positive patch ($p$), $a$ is fixed and $p$ is transformed with different transformations sampled by different transformation parameters. Then a functional relationship is explored in this study: the distance between the descriptor of $a$ and the descriptor of transformed $p$ is treated as dependent variable, while the parameters determining the transformation applied to $p$ is treated as independent variables. By applying geometric transformations, the descriptor distance is attained as the average value for the whole patch pair set. Through this process, a better knowledge of how the descriptor distance changes by different geometric transformations is gained.

### 5.2.3. Task C: Feature based Image Matching

In this task, the matching is conducted for full images. The images contain planar scenes and the relationships between image pairs are described by homography. After the features are detected and described, the matching relationship is predicted using the extracted descriptors. In addition, the ground truth matches between different images are available.

A pair of reference and target images is matched using the matching variants composed by the modules proposed in this thesis. These matches are checked by comparing them to ground truth correspondences. Features that are consistent with ground truth correspondences are considered as correct matches. Once the numbers of initial matches (#$matches$), correct matches (#$correct_matches$) and ground truth correspondences (#$gt_matches$) are known, recall and precision are computed following the commonly applied procedure:

$$\begin{aligned} recall &= \frac{\#correct_matches}{\#gt_matches} \\ precision &= \frac{\#correct_matches}{\#matches} \end{aligned} \tag{5.1}$$

By varying the distance threshold used in matching, the number of computed matches changes and thus both #$matches$ and #$correct_matches$ vary accordingly. This, in turn, leads to the change of recall and precision. Therefore, a curve in 2D space spanned by recall and precision can be drawn. The area under the curve (AuC) is then taken as the evaluation criterion for a pair of input images that are matched. A higher AuC indicates a better feature matching performance.

### 5.2.4. Task D: Image Orientation

Besides evaluating descriptors and other learning modules using the above benchmarks, results of the proposed method used in real applications are also assessed, using an oblique aerial image orientation task as an example. For a block of oblique aerial images, quality measures based on the image orientation result after bundle adjustment are used to assess the quality of the feature matching algorithm. In particular, the

following criteria are reported:

- Number of registered images ($N_{img_reg.}$), i.e., the number of images for which the orientation parameters could be determined. This number is the very first criteria to evaluate the performance of the matching algorithm. Obviously, a larger number of registered images indicates stronger matching performance.

- Number of reconstructed 3D points ($N_{3DP_rec.}$). Under the condition that features on all images are detected with identical detectors and that an equal number of features is obtained for each image, the number of reconstructed points can indirectly reflect how well the features are matched in the blocks. A larger number of reconstructed 3D points indicates more features are correctly matched.

- Average number of matches per image used in the bundle adjustment. Again, a higher number means that more features are matched and used in bundle adjustment.

- Number of intersecting rays per 3D point, the histogram of which reflects the redundancy level of observations for 3D points. For each 3D point, depending on which images these rays come from, four cases are distinguished:

  - A) from only one camera (nadir or oblique) (*only_nad_or_obl*);

  - B) from the nadir and one oblique camera (*nad_obl*);

  - C) from the nadir and at least two different oblique cameras (*obl_nad_obl*);

  - D) from two or more different oblique cameras (*obl_obl*).

  The difficulty of feature matching increases from level A to D as the viewpoint and viewing direction changes between images increases accordingly.

- The estimated standard deviation after bundle adjustment $\sigma_0^{post}$. It measures the overall deviation between data and model.

- Precision of the 3D point coordinates obtained via error propagation. Higher precision indicates better quality of the whole image orientation pipeline.

- The quality of matches. The number of initial matches ($N_{match}^{initial}$), the number of matches after two-view geometric verification using RANSAC ($N_{match}^{inl_2view}$) and the the number of matches which survive after bundle adjustment ($N_{match}^{surv_BA}$) are reported. Correspondingly, the ratio of inliner matches ($R_{initial}^{inl_2view}$) = ($N_{match}^{inl_2view}/N_{match}^{initial}$) and the ratio of surviving matches after bundle adjustment ($R_{surv_BA}^{inl_2view}$) = ($N_{match}^{surv_BA}/N_{match}^{inl_2view}$) are reported.

- Distribution of matching points in the image plane. It should be noted that only the matching points which survive after bundle adjustment are considered. A more even distribution of matching points

in the image plane reduces the risk of running into degenerate cases and contributes to more reliable parameters.

For a comparison using different matching variants, the one leads to higher $N_{img_reg.}$ and $N_{3DP_rec.}$ indicates a better matching performance. Correspondingly, a higher number of observations per image also indicates a better matching performance. For the intersecting rays per 3D point, more matches for harder cases represents better invariance against larger viewpoint and viewing direction change. The precision of 3D object points relies on the matching relationship as well as the localization accuracy of detected features. The two ratios $R_{initial}^{inl_2view}$ and $R_{surv_sfm}^{inl_2view}$ provides a closer look at the robustness of the matching performance. In addition, the distribution of matching points is an important factor, an even distribution is mandatory for the computation of correct image orientation.

The detailed result for this image orientation task (Task D) will be reported in section 5.6.

### 5.2.5. Summary of Tasks and Involved Datasets

Before presenting the detailed results from next section, a summary of the training and test dataset, as well as task description for tasks A, B, C and D is provided in table 5.3. For each task, a different training and test dataset are used. For the subtasks in task C and D except for C1, the involved affine shape, orientation and full affine shape network are trained both on the Brown and the Aerial-Graz dataset. In this way, a better understanding of the generalization of the networks proposed in this thesis is achieved.

| Task | Involved Modules | Subtask | Training Dataset | Test Dataset | Task Description |
|---|---|---|---|---|---|
| A: Patch Based Image Matching | Descriptor | A1 | Brown Liberty | Hpatches (View, Illumination, Full) | Parameter study $\lambda_{wm}$ for weak match network |
| | | A2 | Brown Liberty and Brown All | Hpatches ("a", View, Illumination, Full) | Comparison of average precision with other descriptors |
| B: Descriptor Distance Analysis | Descriptor | - | Brown Liberty | NotreDame, Aerial-Dortmund | Sensitivity of descriptor distance as factor of translation, rotation and affine transformations |
| C: Image Matching | Descriptor, Affine Shape, Orientation, Full Affine Shape | C1 | Aerial-Graz | Hpatches image sequence | Parameter study $\lambda_{skew}$ for affine network |
| | | C2 | Aerial-Graz and Brown Dataset | Hpatches-Rot | Comparison of orientation network performance |
| | | C3 | Aerial-Graz and Brown Dataset | Hpatches-Affine | Comparison of feature affine network performance |
| D: Image Orientation | Descriptor, Affine Shape, Orientation, Full Affine Shape | - | Brown dataset and Aerial-Graz | Oblique Aerial Image Block 1, 2, 3, 4 | Image orientation quality using oblique aerial image blocks |

Table 5.3.: The involved modules, training/test dataset and the task description for all the experimental tasks in this thesis.

## 5.3. Descriptor Learning and Patch Based Image Matching

In this section, results related to descriptor learning are provided and analysed. First, this section provides the parameter study for $\lambda_{wm}$, which controls the relative importance between descriptor loss and weak matched descriptor loss. Then, the trained descriptor is compared to other closely related descriptors.

In this experiment, the descriptor network training is implemented in PyTorch, based on the code published in [Mishchuk et al., 2017] [11] for the HardNet descriptor. During training, mini-batches with a size of 1024 pairs are used and trained in 10 epochs [12]. Standard gradient descent is used with the learning rate initialized at 0.05. The learning rate decay is applied exponentially after each iteration, such that the learning rate at the beginning of an epoch is decreased by 20% by the time the epoch is finished. The losses involved in descriptor learning are logged every 50 training steps, in order to visualize and analyse the change of losses throughout the whole training procedure. For each experiment conducted in section 5.3.1, 10 million matched pairs are sampled as the training dataset, while 30 million matched pairs are used for the experiments conducted in section 5.3.2. All of the descriptor networks mentioned in this section are trained using the Brown Dataset [Winder and Brown, 2007; Brown et al., 2011]. During training, the maximum stretch of the local weak match finder is set as 2.2.

In task A, four different sets of data division using Hpatches [Balntas et al., 2017] are used: “a”, “view”, “illumination” and “full”. The patch set “a” contains features detected on images found only in a subset of the "view" and “illumination” set. The sets “view” and “illumination” contain all features detected from the view and illumination set, respectively, while the set “full” contains features detected in all images. This separation yields a more comprehensive result and the performance against different types of transformation (illumination and viewpoint change) is better comparable.

### 5.3.1. Parameter Study for WeMNet

The relative weight factor of the weak match networks is investigated in this section. With the aforementioned training procedure, the descriptors are trained by varying $\lambda_{wm}$, for which the test values are 0, 0.01, 0.1, 1.0, 3.0, 5.0 and 10.0. $\lambda_{wm} = 0$ is the case for descriptor learning without weak matches. In the following step of the parameter study, those descriptors trained with different test values for $\lambda_{wm}$ are employed, in turn, to compute descriptors of patches contained in the Hpatches dataset for running the patch based image matching (Task A). The matching process is first evaluated with regard to the view and illumination sets, and then with regard to the full dataset as well. The mean average precision obtained are listed in table 5.4. As observed from the listed results, the involvement of weak matches improves the matching performance in cases of both viewpoint and viewing directions (set “View”) and illumination (set “Illumination”) change.

For a better visualization and analysis of the parameter study, the result using the full dataset is further

[11] https://github.com/DagnyT/hardnet (accessed on Jan 26, 2021)
[12] In the context of deep neural network training, an epoch refers to one cycle through the training dataset.

explained in figure 5.9. As can be seen in figure that when $\lambda_{wm}$ equals 5, the best results are achieved on average over different cases (easy, hard and tough). Consequently, $\lambda_{wm} = 5.0$ is selected as the final choice of $\lambda_{wm}$ in this study. All the WeMNet in following experimental tasks relies on the WeMNet with $\lambda_{wm} = 5.0$. Also, the results for $\lambda_{wm} = 3.0$ and 5.0 are very similar, which means that in the corresponding range the trained descriptor is less sensitive to the parameter change. When the model trained by applying $\lambda_{wm} = 5.0$ is compared to the baseline (without the weak match branch and thus $\lambda_{wm} = 0$), a 12.0% overall improvement is gained in terms of mean precision by using the weak match. For the “Hard” and “Tough” sets, 13.2% and 28.1% improvements are achieved, respectively, whereas the improvements for "Easy" set are only marginal (2.5%). This confirms that the involvement of weak matches improves the invariance of descriptors against geometric transformations.

| Descriptor architecture | Training dataset | $\lambda_{wm}$ | Hpatches-Set | Easy | Hard | Tough | Mean |
|---|---|---|---|---|---|---|---|
| HardNet | Liberty | 0 | View | 0.709 | 0.541 | 0.354 | 0.535 |
| | | | Illumination | 0.631 | 0.470 | 0.307 | 0.469 |
| | | | Full | 0.671 | 0.506 | 0.331 | 0.503 |
| HardNet | Liberty | 0.01 | View | 0.709 | 0.544 | 0.359 | 0.537 |
| | | | Illumination | 0.633 | 0.473 | 0.310 | 0.472 |
| | | | Full | 0.672 | 0.510 | 0.335 | 0.505 |
| HardNet | Liberty | 0.1 | View | 0.719 | 0.568 | 0.387 | 0.557 |
| | | | Illumination | 0.633 | 0.489 | 0.331 | 0.484 |
| | | | Full | 0.676 | 0.529 | 0.360 | 0.521 |
| HardNet | Liberty | 1.0 | View | 0.734 | 0.614 | 0.450 | 0.599 |
| | | | Illumination | 0.627 | 0.510 | 0.371 | 0.503 |
| | | | Full | 0.681 | 0.563 | 0.411 | 0.552 |
| HardNet | Liberty | 3.0 | View | 0.747 | 0.623 | 0.455 | 0.609 |
| | | | Illumination | 0.639 | **0.516** | 0.371 | **0.509** |
| | | | Full | 0.694 | 0.571 | 0.414 | 0.560 |
| HardNet | Liberty | 5.0 | View | 0.745 | **0.629** | **0.468** | **0.614** |
| | | | Illumination | 0.630 | 0.515 | **0.379** | 0.508 |
| | | | Full | 0.688 | **0.573** | **0.424** | **0.562** |
| HardNet | Liberty | 10.0 | View | **0.759** | 0.612 | 0.423 | 0.599 |
| | | | Illumination | **0.651** | 0.505 | 0.346 | 0.501 |
| | | | Full | **0.706** | 0.560 | 0.386 | 0.551 |

Table 5.4.: The mean Average Precision (mean AP) of patch based descriptor matching (Task A) for different weights controlling the importance of weak match loss $L_{weak_match}$ in descriptor learning. $\lambda_{wm}$ stands for the weight of the weak match branch. The best value for each case is in bold. HardNet [Mishkin et al., 2017] is employed as the descriptor architecture.

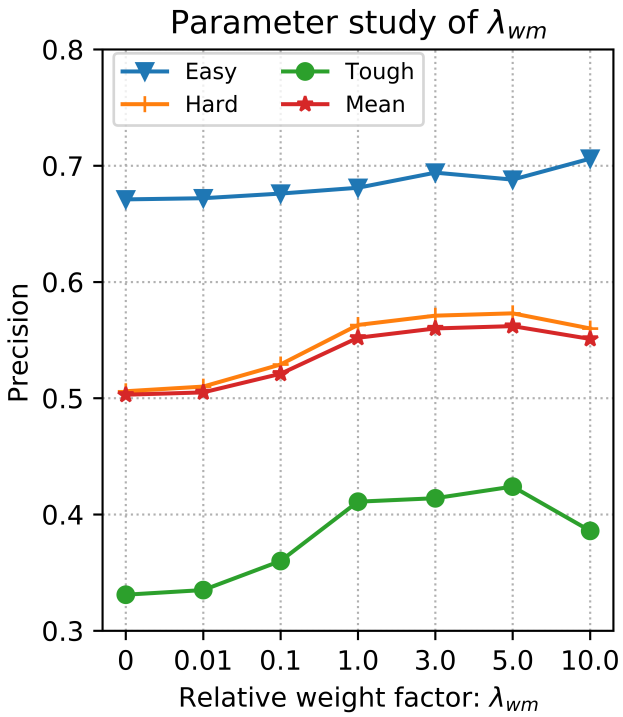



Figure 5.9.: Parameter study for the relative importance of weak match network in descriptor learning. The results are based on the Hpatches full dataset, i.e., the third row of each test case in table5.4.

### 5.3.2. Comparison to Related Work

To evaluate the performance of descriptors, the Hpatches benchmark is used also in this section and the closely related state-of-the-art CNN descriptors HardNet [Mishkin et al., 2017], HardNetPS [Mitra et al., 2018] and SoSNet [Tian et al., 2019] are selected for comparison.

The basic architecture of the descriptor network for all of the selected descriptors is based on L2-Net [Tian et al., 2017]. The necessary information on the selected comparisons is listed in the table 5.5. Among those methods, HardNetPS [Mitra et al., 2018] uses a considerably larger training dataset [13], in which more challenging viewpoint and viewing direction changes are simulated and only the matched pairs with large intersection angles are selected for descriptor learning. For SoSNet and HardNet, the weights trained and published by their authors are used.

The result of the comparison between different configurations is provided in table 5.6. As can be seen, the comparison refers to different sets with different levels of transformation (easy, hard and tough). Therefore, it is easier to differentiate the improvements of the proposed descriptor for different cases. A graphical

[13] the PS dataset is accessible via https://github.com/rmitra/PS-Dataset (accessed on December 27, 2020). Among the 30 subsets contained in the PS dataset, 25 of them are used as training sets and the remaining 5 are used for validation purposes.

visualization is provided in figure 5.10.

| Method | Descriptor Variants | Training Dataset | Self Trained? |
|---|---|---|---|
| HardNet [Mishkin et al., 2017] | *HardNet_Lib* | Liberty | no |
| | *HardNet_Brown6* | All Brown Sets | no |
| HardNetPS [Mitra et al., 2018] | *HardNet_PS* | PS Dataset [Mitra et al., 2018] | no |
| SoSNet [Tian et al., 2019] | *SoSNet_Lib* | Liberty | no |
| WeMNet (this thesis) | *WeMNet_Lib* | Liberty | yes |
| | *WeMNet_Brown6* | All Brown Sets | yes |

Table 5.5.: The selected related state-of-the-art learned CNN descriptors for comparison. All the descriptor variants are based on L2-Net [Tian et al., 2017] architecture. *SoSNet_Brown6* is not available as the results gained with the full Brown Dataset are not published. "Self Trained?" means whether the involved network is trained by the author of this thesis.

As can be seen from table 5.6, WeMNet performs better than the other two state-of-the-art methods (HardNet and SOSNet) in all cases when different sets of images are used for evaluation, in terms of the mean AP. For the easy transformation level, the performance difference between the different methods is only marginal. In the case of illumination change, WeMNet performs slightly worse than the other two methods. Compared to the easy case, the improvements of WeMNet become increasingly visible. For the tough case, the improvement caused by WeMNet is more significant.

As shown in figure 5.10 (set:Illumination), WeMNet performs on a level comparable to HardNet and SoSNet. However, for the viewpoint change (c.f. figure 5.10 (set:View)), the improvement achieved by WeMNet is noticeable. This confirms that the involvement of the weak match branch contributes to a higher invariance against viewpoint change for WeMNet descriptors. The comparison between WeMNet and HardNet_PS confirms, not surprisingly, that the involvement of larger and more challenging viewpoint and viewing direction changes in the patch dataset can result in a better descriptor.

| Descriptor Variants | Hpatches-Set | Easy | Hard | Tough | mean |
|---|---|---|---|---|---|
| *HardNet_Lib* | "a" | 0.684 | 0.508 | 0.328 | 0.507 |
| | View | 0.729 | 0.564 | 0.375 | 0.556 |
| | Illumination | 0.663 | 0.501 | 0.331 | 0.498 |
| | Full | 0.697 | 0.533 | 0.353 | 0.528 |
| *HardNet_Brown6* | "a" | 0.705 | 0.533 | 0.353 | 0.530 |
| | View | 0.748 | 0.587 | 0.397 | 0.577 |
| | Illumination | **0.677** | 0.518 | 0.349 | 0.515 |
| | Full | 0.713 | 0.553 | 0.374 | 0.547 |
| *HardNet_PS* | "a" | 0.693 | **0.585** | **0.446** | **0.574** |
| | View | **0.793** | **0.690** | **0.540** | **0.674** |
| | Illumination | 0.586 | 0.492 | **0.379** | 0.486 |
| | Full | 0.691 | **0.593** | **0.460** | **0.582** |
| *SoSNet_Lib* | "a" | 0.690 | 0.519 | 0.339 | 0.516 |
| | View | 0.736 | 0.577 | 0.388 | 0.567 |
| | Illumination | 0.668 | 0.511 | 0.341 | 0.507 |
| | Full | 0.703 | 0.545 | 0.365 | 0.537 |
| *WeMNet_Lib* | "a" | 0.690 | 0.560 | 0.404 | 0.551 |
| | View | 0.745 | 0.629 | 0.468 | 0.614 |
| | Illumination | 0.630 | 0.515 | **0.379** | 0.508 |
| | Full | 0.688 | 0.573 | 0.424 | 0.562 |
| *WeMNet_Brown6* | "a" | **0.718** | 0.570 | 0.397 | 0.562 |
| | View | 0.771 | 0.637 | 0.455 | 0.621 |
| | Illumination | 0.659 | **0.525** | 0.371 | **0.518** |
| | Full | **0.716** | **0.582** | 0.414 | 0.570 |

Table 5.6.: Average precision of the proposed method and related methods. The bold numbers indicate the best result for each case.

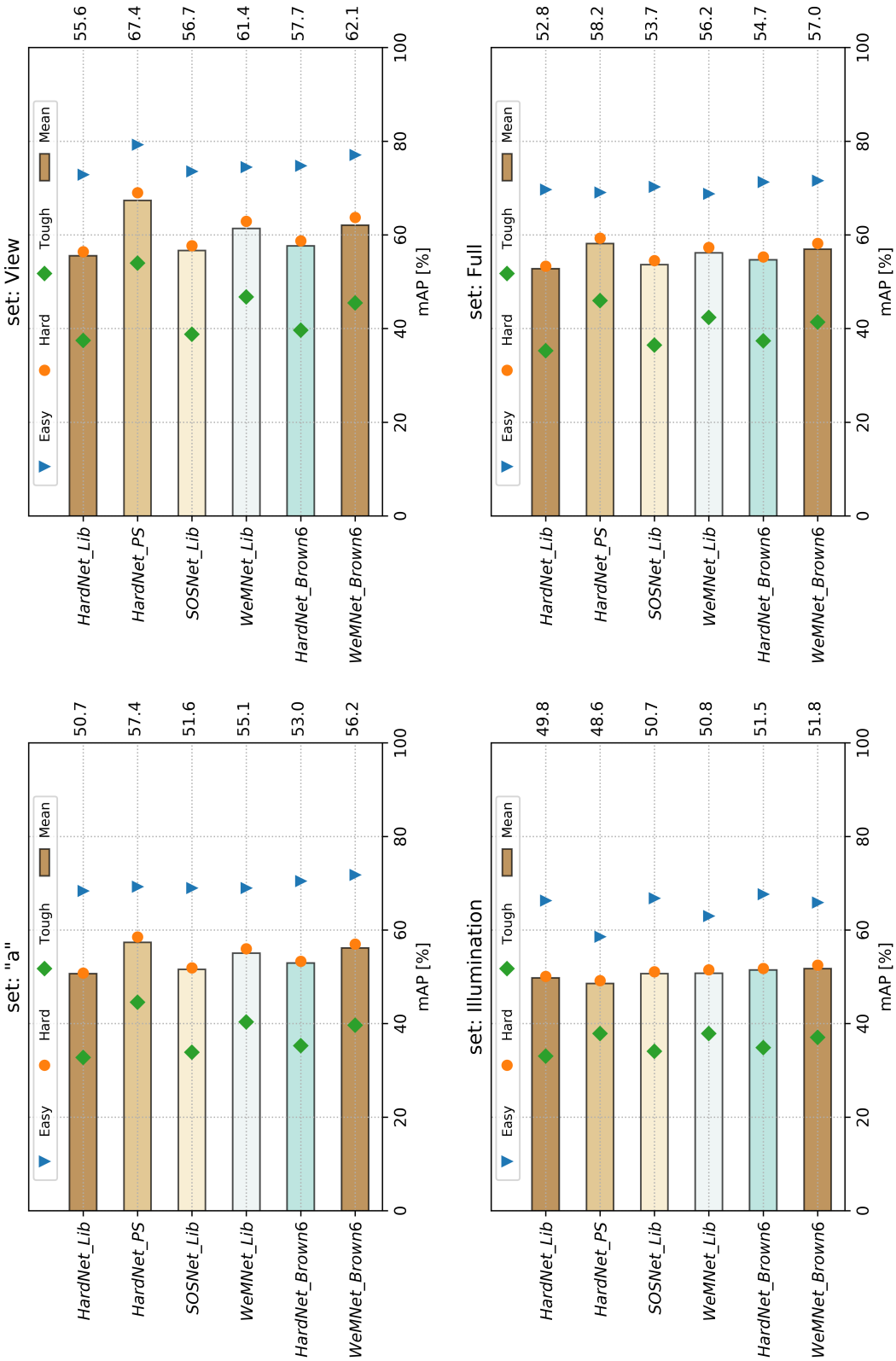



Figure 5.10.: The performance of different descriptor variants using the Hpatches dataset with different set configuration. From left to right and top to bottom, the sets "a", view, illumination and full (as indicated by the title of each figure) are used. It is worth note that for better checking the influence of using different datasets, the descriptor variant using Liberty dataset is shown in the first, third and fourth bars of each sub-figure, while the two descriptors all Brown dataset is placed in the bottom two bars of each sub-figure.

## 5.4. Descriptor Distance Analysis

Considering pairs of matched features, how does the descriptor distance change when different types of transformations are applied? To answer this question, different types of descriptors are applied within different domains of dataset in this experiment. Concretely, the descriptors involved in this study are SIFT[Lowe, 1999], HardNet[Mishkin et al., 2017] and the proposed WeMNet, and the datasets used are NotreDame from the Brown dataset (close range terrestrial images) and Aerial-Dortmund (aerial images).

Among possible types of transformations, translation, rotation and affine shape transformation are selected for analysis, due to the fact that those transformations are in line with the geometric transformations in feature based matching. For each of the experiments introduced below, the used patch dataset contains 100,000 matched pairs.

The details of the three transformations are as follows:

- *Translation*: translate $p$ with vector $\delta_x, \delta_y$ in range of $[-1.5\lambda, 1.5\lambda]$ with a step size of $0.375\lambda$, where $\lambda$ stands for the detected scale of the feature[14].
- *Rotation*: rotate $p$ in range of $[-180°, 180°)$ with a step size of $5°$.
- *Affine shape*: transform $p$ according to the transformation model in equation 4.7. The longitude $\phi$ is a transformation parameter in range of $[0, 180°]$ and the stretch $t$ in range of $[1, 4]$.

In order to compare the descriptor distance of patch pairs, two different transformation rules are used for the *Rotation* and *Affine shape* transformations:

1. Fix the anchor patch ($a$) and apply transformation to the positive patch ($p$)
2. Apply the same transformation to the anchor ($a$) and the positive patch ($p$).

For *Translation*, only the first case is studied. The first case provides a chance to inspect how the descriptor distance changes in comparison to a reference patch. With the second case one can inspect how the descriptor distance is influenced by different common transformations.

### 5.4.1. Translation

How does the descriptor distance change if the features being described undergo small amounts of translation? This experiment uses a small number of steps for translation. The result is shown in figure 5.11. The two axes in the horizontal plane represent the translation in $x$ and $y$ direction ($\triangle_x$ and $\triangle_y$) in the image plane. As explained, the unit 1 in this experiment represents for $0.375\lambda$, where $\lambda$ is the scale of the detected feature. The vertical axis stands for the descriptor distance between the descriptor calculated on the reference patch

[14] The patches of size $32 \times 32$ pixels (corresponding to $12\lambda$) are used; each pixel is equal to a width of $0.375\lambda$

and that on the patch being translated. The result figure reveals that differences between the results for each descriptor using different datasets (NotreDame and Aerial-Dortumund) are small.

As the resulting figures show, the three different descriptor variants all share the same trend, i.e., the descriptor distance increases when the relative translation between the reference and translated patch increases. However, the descriptor distance computed using the SIFT descriptor is distinctively lower than the distance computed using HardNet and WeMNet. The distance computed using WeMNet is slightly lower than the one obtained by using HardNet, as can be detected by checking the colour of contour maps shown at the bottom of the figure.

When the local feature patches are moved from central (low translation) to larger steps, the descriptor distance varies significantly. For instance, as shown in figure 5.11b, this distance changes from 0.6 (low translation) to 1.1 ($1.5\lambda$ translation). This suggests that the descriptor is only invariant against translation in a relatively limited range, e.g., $\sqrt{\triangle_x^2 + \triangle_y^2} < \lambda$ (equivalent to 2.67 pixels in the patch).

### 5.4.2. Rotation

With regard to rotation, similar questions apply: How does the descriptor distance for matched feature pairs change, when one of them is rotated? How does that distance change if the feature pairs are aligned and rotated by the same angles? Two cases of rotation are presented in this section. First, the anchor patch ($a$) is fixed and only the positive patch ($p$) is rotated (FixA_RotP), and the distance between the descriptors of the reference patch and the rotated patch is computed. Second, both the anchor patch and the positive patch are rotated by the same angle (RotA_RotP), then the corresponding descriptor distance is calculated for both of the rotated features. For both cases, the standard deviation at each rotation angel is also calculated. The results are shown in figure 5.12, in which the horizontal axis of each graph stands for the rotation angles from $-180°$ to $180°$, and the vertical axes stand for the descriptor distance.

As illustrated in figure 5.12, the curves computed by HardNet and WeMNet using different datasets are very similar to each other, while the curves computed by SIFT show some noticeable differences when different datasets are used.

For the first case, FixA_RotP, the descriptor distance change for SIFT, HardNet and WeMNet varies in different ways. For SIFT, the descriptor distance is only sensitive to the rotation change in a relatively narrow range, around $-45°$ to $45°$. In the remaining range, the shape of the curve depends on the used data. In the NotreDame dataset, this distance varies smoothly but for the aerial dataset, more complex variations are observed, as shown in figure 5.12d. For HardNet, the descriptor distance is sensitive in a larger range of rotation angles compared to SIFT, around $-60°$ to $60°$. In the range beyond $-60°$ to $60°$, the descriptor distance turns into an almost constant value. For WeMNet, this descriptor distance is sensitive to the rotation change in the full range of $-180°$ to $180°$. For all three descriptors, the descriptor distances calculated at $\pm 180°$ is almost doubled when it is calculated around $0°$. This indicates that running the feature orientation

to align features before description is indeed a necessary step.

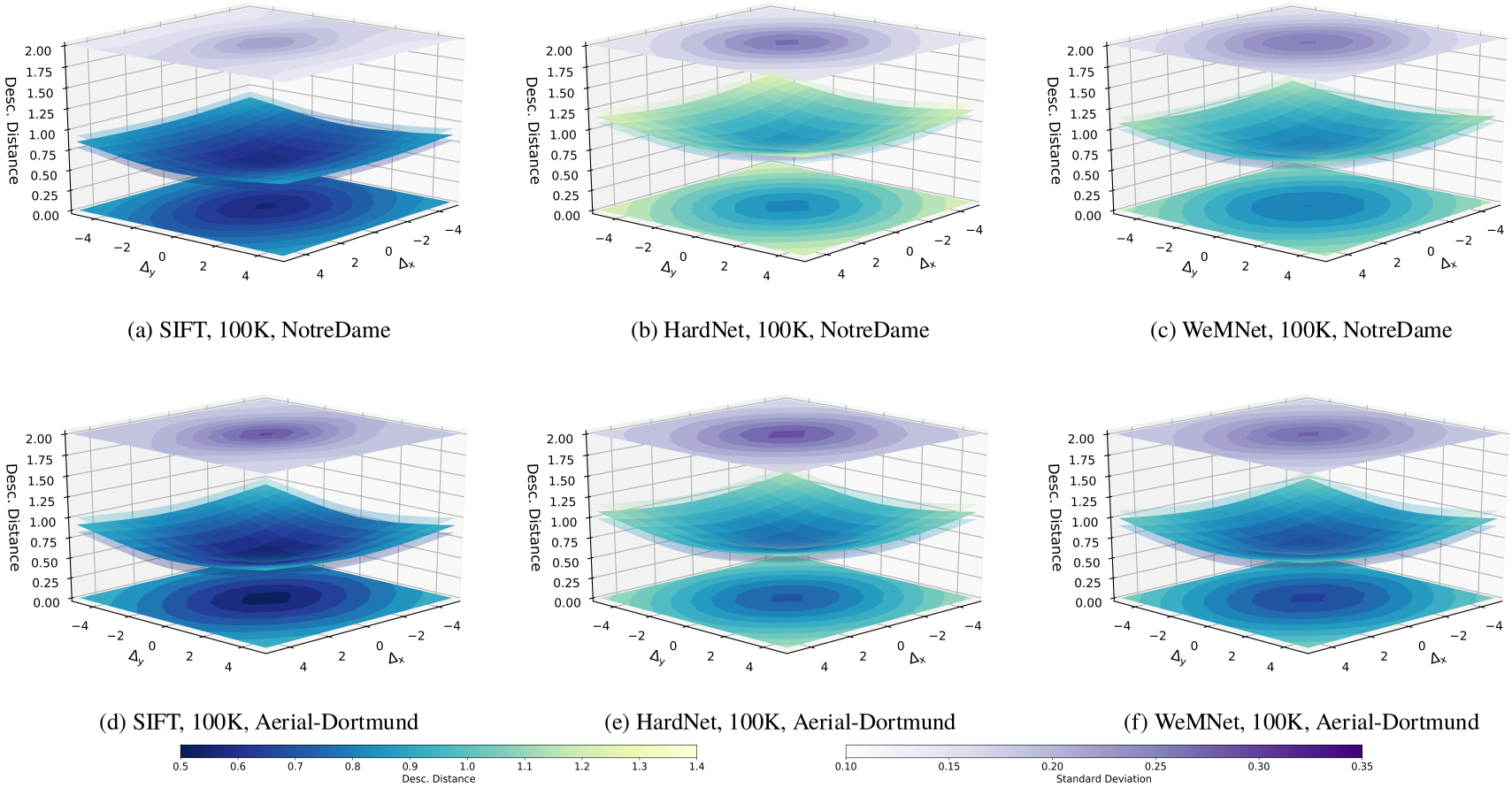
For the second case RotA_RotP, similar results are obtained for SIFT, HardNet and WeMNet. As shown by the green cross marking the line in each sub graph of figure 5.12, the descriptor distance is stable throughout different rotation angles. This is not surprising, as in this case the patches are always aligned, and no distinct descriptor distance change should be expected in any case.

### 5.4.3. Affine Shape Transformation

In this chapter, the descriptor distance change under affine shape transformations is assessed. Similar to the section on rotation, two different cases are explored. First, the anchor patch is fixed and the affine shape transformation is only applied to the positive patch (FixA_AffP). Second, both the anchor and the positive patch undergo the same amount of affine shape transformation (AffA_AffP). The result is shown in Figure 5.13, in which the horizontal axes are the stretch and the longitude for controlling the affine shape transformation (see section 4.3.1) and the vertical axis stands for the descriptor distance calculated with the underlying transformation. Note that the whole range of longitude has been covered in the simulation; for stretch, the simulation range of 1 to 4 is already reasonably large.

For the case FixA_AffP, the contour maps shown in the bottom of each graph in figure 5.13 are quite different. For SIFT, the descriptor distance change in the direction of longitude is considerably larger than in the direction of stretch. This change difference between the stretch and longitude directions is alleviated for HardNet, but an obvious difference along the two directions is still observed (see figure 5.13b and 5.13e). In contrast to SIFT and HardNet, WeMNet results in a change of descriptor distance in both longitude and stretch directions with nearly equal amount. However, this equivalence is more pronounced in the results achieved using Aerial-Dortmund than in the results achieved using NotreDame.

For the case AffA_AffP, similar results are obtained for SIFT, HardNet and WeMNet. This is no surprise, as in all cases the patches are aligned because the same level of affine shape distortion is used. However, the distance computed by using SIFT is notably lower than that computed using HardNet and WeMNet.




(a) SIFT, 100K, NotreDame

(b) HardNet, 100K, NotreDame

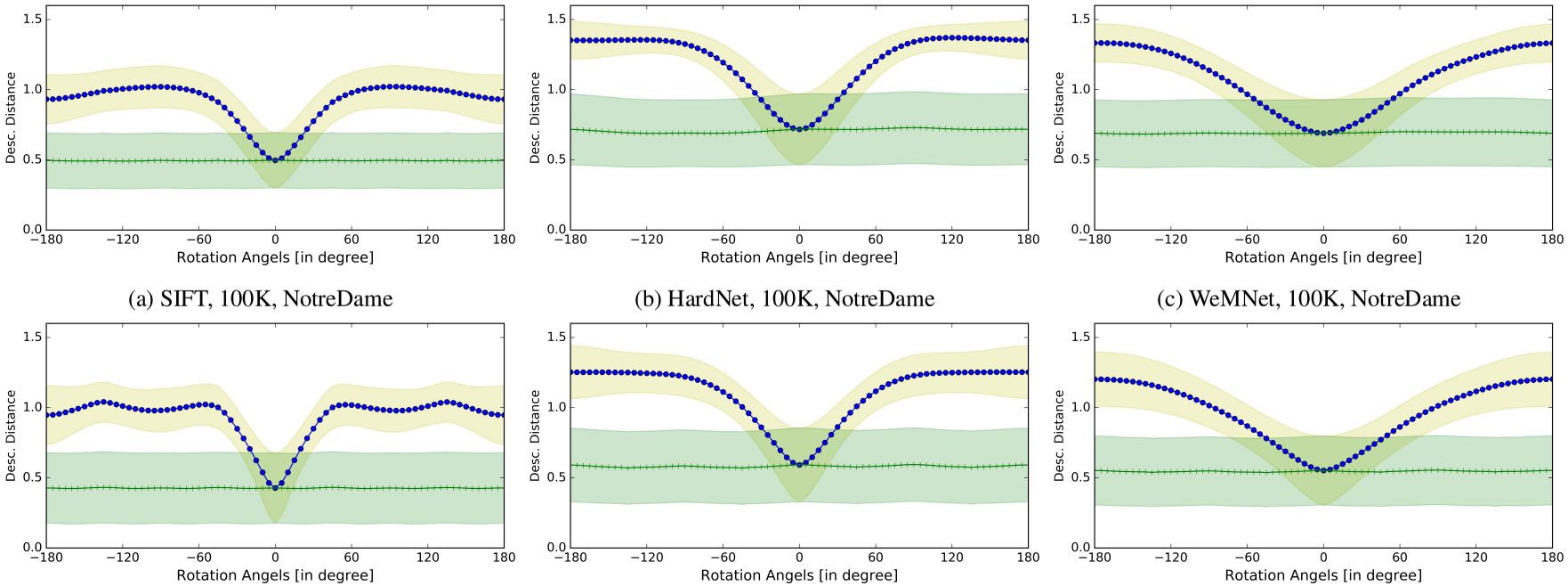
(c) WeMNet, 100K, NotreDame

(d) SIFT, 100K, Aerial-Dortmund

(e) HardNet, 100K, Aerial-Dortmund

(f) WeMNet, 100K, Aerial-Dortmund

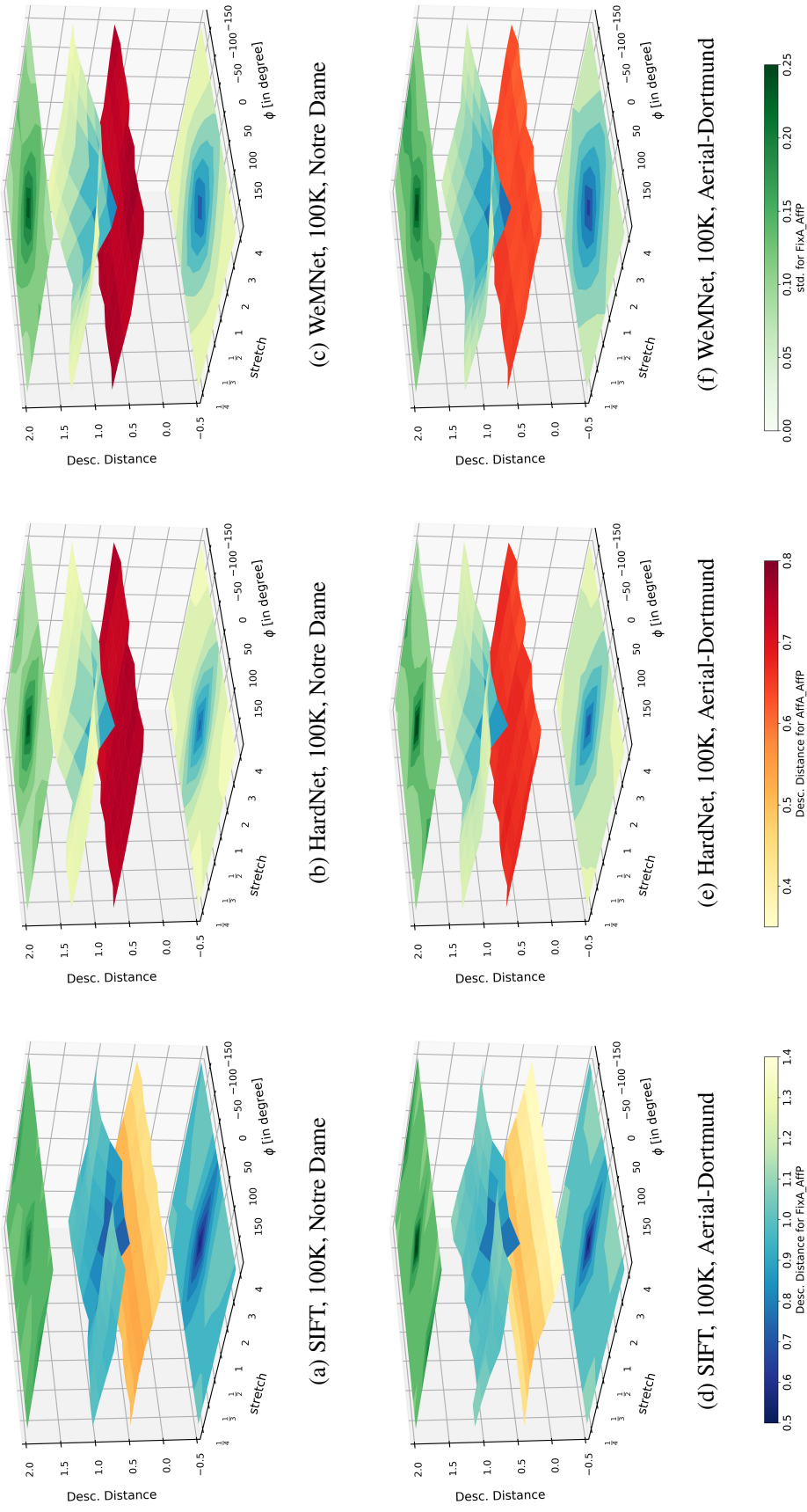
Figure 5.11.: Translation analysis of local features against movements in both x and y direction. The first and second row show the descriptor distance analysis of different datasets, composed of Notre Dame and Aerial-Dortmund. The bottom two colour bars are used for the descriptor distance and standard deviation analysis. In each figure, the horizontal axes indicate the amount of translation in x and y direction of the image plane, and the vertical axes indicate the distance between descriptor pairs. The surface in the middle of each figure shows the descriptor distance change. In addition, the contour map at the top of each figure shows the standard deviation change, whereas the contour map in the bottom of each figure represents the descriptor distance, both encoded with the colour bar shown in the bottom.




(a) SIFT, 100K, NotreDame (b) HardNet, 100K, NotreDame (c) WeMNet, 100K, NotreDame

(d) SIFT, 100K, Aerial-Dortmund (e) HardNet, 100K, Aerial-Dortmund (f) WeMNet, 100K, Aerial-Dortmund

Figure 5.12.: Descriptor distance analysis against a rotation around the centre of the feature support window. The horizontal axis represents the rotation angle from -180 to 180 degrees while the vertical axis stands for the descriptor distance. "FixA_RotP" refers to the case that the anchor patch is fixed and only the positive patches are rotated, whereas "RotA_RotP" is the case in which both anchor and positive patches are rotated by the same angles (i.e., aligned). The range of ± standard deviation for "FixA_RotP" and "RotA_RotP" is indicated by light green and light yellow regions. The first and second row show the descriptor distance analysis conducted with different datasets, composed of Notre Dame and Aerial-Dortmund.




(a) SIFT, 100K, Notre Dame (b) HardNet, 100K, Notre Dame (c) WeMNet, 100K, Notre Dame

(d) SIFT, 100K, Aerial-Dortmund (e) HardNet, 100K, Aerial-Dortmund (f) WeMNet, 100K, Aerial-Dortmund

Figure 5.13.: Descriptor distance analysis against affine shape transformations. The first and second row show the descriptor distance analysis of different datasets, composed of Notre Dame and Aerial-Dortmund. For each figure in the first and second row, the horizontal x axis represents the longitude angle of -180 to 180 degrees and the horizontal y axis stands for stretch. The descriptor distance is indicated by the vertical axis z. The contours at the top and bottom of each figure show the deviation of descriptor distance and the descriptor distance for "FixA_AffP", respectively. In between the descriptor distance change curves for "FixA_AffP" and "AffA_AffP" are shown. The two surfaces are distinguished by the colour bars shown in the bottom of this figure.

## 5.5. Image Matching Analysis

In this section, the learned affine shape, orientation and full affine shape networks are evaluated on image matching benchmarks. This section first explains how the feature affine shape, orientation and full affine shape network are trained. Then the derivation of ground truth correspondence is introduced. The following part studies the parameter $\lambda_{skew}$, which is used to control the relative importance of skew loss in affine shape learning. Finally, the proposed method is compared to other methods using the rotation and affine subset of the Hpatches image sequence benchmark.

**Training of Feature Affine Shape and Orientation Network** The affine shape (MoNet) and orientation (MGNet) modules are trained using mini-batches with a size of 1024. The learning rate policy used here is the same as used in descriptor learning, as explained at the beginning of section 5.3. In total, both MoNet and MGNet are trained in 10 epochs with 12 million patch pairs. For affine shape training (MoNet), the stretch for simulating affine transformation is increased from 4.0 to 5.8 in the later epochs. The details of the change are listed in table 5.7.

| Epoch number | 1 | 2 | 3 | 4 | >=5 |
|---|---|---|---|---|---|
| Stretch used for simulation | 4.0 | 4.5 | 4.8 | 5.3 | 5.8 |

Table 5.7.: The stretch used for training the proposed MoNet.

**Training of Full Affine Shape Network** The training of Full-AffNet also uses mini-batches with a batch size of 1024 and training data size of 12 million patches. Due to the fact that the loss of the full affine shape network is composed of three partial losses and a slower convergence is observed during training, the Full-AffNet is trained with 40 epochs. The learning rate decays in the same way as in descriptor learning. $\lambda_{skew}$ is fixed as 0.001 during training, whereas $\lambda_{ori}$ is adjusted during training, such that the two partial losses can be optimized simultaneously. In this thesis, $\lambda_{ori}$ is set as 0.1 as the initial values which are used for the first five epochs. In the beginning of the sixth epoch, $\lambda_{ori}$ is adjusted to 0.2. Throughout all training epochs, $\lambda_{stretch}$ is fixed as 1. Using this setting, $L_{ori}$ and $L_{strectch}$ are jointly minimized.

To train MoNet, MGNet and Full-AffNet, 15% percent of the training pairs are used for validation purposes. Thus, around 10 million feature patches are used for training. The validation loss is calculated after each epoch is finished and monitored throughout the whole training procedure. Finally, the trained model with the best validation result (i.e., the lowest validation loss) is taken as the final model, to be later used in the tasks C and D.

**Ground Truth Correspondence Derivation:** To determine the ground truth match for a pair of images, each of the detected features in one target image is projected to its reference image using the ground-truth homography. If a feature detected in the reference image is located inside a small range of this projected

feature, i.e. less than $\epsilon_{gt}$, then this pair of features is judged as ground-truth correspondence. The small range mentioned here is tested with multiple choices for the affine transformation set, i.e., with 1, 2 and 3 pixels. For the rotation set, this threshold is fixed at 2 pixels. The main reason of investigating different thresholds in the affine case is that the noise of feature detection for images containing only rotation change is assumed to be lower than that for images including affine transformation. Through a careful check, this method can retrieve most correct correspondences. The ambiguity is low because there usually are no other distracting features in the reference image inside the range of several pixels, due to the fact that non-maximum suppression is applied during the feature detection stage.

Once the ground match correspondences are known, the images in the Hpatches image sequence are matched and AuC for the recall-precision curve, as defined in section 5.2.3, is computed and used as the evaluation criterion.

### 5.5.1. Parameter Study for Affine Shape Learning

The parameter $\lambda_{skew}$ that controls the relative importance of skew loss is tested first. To explore the sensitivity of $\lambda_{skew}$, the matching performance of using different values for $\lambda_{skew}$ is compared. This study employs the Hpatches affine dataset for task C.

Table 5.8 and 5.9 provide results for this study. The descriptors HardNet and WeMNet and the training dataset Aerial-Graz are used for all experiments. $\lambda_{skew}$ is tested with options 0.01, 0.1 and 1.0. The three cases converge and the ground truth matches are derived with different threshold $\epsilon_{gt}$, varying from 1.0 to 3.0 pixels. Training does not convergent when $\lambda_{skew} = 2.0$, in which the stretch is increased to infinity and the output predicted affine corrected pattern collapses to a line.

In each test, the descriptors are matched with ten different descriptor distance thresholds. This descriptor distance threshold is used for descriptor matching and is not to be confused with $\epsilon_{gt}$. This distance threshold gradually changes from a small value to a large one, for each of which recall and precision are computed. Correspondingly, the area under the curve (AuC) for precision-recall is calculated. The precision and recall are first averaged over different descriptor distance thresholds for each image pair and then averaged over all image pairs. As introduced before in the evaluation protocol for Task C, the AuC is averaged over all image pairs.

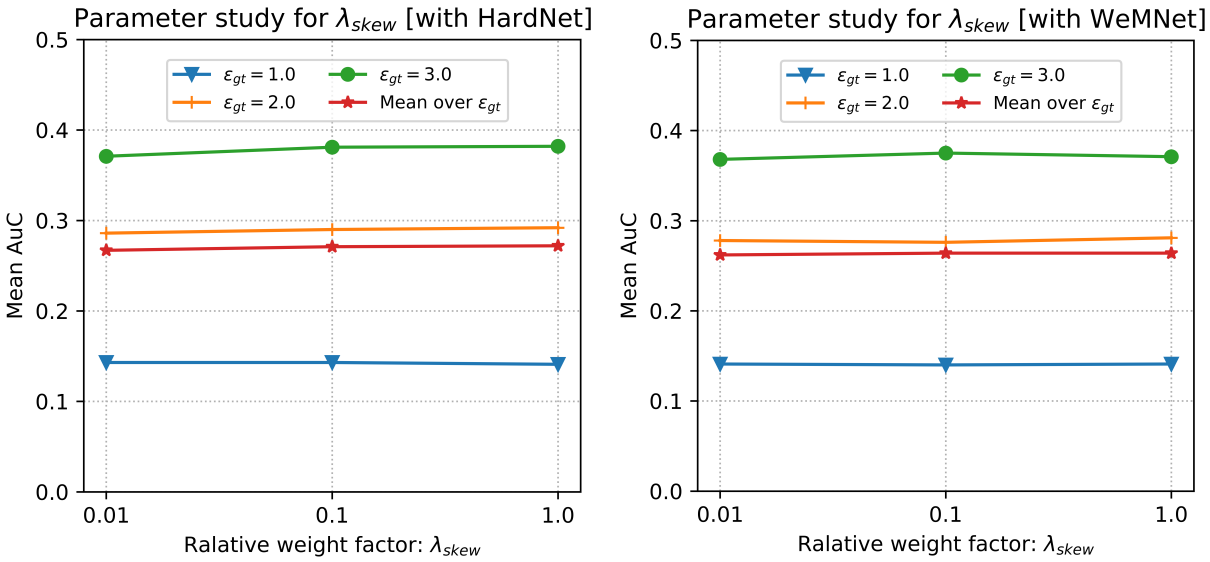
To better view the result of the parameter study, the results shown in table 5.8 and 5.9 are visualized in figure 5.14. In this way, it is easier to see that the mean AuC is not sensitive to the choice of $\lambda_{skew}$. This insensitivity also applies for both HardNet and WeMNet. The difference for different choices of $\epsilon_{gt}$ is extremely small and thus negligible. For simplicity's sake, in the further study, $\lambda_{skew}$ is set to be 1.0 for training MoNet.

| Descriptor | Training Dataset of MoNet | $\lambda_{skew}$ | $\epsilon_{gt}$ | Mean Precision | Mean Recall | Mean AUC |
|---|---|---|---|---|---|---|
| HardNet | Aerial-Graz | 0.01 | 1.0 | 0.149 | 0.403 | 0.143 |
| | | | 2.0 | 0.329 | 0.375 | 0.286 |
| | | | 3.0 | 0.425 | 0.354 | 0.371 |
| HardNet | Aerial-Graz | 0.1 | 1.0 | 0.151 | 0.404 | 0.143 |
| | | | 2.0 | 0.333 | 0.376 | 0.290 |
| | | | 3.0 | 0.434 | 0.357 | 0.381 |
| HardNet | Aerial-Graz | 1.0 | 1.0 | 0.150 | 0.402 | 0.141 |
| | | | 2.0 | 0.331 | 0.379 | 0.292 |
| | | | 3.0 | 0.430 | 0.355 | 0.382 |

Table 5.8.: Parameter study result for $\lambda_{skew}$ using HardNet as descriptor. $\epsilon_{gt}$: threshold for ground-truth correspondences in pixel.

| Descriptor | Training Dataset of MoNet | $\lambda_{skew}$ | $\epsilon_{gt}$ | Mean Precision | Mean Recall | Mean AUC |
|---|---|---|---|---|---|---|
| WeMNet | Aerial-Graz | 0.01 | 1.0 | 0.135 | 0.375 | 0.141 |
| | | | 2.0 | 0.295 | 0.352 | 0.278 |
| | | | 3.0 | 0.383 | 0.330 | 0.368 |
| WeMNet | Aerial-Graz | 0.1 | 1.0 | 0.132 | 0.376 | 0.140 |
| | | | 2.0 | 0.294 | 0.352 | 0.276 |
| | | | 3.0 | 0.394 | 0.333 | 0.375 |
| WeMNet | Aerial-Graz | 1.0 | 1.0 | 0.137 | 0.374 | 0.141 |
| | | | 2.0 | 0.298 | 0.351 | 0.281 |
| | | | 3.0 | 0.388 | 0.330 | 0.371 |

Table 5.9.: Parameter study result for $\lambda_{skew}$ using WeMNet as descriptor. $\epsilon_{gt}$: threshold for ground-truth correspondences in pixel.




(a) parameter study using HardNet as descriptor

(b) parameter study using WeMNet as descriptor

Figure 5.14.: The parameter study for $\lambda_{skew}$. The results are based on the mean AuC in table 5.8 and 5.9, as well as the mean value of three different combinations.

### 5.5.2. Image Matching for Rotation Dataset

In this section, the feature orientation module is evaluated with the Hpatches-Rot, in which the rotation change is the dominate type of transformation between image pairs. Therefore, Hpatches-Rot serves as a suitable test-bed for the performance of feature orientation related networks. The proposed orientation network is compared to some closely related methods.

The different approaches involved in this comparison are summarized in table 5.10. All variants employ the Hessian detector to detect features; different orientation and descriptor modules are combined. The abbreviation of each variant depends on the types of its employed detector, orientation and descripto, as well as the training dataset for the affine shape and orientation related networks. Although all variants rely on the same detector – Hessian detector, "H" is preserved for a better interpretation. The SIFT principal direction assignment strategy [Lowe, 2004] is used and named SIFT in feature orientation. Its combination with the SIFT descriptor, HardNet and WeMNet results in the first three variants listed in table 5.10. OriNet is the network proposed in [Mishkin et al., 2018] for feature orientation. Both MGNet and OriNet are trained on two different datasets, i.e. Brown and Aerial-Graz. For the OriNet trained on the Brown dataset, the weights pre-trained by the author of Mishkin et al. [2018] are used. As there is only rotation change between different images involved in this experiment, affine shape is not estimated. This configuration results in the 8 variants (from AOH-Brown to MW-Graz) listed in table 5.10. Although affine shape and orientation of features are jointly estimated in Full-AffNet and the dataset only contains rotation, Full-AffNet is still tested in this investigation. This leads to the last four variants listed in table 5.10. Figure 5.15 shows the result based on

| Variants | Feature Orientation | Training Data used for Orientation or Full Affine Shape | Descriptor | Notes |
|---|---|---|---|---|
| SS | SIFT | No training | SIFT | Lowe [2004] |
| SH | SIFT | No training | HardNet | |
| SW | SIFT | No training | WeMNet | |
| OH-Brown | OriNet | Brown | HardNet | Orientation used in Mishkin et al. [2018] |
| OH-Graz | OriNet | Aerial-Graz | HardNet | |
| OW-Brown | OriNet | Brown | WeMNet | |
| OW-Graz | OriNet | Aerial-Graz | WeMNet | |
| MH-Brown | MGNet | Brown | HardNet | MGNet for orientation, proposed in this thesis |
| MH-Graz | MGNet | Aerial-Graz | HardNet | |
| MW-Brown | MGNet | Brown | WeMNet | |
| MW-Graz | MGNet | Aerial-Graz | WeMNet | |
| FuH-Brown | Full-AffNet | Brown | HardNet | Full-AffNet for full affine shape estimation, proposed in this thesis |
| FuH-Graz | Full-AffNet | Aerial-Graz | HardNet | |
| FuW-Brown | Full-AffNet | Brown | WeMNet | |
| FuW-Graz | Full-AffNet | Aerial-Graz | WeMNet | |

Table 5.10.: Different combinations of orientation module and descriptors used in the rotation set experiments. All variants use the Hessian feature detector.

the mean AuC. Note that the standard deviations of the computed values, given the 59 groups of images, is relatively high and can amount to as much as 0.5 AuC. Therefore the individual values have to be interpreted cautiously.

With drawing attention to the differences between the different versions of a network (Full-AffNet, MGNet, OriNet) trained with different datasets (Brown or Aerial-Graz), it can be observed from figure 5.15 that the performance difference for MGNet and Full-AffNet are negligible: This high similarity in performance indicates that MGNet and Full-AffNet trained on different datasets performs equally well for rotation invariance. When OriNet is used for feature orientation, the networks trained on different datasets show more noticeable performance difference: OH-Graz is slightly better than OH-Brown, but OW-Graz performs worse than OW-Brown with a more noticeable margin.

In order to compare the two descriptors HardNet ad WeMNet, the other influencing factors must be identical, of course. For all three networks (SIFT orientation as well) and both training datasets, WeMNet performs better than HardNet as larger values for AuC are obtained. This confirms that that WeMNet has better rotation invariance than HardNet

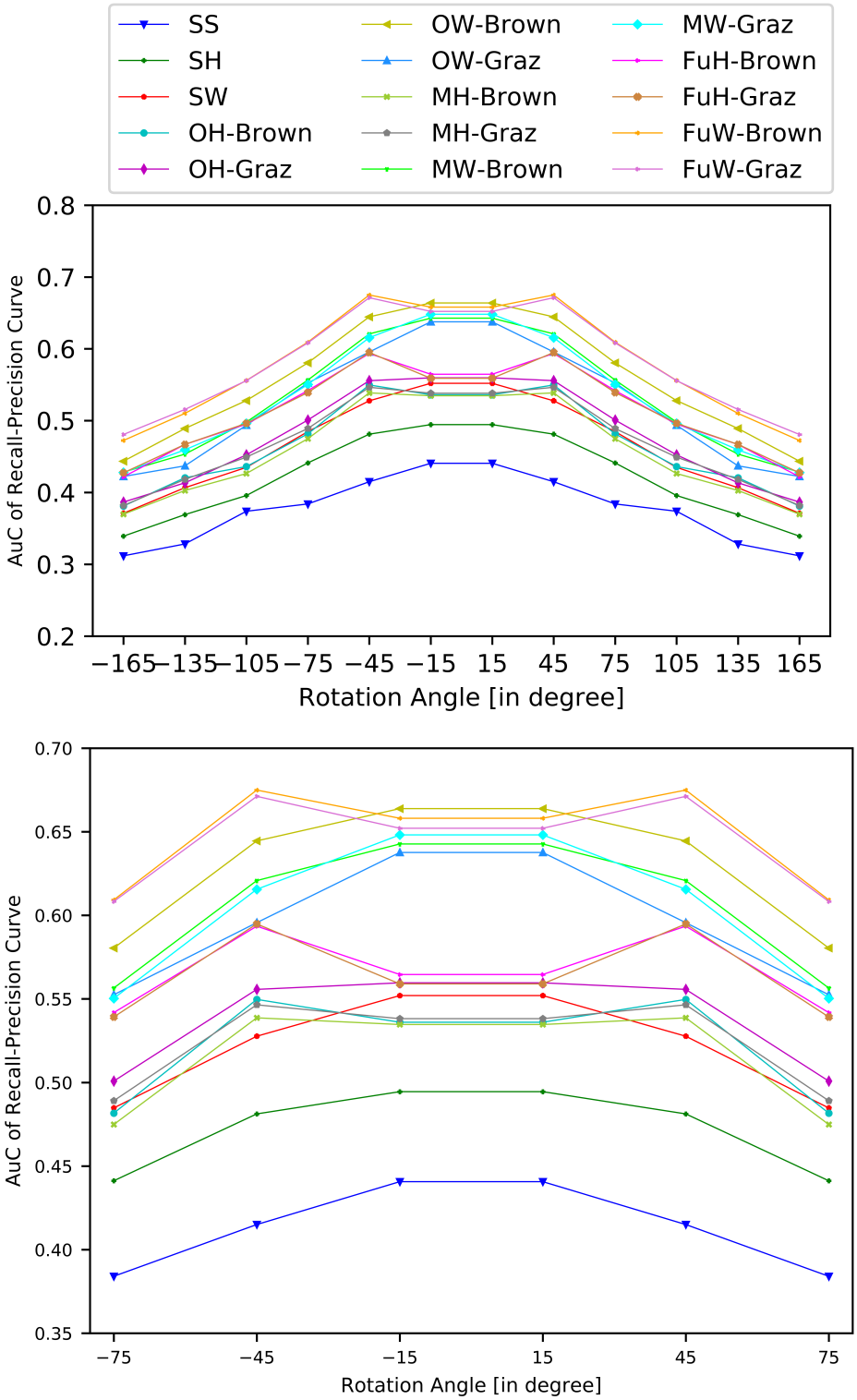



Figure 5.15.: The top graph is the comparison result for the Hpatches rotation dataset; the bottom one is zoomed view of the result in range of -75 to 75 degree.

Besides comparing the performance difference of networks trained with different training datasets and using different descriptors, the performance of Full-AffNet, OriNet and MGNet can be further compared. First, Full-AffNet performs better than OriNet and MGNet constantly. Second, when OriNet and MGNet are compared, the following comparison result are obtained:

- OW-Brown performs better than MW-Brown with a noticeable margin,
- OW-Graz performs slightly worse than MW-Graz,
- OH-Brown performs similar to MH-Brown,
- OH-Graz performs similar to MH-Graz.

based on those comparisons, the MGNet and OriNet show a similar performance in terms of rotation invariance. Third, the learned networks Full-AffNet, OriNet and MGNet performs notably better than the SIFT orientation strategy. This is confirmed by the the comparisons of SH and SW to its competitors using the same descriptor but varying feature orientation modules.

### 5.5.3. Image Matching for Hpatches Affine Dataset

For the image patch affine dataset, different combinations are tested. The criteria for evaluation is again the average AuC, computed in the way explained previously. MoNet and Full-AffNet are the methods proposed in this thesis. For all tested combinations, the Hessian detector is again used. Regarding the affine shape, orientation and feature description modules, combinations are illustrated in table 5.11. The abbreviation of each variant follows the same rule used in the last investigation. All the involved networks for those three modules are trained on both Brown and Aerial-Graz Dataset. AOH-Brown stands for Mishkin et al. [2018], in which the affine shape and orientation module are learned using descriptor distance based loss. The Brown version refers to the published networks trained by the authors of Mishkin et al. [2018], while the Graz version is trained by the author of this thesis using the Aerial-Graz dataset. Among all of the combinations, only BSS is a work with hand-crafted modules. The MoNet, MGNet (variant A of this thesis) and Full-AffNet (variant B of this thesis), as well as the AffNet and OriNet ([Mishkin et al., 2018]) are tested in different combinations with HardNet and WeMNet. In total, 15 different combinations are tested.

All the variants are tested on the Hpatches-Aff dataset, where different thresholds for ground truth matches are used, i.e., 1.0, 2.0 and 3.0 pixels. The results are shown in figure 5.16. Again, the standard deviations are relatively large.

| Variants | Affine Shape | Feature Orientation | Training Data Aff. + Ori. | Descriptor | Notes |
|---|---|---|---|---|---|
| BSS | Baumberg | SIFT | No training | SIFT | Baseline |
| BSH | Baumberg | SIFT | No training | HardNet | |
| BSW | Baumberg | SIFT | No training | WeMNet | |
| AOH-Brown | AffNet | OriNet | Brown | HardNet | Brown version: [Mishkin et al., 2018]; Graz version: trained by the author of this thesis |
| AOH-Graz | AffNet | OriNet | Aerial-Graz | HardNet | |
| AOW-Brown | AffNet | OriNet | Brown | WeMNet | |
| AOW-Graz | AffNet | OriNet | Aerial-Graz | WeMNet | |
| MMH-Brown | MoNet | MGNet | Brown | HardNet | variant A of this thesis |
| MMH-Graz | MoNet | MGNet | Aerial-Graz | HardNet | |
| MMW-Brown | MoNet | MGNet | Brown | WeMNet | |
| MMW-Graz | MoNet | MGNet | Aerial-Graz | WeMNet | |
| FuH-Brown | FullAffine-Net | | Brown | HardNet | variant B of this thesis |
| FuH-Graz | FullAffine-Net | | Aerial-Graz | HardNet | |
| FuW-Brown | FullAffine-Net | | Brown | WeMNet | |
| FuW-Graz | FullAffine-Net | | Aerial-Graz | WeMNet | |

Table 5.11.: Different combinations of affine shape, orientation modules and descriptors used in the image matching performance on Hpatches-Affine dataset. All variants use the Hessian feature detector. Training Data Aff. + Ori. : Training data used for feature affine shape and orientation modules.

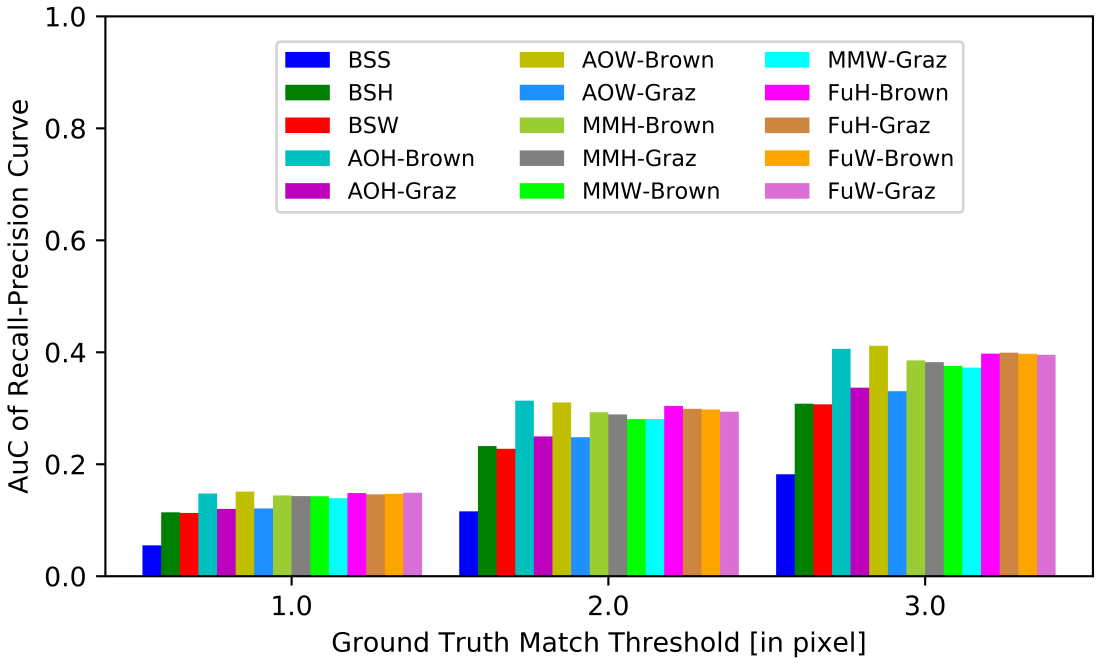



Figure 5.16.: The result for the Hpatches-Aff dataset. The three different groups of bins are computed by using different ground truth match thresholds indicated by the values on the horizontal axis. The generation process of the Hpatches affine dataset is further explained in section 5.1.2.

First, the comparisons for the networks trained by different datasets reveals that MoNet + MGNet and Full-AffNet show a very similar performance. In contrast, the performance of AffNet+OriNet trained on Graz (AOH-Graz, AOW-Graz) is notably worse than the same combination trained on the Brown Dataset (AOH-Brown, AOW-Brown). Second, all the trained networks, except for AffNet and OriNet trained on Graz, performs notably better than the Baumberg [Baumberg, 2000] related variants. Third, the performance of Full-AffNet is comparable to the combination of AffNet and OriNet, when the Brown Dataset is used for training. For the variant A of this thesis, i.e. MoNet + MGNet, a slight performance drop is observed for the Brown dataset, compared to Full-AffNet and AffNet + OriNet. Furthermore, the results of variants using WeMNet and HardNet are very similar.

## 5.6. Image Orientation

Except for evaluating descriptors and other learning modules from the above benchmarks involved in task A, B and C, results of the proposed method used in real applications are also assessed, using an image orientation task as an example. As explained in section 5.1.2, four blocks of images taken by an aerial penta-camera system including nadir and oblique images with significant change in viewing direction and viewpoints are used as input dataset for this assessment. After feature detection and matching, a bundle adjustment is carried out to determine he exterior image orientation parameters. A number of quality measures for the result are selected as evaluation criteria (see below for details).

This section first describes the steps for determining the orientation of different image blocks and all relevant details for the experimental setup, then the image orientation results obtained with different variants proposed in this thesis are presented. In the next step, the respective matching quality is analysed.

### 5.6.1. Determination of Image Orientation

Each small block of images is processed according to the pipeline shown in 5.17 and orientation results after bundle adjustment are obtained. For each image block, the processing steps are explained below.

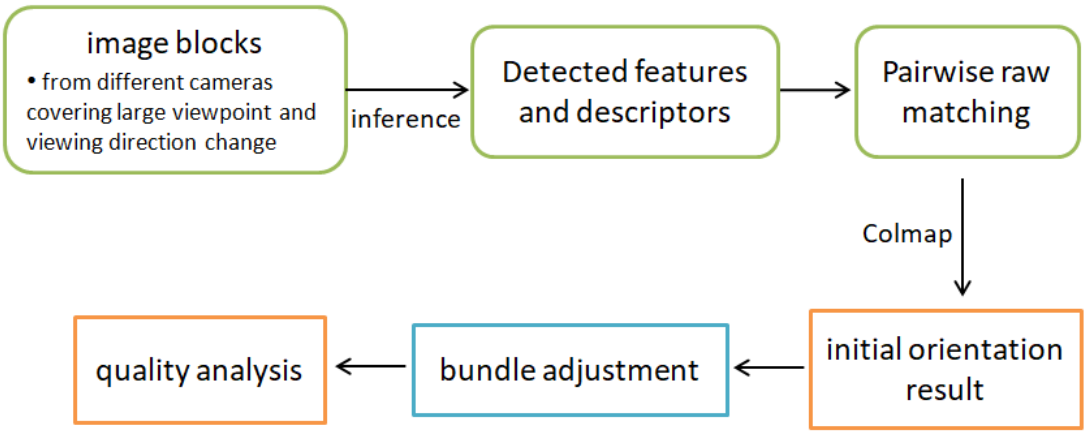



Figure 5.17.: Overview of the the experiment used to determine the orientation of small image blocks.

1. Detect features for all images with the Hessian detector, as explained in section 4.6. In this step, a fixed number of Hessian features with potentially high repeatability against viewpoint changes is detected in scale space. 12000 features per image in all four blocks are extracted[15].

2. Estimate the affine shape as well as the orientation of local features and obtain the descriptors using the different networks, as explained in section 4.6.

3. Apply nearest neighbour ratio matching [Lowe, 2004] to obtain initial matches for each possible pair of images in a block. The ratio between nearest and second nearest matching features is set to 0.85. The output of this step are matching relationship files which contain the initial matches.

4. Providing the initial matches as input, run the Structure from Motion (SfM) software COLMAP[16] [Schönberger and Frahm, 2016] to obtain initial orientation parameters. Before being fed into the SfM pipeline, the matches are geometrically verified by estimating the epipolar geometry.

5. Transform the initial orientation results into a common coordinate system for further comparison[17]. To achieve this goal, one image is selected as origin of the common coordinate system, setting both its projection centre and rotation angles to zero. A second image is selected to define the scale between the two projection centres. For all employed blocks, the scale is computed from the provided projection centre coordinates, which are measured by a GPS-IMU (Inertial Measurement Unit) system during flight. According to Nex et al. [2015] the accuracy of the projection centres is approximately 1m. The length of the baseline between the two images is in a range between 500 and 600 metres. The selected first and second image are identical for each image block processed by the different variants.

[15] Due to the fact that the image size is relatively large, features in each image are extracted on four separate non-overlapped tiles of the original image and are subsequently combined. Since this does not affect the result, it is a way to parallelize the whole algorithm.

[16] https://github.com/colmap/colmap

[17] As matches obtained from different pipelines vary, so do the orientation parameters, especially when different image pairs are used as initial pairs in SfM. Additionally, not all images can be registered with all pipelines.

6. Run robust bundle adjustment. The bundle adjustment delivers the final orientation parameters. Robust estimation is used and observations with residuals larger than 3.2 times the standard deviation of the image coordinates are considered outliers, and are excluded from further iterations. According to the camera calibration protocol, the camera distortion effect is marginal and the distortion parameters are therefore neglected.

### 5.6.2. Experiment Setup Details

In this experiment, the proposed methods are compared to other closely related works. The selected combinations of feature detector, affine shape estimation, orientation and description module involved in this experiment are identical to the one listed in table 5.11. As mentioned already, in total, 15 combinations are tested.

As the distribution of descriptor distance between matched features generated by different descriptor networks varies, it is difficult to set a common value for the nearest neighbour distance based matching strategy. Instead, the nearest neighbour ratio, as defined in section 2.1.7, is utilized in the current task D. For all the experiments, the distance ratio is set as 0.85. This setting of 0.85 is slightly larger than the typical value suggested for hand-crafted descriptors. On the other hand, the learning based descriptors optimize the distance directly, instead of minimizing the ratio between the distance to the nearest and second nearest neighbour. Therefore, 0.85 is chosen as a well-balanced value for all variants of methods.

### 5.6.3. Orientation Result of Different Blocks

This experiment uses the four different blocks of images presented in section 5.1.2. Table 5.12 summarizes the result for block 1 and 2, while table 5.13 summarizes the result for block 3 and 4. In these two tables, the following items are listed (see also section 5.2.4):

- number of registered images ($N_{img_reg.}$)
- number of reconstructed 3D points ($N_{3DP_rec.}$)
- the estimated standard deviation of image coordinates after bundle adjustment $\sigma_0^{post}$ (in pixel)
- the mean number of matching points per image
- the precision (in meter) of object coordinates in X, Y, Z direction

As can be seen from both tables 5.12 and 5.13, the hand-crafted variant BSS can only reconstruct part of the images in all the four blocks. The number (and ratio) of reconstructed images using BSS varies from block 1 to block 4. For block 1 fewer images are reconstructed, possibly due to some areas with poor texture.For block 2, more images are reconstructed. Here, the difficulty of matching is decreased, as the scene contains

more texture. $N_{img_reg.}$ for block 3 and 4 lies in the middle, probably owing to the fact that the images in those two blocks contains more occlusions due to high buildings, although the images are well textured.

For AffNet and OriNet trained on the Aerial-Graz dataset, the related two variants are AOH-Graz and AOW-Graz. With those two variants, none of the 4 blocks can be reconstructed completely in terms of $N_{img_reg.}$. This observation is consistent with the observations in task C, in which AOH-Graz and AOW-Graz have shown a significant performance drop in terms of invariance against rotation and affine shape transformations.

Using learned descriptors all four blocks can be completely reconstructed in terms of $N_{img_reg.}$, as seen from the results of BSH and BSW. Correspondingly, the number of reconstructed 3D points increases dramatically from BSS to BSH and BSW. Also, the precision of object point coordinates improves considerably, as shown in the result for blocks 1, 3 and 4. When comparing $N_{3DP_rec.}$ for all variants using WemNet and HardNet as descriptor (e.g., AOH-Brown vs. AOW-Brown, MMH-Brown vs. MMW-Brown ), it is found that in most but not in all cases WeMNet performs better than HardNet.

For the combination of MoNet and MGNet (variant A), the difference between the two versions trained on Brown and Aerial-Graz datasets shows some dependency to the choice of employed descriptor. When WeMNet is employed as feature descriptor, then for all four groups the version trained on Brown, i.e. MMW-Brown, performs better than the same networks trained on Aerial-Graz (MMW-Graz). However, when HardNet is employed as feature descriptor, the performance difference between MMH-Brown and MMH-Graz is more dataset dependent: for block 1 and 4, MMH-Brown performs worse than MMH-Graz; for block3 MMH-Brown performs better than MMH-Graz, while for block 2 the two variants perform similar.

For variant B, i.e. Full-AffNet, the two versions trained on Brown and Aerial-Graz are very comparable to each other. In all the four blocks, the difference of $N_{3DP_rec.}$ between the following pairs of variants:

- FuH-Brown and FuH-Graz
- FuW-Brown and FuW-Graz

is small.

| Block Index | Det+Aff.+ Ori.+Desc. | # reg. img | # rec. pts | $\sigma_0^{post}$ | #m. matches. per img. | precision of X,Y,Z [m] |
|---|---|---|---|---|---|---|
| Block-1 | BSS | 7/18 | 849 | 0.76 | 283.4 | 0.205, 0.375, 0.593 |
| | BSH | 18/18 | 3124 | 0.78 | 557.9 | 0.352, 0.170, 0.367 |
| | BSW | 18/18 | 3233 | 0.85 | 591.1 | 0.375, 0.156, 0.373 |
| | AOH-Brown | 18/18 | 4695 | 0.76 | 882.7 | 0.312, 0.145, 0.305 |
| | AOH-Graz | 11/18 | 1624 | 0.80 | 476.0 | 0.080, 0.234, 0.261 |
| | AOW-Brown | 18/18 | **5260** | 0.83 | 1012.7 | 0.332, 0.149, 0.321 |
| | AOW-Graz | 12/18 | 1549 | 0.84 | 426.1 | 0.090, 0.249, 0.291 |
| | MMH-Brown | 18/18 | 4293 | 0.76 | 800.3 | 0.330, 0.146, 0.330 |
| | MMH-Graz | 18/18 | **4968** | 0.75 | 741.1 | 0.335, 0.142, 0.328 |
| | MMW-Brown | 18/18 | 4455 | 0.80 | 826.4 | 0.350, 0.159, 0.356 |
| | MMW-Graz | 18/18 | 3966 | 0.78 | 735.3 | 0.366, 0.161, 0.404 |
| | FuH-Brown | 18/18 | 4252 | 0.74 | 791.0 | 0.322, 0.141, 0.313 |
| | FuH-Graz | 18/18 | 4584 | 0.75 | 845.2 | 0.324, 0.142, 0.320 |
| | FuW-Brown | 18/18 | **4722** | 0.79 | 886.0 | 0.333, 0.140, 0.323 |
| | FuW-Graz | 18/18 | 4716 | 0.80 | 889.8 | 0.338, 0.151, 0.332 |
| Block-2 | BSS | 15/19 | 5255 | 0.64 | 1110.7 | 0.190, 0.276, 0.309 |
| | BSH | 19/19 | 11116 | 0.82 | 2110.2 | 0.243, 0.239, 0.306 |
| | BSW | 19/19 | 11890 | 0.91 | 2284.9 | 0.267, 0.258, 0.335 |
| | AOH-Brown | 19/19 | 14462 | 0.81 | 2904.5 | 0.230, 0.221, 0.287 |
| | AOH-Graz | 12/19 | 7804 | 0.80 | 2366.1 | 0.063, 0.294, 0.277 |
| | AOW-Brown | 19/19 | **15080** | 0.87 | 3059.6 | 0.250, 0.238, 0.312 |
| | AOW-Graz | 12/19 | 8187 | 0.84 | 2521.3 | 0.067, 0.312, 0.293 |
| | MMH-Brown | 19/19 | 13361 | 0.79 | 2617.0 | 0.236, 0.229, 0.295 |
| | MMH-Graz | 19/19 | 13389 | 0.79 | 2618.4 | 0.234, 0.227, 0.293 |
| | MMW-Brown | 19/19 | 13946 | 0.86 | 2780.8 | 0.254, 0.245, 0.316 |
| | MMW-Graz | 19/19 | 14132 | 0.86 | 2809.7 | 0.248, 0.239, 0.311 |
| | FuH-Brown | 19/19 | 14025 | 0.79 | 2759.0 | 0.234, 0.223, 0.290 |
| | FuH-Graz | 19/19 | 13800 | 0.79 | 2712.5 | 0.235, 0.223, 0.291 |
| | FuW-Brown | 19/19 | **14662** | 0.85 | 2925.5 | 0.250, 0.237, 0.311 |
| | FuW-Graz | 19/19 | **14468** | 0.87 | 2896.9 | 0.254, 0.241, 0.314 |

Table 5.12.: The result for all block 1 and 2 after robust bundle adjustment. #reg. img: number of registered images over available number of images; # rec. pts: number of reconstructed 3D points; $\sigma_0^{post}$ posterior standard deviation in pixel; #m. matches. per img.: number of mean matches per image. Precision of X, Y, Z refers to the scaling of the block and is in unit meter. The bold numbers indicate the highest three numbers of reconstructed 3D points.

| Block Index | Det+Aff.+ Ori.+Desc. | # reg. img | # rec. pts | $\sigma_0^{post}$ | #m. matches. per img. | precision of X,Y,Z [m] |
|---|---|---|---|---|---|---|
| Block-3 | BSS | 10/17 | 1346 | 0.62 | 365.2 | 0.126, 0.131, 0.336 |
| | BSH | 17/17 | 4053 | 0.69 | 752.9 | 0.122, 0.127, 0.204 |
| | BSW | 17/17 | 4109 | 0.76 | 791.9 | 0.128, 0.130, 0.209 |
| | AOH-Brown | 17/17 | **5061** | 0.69 | 1040.8 | 0.109, 0.112, 0.175 |
| | AOH-Graz | 10/17 | 3182 | 0.70 | 973.4 | 0.061, 0.139, 0.206 |
| | AOW-Brown | 17/17 | **5517** | 0.73 | 1134.7 | 0.110, 0.124, 0.185 |
| | AOW-Graz | 10/17 | 3361 | 0.74 | 1047.0 | 0.065, 0.147, 0.220 |
| | MMH-Brown | 17/17 | 4493 | 0.68 | 883.8 | 0.116, 0.119, 0.188 |
| | MMH-Graz | 17/17 | 3997 | 0.69 | 782.5 | 0.117, 0.105, 0.190 |
| | MMW-Brown | 17/17 | 4438 | 0.72 | 876.8 | 0.151, 0.118, 0.203 |
| | MMW-Graz | 17/17 | 4394 | 0.73 | 877.0 | 0.117, 0.118, 0.200 |
| | FuH-Brown | 17/17 | 4415 | 0.68 | 875.4 | 0.114, 0.118, 0.185 |
| | FuH-Graz | 17/17 | 4223 | 0.66 | 829.6 | 0.116, 0.120, 0.193 |
| | FuW-Brown | 17/17 | **5137** | 0.71 | 1030.8 | 0.113, 0.121, 0.185 |
| | FuW-Graz | 17/17 | 4976 | 0.71 | 994.2 | 0.113, 0.124, 0.190 |
| Block-4 | BSS | 10/20 | 1098 | 0.70 | 354.0 | 0.104, 0.219, 0.416 |
| | BSH | 20/20 | 5009 | 0.70 | 856.1 | 0.128, 0.137, 0.189 |
| | BSW | 20/20 | 5919 | 0.76 | 1012.9 | 0.134, 0.138, 0.197 |
| | AOH-Brown | 20/20 | **7525** | 0.71 | 1346.5 | 0.114, 0.120, 0.173 |
| | AOH-Graz | 12/20 | 4279 | 0.73 | 1187.3 | 0.052, 0.160, 0.185 |
| | AOW-Brown | 20/20 | **8107** | 0.75 | 1469.0 | 0.120, 0.125, 0.176 |
| | AOW-Graz | 12/20 | 4472 | 0.76 | 1263.8 | 0.054, 0.169, 0.194 |
| | MMH-Brown | 20/20 | 5506 | 0.70 | 967.2 | 0.123, 0.131, 0.200 |
| | MMH-Graz | 20/20 | 6008 | 0.71 | 1052.8 | 0.135, 0.142, 0.241 |
| | MMW-Brown | 20/20 | 6380 | 0.74 | 1130.5 | 0.124, 0.129, 0.184 |
| | MMW-Graz | 20/20 | 6309 | 0.74 | 1125.6 | 0.123, 0.134, 0.188 |
| | FuH-Brown | 20/20 | 6323 | 0.69 | 1133.8 | 0.120, 0.128, 0.172 |
| | FuH-Graz | 20/20 | 6807 | 0.69 | 1189.3 | 0.112, 0.124, 0.177 |
| | FuW-Brown | 20/20 | 6600 | 0.73 | 1201.5 | 0.122, 0.138, 0.181 |
| | FuW-Graz | 20/20 | **6818** | 0.74 | 1226.5 | 0.124, 0.131, 0.183 |

Table 5.13.: The result for all block 3 and 4 after robust bundle adjustment. #reg. img: number of registered images over available number of images; # rec. pts: number of reconstructed 3D points; $\sigma_0^{post}$ posterior standard deviation in pixel; #m. matches. per img.: number of mean matches per image. Precision of X, Y, Z refers to the scaling of the block and is in unit meter. The bold numbers indicate the highest three numbers of reconstructed 3D points.

For the affine shape and orientation module, both separating them into two modules, i.e., AffNet + OriNet or MoNet + MGNet (Variant A), and combining them into one, i.e., Full-AffNet (Variant B), leads to a complete reconstruction, except for the AffNet and OriNet trained on Aerial-Graz. By comparing $N_{3DP_rec.}$ for all blocks, it is found that Full-AffNet performs slightly better than the combination of MoNet and MGNet, whereas the combination of AffNet and OriNet trained on Brown Dataset performs slightly better than Full-AffNet.

The posterior standard deviation, $\sigma_0^{post}$, indicates the size of overall residuals (re-projection error) after bundle adjustment. This number lies in the range of 0.6 to 0.9 pixel for most experiments, owing to the fact that convergent views are included and the number of observations is only in the range of 2 to 3 times the number of unknowns. Also, for two convergent views, the uncertainly of feature matching also increases. Therefore, when more matches from convergent views are included, $\sigma_0^{post}$ will increase. This is observed by comparing all variants with WeMNet and HardNet, e.g., MMW and MMH. The reason that a higher $\sigma_0^{post}$ is obtained for WeMNet variants could be that more matches from convergent views are included. Similarly, this also partially contributes to the fact that variants with learned descriptors (HardNet and WemNet) have a higher $\sigma_0^{post}$ than variants with SIFT descriptors because more matches from convergent views are included using learned descriptors (see figure 5.19).

The precision for X, Y, Z are the average standard deviations of object points in the three dimensions defined in the object coordinate systems after bundle adjustment. The configuration of the blocks, as well as the matching quality, affects this value. For urban image blocks 3 and 4, the deep learning based variants outperform BSS considerably in all three coordinates. In the suburban blocks 1 and 2, more challenging and non-evenly distributed views in the different directions are contained. The uneven distribution of views in block 1 contributes to the fact the precision of object points in X and Y direction for that block is not roughly equal. Corresponding to the aforementioned reason with regard to the use of $\sigma_0^{post}$ for variants with WeMNet and HardNet, object point precision for variants related to WeMNet is slightly worse than for variants using HardNet. The object coordinate precision difference between variants with different orientation and affine shape estimation modules is less distinguishable, which means those variants, i.e., Baumberg+SIFT, AffNet+OriNet, MoNet+MGNet and FullAffine-Net, are comparable in terms of X, Y, Z coordinate precision.

In order to analyse the influence of including oblique views on the 3D object point precision, for each image block only the nadir images are used to determine the image orientation parameters [18]. For this study, only one matching variant is utilized, due to the fact that matching nadir view should be handled by all variants as very limited level of geometric transformations are included for adjacent nadir views. In particular, the MMH-Brown variant is used. The results are listed in table 5.14.

[18] Due to the fact only three or four nadir images are contained in each block, the option of ignoring two view tracks is switched off during SfM, while that option is switched on for the image orientation using the complete blocks.

| Block Index | # reg. img | # rec. pts | $\sigma_0^{post}$ | #m. obs. per img. | precision of X,Y,Z [m] |
|---|---|---|---|---|---|
| Block-1 | 4/4 | 2341 | 0.62 | 1263.3 | 0.114, 0.254, 0.711 |
| Block-2 | 4/4 | 5268 | 0.76 | 3060.5 | 0.094, 0.208, 0.588 |
| Block-3 | 3/3 | 2318 | 0.64 | 1679.7 | 0.121, 0.107, 0.372 |
| Block-4 | 4/4 | 3483 | 0.72 | 1882.8 | 0.112, 0.130, 0.365 |

Table 5.14.: Image orientation result using MMH-Brown as matching variant for all blocks after bundle adjustment. #reg. img: number of registered images over available number of images; # rec. pts: number of reconstructed 3D points; $\sigma_0^{post}$ posterior standard deviation in pixel; #m. obs. per img.: number of mean observations per image. Precision of X, Y, Z refers to the scaling of the block and is in unit meter.

For all blocks, it is clear the main improvement comes from the Z direction. Compared to the object precision in Z direction for the case of only using nadir, a nearly 50% improvements is observed when oblique views are included (see table 5.13). In the X and Y direction, the improvement on precision is significantly less obvious. For block 1 and 2, the precision in X and Y direction is not roughly equal, largely owing to the fact the distribution of the nadir images in block 1 and 2 is uneven in X and Y direction (see figure 5.8).

### 5.6.4. Matching Quality Analysis

The above analysis confirms that the involvement of learned modules for affine shape estimation, orientation and description improves the quality of image orientation for blocks containing convergent views. With the help of those learned modules, the image blocks are reconstructed in a more complete way with more reconstructed 3D points, and more accurate object points can be derived. However, a closer look at the matching quality is still needed and is provided in this section.

In order to better analyse the matching quality of different variants, the following matching quality related terms are analysed:

- number of initial matches ($N_{match}^{initial}$)
- number of inlier matches ($N_{match}^{inl_2view}$) after running two view geometry estimation using RANSAC. Those matches are used as the input matches for SfM.
- number of matches after bundle adjustment ($N_{match}^{surv_BA}$)
- ratio of inlier matches/initial matches ($R_{initial}^{inl_2view}$)
- ratio of matches after bundle adjustment/inlier matches($R_{surv_BA}^{inl_2view}$)

For each variant, the number of initial matches ($N_{match}^{initial}$), matches verified by two-view geometry ($N_{match}^{inl_2view}$) and the number of matches after bundle adjustment ($N_{match}^{inl_2view}$) serve as direct indicators of the matching

performance. Provided that identical feature detectors are used and the same number of high quality features is detected as input, a higher number in all of these three criteria is expected from a better feature matching variant. Moreover, the ratio of inlier matches after running two view geometry estimation using RANSAC, i.e., $R_{initial}^{inl_2view}$, effectively indicates how discriminative a feature based image matching variant is. A higher ratio is normally achieved by a variant with more discriminative power. Compared to $R_{initial}^{inl_2view}$, the ratio of matches after bundle adjustment, i.e., $R_{surv_BA}^{inl_2view}$, depends on both the quality of matches and the detailed techniques used in SfM, e.g., connecting new views or the removal of inconsistent observations. Therefore, $R_{initial}^{inl_2view}$ is taken as a major indicator and $R_{surv_BA}^{inl_2view}$ is used as a side indicator for the overall quality measure in the image orientation task. Table 5.15 summarizes those terms for block 1 and 2, while table 5.16 summarizes those for block 3 and 4.

For a better comparison, the aforementioned numbers and ratios of tables 5.15 and 5.16 are also shown as bar graphs in figure 5.18. $N_{match}^{inl_2view}$ computed by BSS is lower than for other variants (as shown in the second row of figure 5.18), and among those verified matches, a much smaller number is available after bundle adjustment (as shown in the first row of figure 5.18). This indicates that BSS performs worse than the other variants. For the AffNet and OriNet trained on Aerial-Graz (AOH-Graz and AOW-Graz), $N_{match}^{inl_2view}$ is lower than the same term generated by the version trained on Brown (AOH-Brown and AOW-Brown), and this decrease becomes more significant for $N_{match}^{surv_BA}$. This indicates that probably a higher proportion of two-view inlier matches generated by the variants using AffNet and OriNet trained on Aerial-Graz are excluded by the view connecting process of image orientation, compared to other learning based variants. As a result, significantly fewer views are connected and accordingly, less 3D object points are reconstructed by AOH-Graz and AOW-Graz.

When comparing $N_{match}^{inl_2view}$ and $R_{initial}^{inl_2view}$ for each variant for WeMNet and HardNet, it is found that variants with WeMNet constantly yield better results than those with HardNet. This again confirms that WeMNet achieves higher invariance against viewpoint and viewing direction change than HardNet. When taking into consideration the variants for affine shape and orientation (as shown in the second and third row of figure 5.18), the pipelines of AffNet+OriNet, MGNet+MGNet and FullAffine-Net achieve comparable results.

In the next step, the interacting rays at each 3D object point is analysed. A very first result is the distribution of track length (the number of rays per 3D point) for each variant. However, the feature track length cannot distinguish from which views a feature track is generated. For instance, a longer feature track with matches only from one camera view might contribute less to the stability of the image orientation task than a shorter feature track with matches from two or more different camera views, as the image block can be better connected with cross camera view feature tracks, thus stabilizing the reconstructed block. Based on the source camera view of each feature in a feature track, the feature tracks are classified into four different classes, i.e., “only_nadir_or_obl”, “nadir_obl”, “obl_nadir_obl”, “obl_obl”. A detailed explanation of the meaning of each case is given in section 5.2.4. The result of the view intersection is provided in figure 5.19.

As can be seen in figure 5.19, nearly for all variants “only_nadir_or_obl” accounts for the highest proportion.

HBSS shows a higher proportion than other variants in three of the four blocks (block 1, 2 and 3) and has virtually no cases for "obl_nadir_obl" and "obl_obl". This indicates that BSS can only retrieve a very limited number of matches from harder cases. All other machine learning variants retrieve matches in the cases "nadir_obl" and "obl_nadir_obl". For the hardest cases "obl_obl", blocks 1 and 2 have a higher proportion (3-5%) than blocks 3 and 4, probably due to the fact that matching of cross oblique views is more difficult in urban areas than in suburban ones because of the more complex local shape change caused by high levels of depth change in the city scene. However, the proportion of "nadir_obl" in city areas (blocks 3 and 4) is nearly twice that of the suburban blocks (blocks 1 and 2). Also, a nearly 50% improvement of "obl_nadir_obl" is observed in urban blocks (block 3 and 4) when compared to rural blocks (block 1 and 2). These two improvements contribute significantly to the stability of aerial image blocks after image orientation and, therefore, the precision of object point coordinates in urban areas (blocks 3 and 4) is considerably higher than in suburban areas. In this context, it should be recalled, that the oblique camera system, ground sampling distance and flying height are identical for all image blocks.

Compared to other learning based variants, AOH-Graz and AOW-Graz have significantly larger proportions for "nadir_obl", while both variants have significantly lower proportions for "obl_nadir_obl" and "obl_obl". This suggests that the AffNet and OriNet trained on Aerial-Graz are capable of linking nadir and one oblique view, but linking matches cross oblique views is too challenging for AOH-Graz and AOW-Graz in many cases.

When comparing the results of different variants in figure 5.19, it can be observed that the difference varies from block to block. In block 2, the difference between different learning based variants is less distinctive, possibly due to the fact that this block is the easiest to match. It presents a well-textured, relatively flat scene without high buildings, allowing different methods to perform equally well. As for the other three blocks, the differences between different variants do not show any systematic pattern. Therefore, the distribution of ray intersections at each 3D point is considered to be data dependent.

Apart from analyzing the intersection of rays per 3D point, the distribution of the matching points delivered after bundle adjustment in the 2D image plane is also an important indicator for the performance of the employed matching algorithms. In this experiment, the image planes are divided into grids of $50 \times 50$ pixels and the matching points located in each of these grids are counted. In the next step, the distribution of different variants in the image plane is computed. In general, a good matching algorithm should deliver a more even and dense distribution of matching points in the image plane. However, this distribution also depends on the scene content. The distribution of matching points for is shown in figure 5.20 (blocks 1 and 2) and 5.21 (blocks 3 and 4).

For the results of all four blocks, BSS leads to a sparse and non-uniform distribution of matching points in the image plane. This is mainly attributed to the fact that the hand-crafted modules in BSS are less invariant against the viewpoint and viewing direction change. Not surprisingly, AOH-Graz and AOW-Graz lead to a sparser result than other variants related to learned modules. This is consistent with the results repeated so far for AOH-Graz and AOW-Graz. All the other variants related to learned modules, result in denser and

more evenly distributed matching points in the 2D image plane.

Observing the result for block 2, as shown in figure 5.20b, it is evident that the matching points in this block are more evenly distributed with a higher concentration towards the central part of the image plane. The relatively large number (compared to the other three blocks) of matching points obviously corresponds to the large number of reconstructed 3D points. Also, this even distribution of matching points along different directions in the image plane leads to a similar range of mean object coordinate precision in X and Y direction. Compared to block 2, block 1 has a much more uneven distribution of matching points, largely due to the fact that the scene contains a large area with poor texture, as shown in figure 5.7a.

For the urban blocks 3 and 4, the results are shown in figure 5.21. Both blocks have evenly distributed matched points. However, when compared to block 3, a denser distribution is observed in block 4, leading to a 20-30% improvement for the number of reconstructed 3D points in comparison to block 3. The even distribution of the matched points in both blocks is largely due to the fact that well-textured content in the urban blocks leads to a even distribution of detected features.

| Block Index | Det+Aff.+ Ori.+Desc. | $N_{match}^{initial}$ | $N_{match}^{inl_2view}$ | $N_{match}^{surv_BA}$ | $R_{initial}^{inl_2view}$ | $R_{surv_BA}^{inl_2view}$ |
|---|---|---|---|---|---|---|
| Block-1 | BSS | 282755 | 21092 | 1232 | 7.46% | 5.84% |
| | BSH | 150067 | 32933 | 9585 | 21.95% | 29.10% |
| | BSW | 147788 | 35092 | 10188 | 23.74% | 29.03% |
| | AOH-Brown | 150453 | 43083 | **16027** | 28.64% | **37.20%** |
| | AOH-Graz | 122482 | 32179 | 4721 | 26.27% | 14.67% |
| | AOW-Brown | 145854 | **47398** | **18678** | **32.50%** | **39.41%** |
| | AOW-Graz | 122915 | 35936 | 4773 | 29.24% | 13.28% |
| | MMH-Brown | 153527 | 38147 | 14121 | 24.85% | 37.02% |
| | MMH-Graz | 152577 | 38083 | 12939 | 24.96% | 33.98% |
| | MMW-Brown | 154556 | 41821 | 14574 | 27.06% | 34.85% |
| | MMW-Graz | 152238 | 41573 | 13145 | 27.31% | 31.62% |
| | FuH-Brown | 153259 | 40578 | 14135 | 26.48% | 34.83% |
| | FuH-Graz | 151415 | 40067 | 14890 | 26.46% | **37.16%** |
| | FuW-Brown | 145187 | **43695** | 15904 | **30.10%** | 36.40% |
| | FuW-Graz | 145766 | **44411** | **16013** | **30.47%** | 36.06% |
| Block-2 | BSS | 362802 | 47099 | 15512 | 12.98% | 32.93% |
| | BSH | 213143 | 74433 | 43846 | 34.92% | 58.91% |
| | BSW | 211735 | 80940 | 48026 | 38.23% | 59.34% |
| | AOH-Brown | 221214 | 95678 | **63938** | 43.25% | **66.83%** |
| | AOH-Graz | 184145 | 75990 | 30974 | 41.27% | 40.76% |
| | AOW-Brown | 216376 | **103205** | **68273** | **47.70%** | **66.15%** |
| | AOW-Graz | 182911 | 82518 | 33473 | 45.11% | 40.56% |
| | MMH-Brown | 219980 | 85929 | 55356 | 39.06% | 64.42% |
| | MMH-Graz | 218812 | 85901 | 55235 | 39.26% | 64.30% |
| | MMW-Brown | 220550 | 92553 | 59675 | 41.96% | 64.48% |
| | MMW-Graz | 219155 | 93494 | 60542 | 42.66% | 64.75% |
| | FuH-Brown | 221538 | 90465 | 59267 | 40.83% | **65.51%** |
| | FuH-Graz | 219130 | 89668 | 58082 | 40.92% | 64.77% |
| | FuW-Brown | 214697 | **97696** | **63795** | **45.50%** | 65.30% |
| | FuW-Graz | 213481 | **97452** | 63179 | **45.65%** | 64.83% |

Table 5.15.: Number of matches generated by different variants for image block 1 and 2. $N_{match}^{initial}$: number of initial matches. $N_{match}^{inl_2view}$: number of inlier matches after running two view geometry estimation using RANSAC. $N_{match}^{surv_BA}$: number of matches after running bundle adjustment. $R_{initial}^{inl_2view}$: ratio of inlier matches/initial matches. $R_{surv_BA}^{inl_2view}$: ratio of matches after bundle adjustment/inlier. Except for $N_{match}^{inl_2view}$, the highest three numbers for each term are indicated in bold type.

| Block Index | Det+Aff.+ Ori.+Desc. | $N_{match}^{initial}$ | $N_{match}^{inl_2view}$ | $N_{match}^{surv_BA}$ | $R_{initial}^{inl_2view}$ | $R_{surv_BA}^{inl_2view}$ |
|---|---|---|---|---|---|---|
| Block-3 | BSS | 398160 | 25542 | 2537 | 6.41% | 9.93% |
| | BSH | 240728 | 40091 | 11469 | 16.65% | 28.60% |
| | BSW | 230211 | 41440 | 12404 | 18.00% | 29.93% |
| | AOH-Brown | 241260 | **48390** | **17457** | 20.06% | **36.08%** |
| | AOH-Graz | 198911 | 34583 | 8346 | 17.39% | 24.13% |
| | AOW-Brown | 226582 | **51292** | **19071** | **22.64%** | **37.18%** |
| | AOW-Graz | 185414 | 38195 | 9413 | **20.60%** | 24.64% |
| | MMH-Brown | 238068 | 42128 | 14245 | 17.70% | 33.81% |
| | MMH-Graz | 234594 | 41490 | 12399 | 17.69% | 29.88% |
| | MMW-Brown | 229634 | 45125 | 14235 | 19.65% | 31.55% |
| | MMW-Graz | 225185 | 44625 | 14535 | 19.82% | 32.57% |
| | FuH-Brown | 233589 | 43793 | 14218 | 18.75% | 32.47% |
| | FuH-Graz | 232805 | 42417 | 13324 | 18.22% | 31.41% |
| | FuW-Brown | 219296 | **46465** | **16728** | **21.19%** | **36.00%** |
| | FuW-Graz | 218711 | 45913 | 16006 | 20.99% | 34.86% |
| Block-4 | BSS | 555595 | 39548 | 2289 | 7.12% | 5.79% |
| | BSH | 334780 | 58108 | 16758 | 17.36% | 28.84% |
| | BSW | 318473 | 60342 | 19704 | 18.95% | 32.65% |
| | AOH-Brown | 327279 | **67612** | **27111** | 20.66% | **40.10%** |
| | AOH-Graz | 274719 | 51170 | 13180 | 18.63% | 25.76% |
| | AOW-Brown | 303723 | **72457** | **30212** | **23.86%** | **41.70%** |
| | AOW-Graz | 252957 | 54291 | 14651 | 21.46% | 26.99% |
| | MMH-Brown | 329149 | 60471 | 19117 | 18.37% | 31.61% |
| | MMH-Graz | 324911 | 60215 | 20816 | 18.53% | 34.57% |
| | MMW-Brown | 314540 | 64533 | 22569 | 20.52% | 34.97% |
| | MMW-Graz | 310978 | 64265 | 22581 | 20.67% | 35.14% |
| | FuH-Brown | 325979 | 63324 | 22632 | 19.43% | 35.74% |
| | FuH-Graz | 324651 | 62874 | 23347 | 19.37% | 37.13% |
| | FuW-Brown | 303728 | 66507 | 24516 | **21.90%** | 36.86% |
| | FuW-Graz | 302219 | **66816** | **24891** | **22.11%** | **37.25%** |

Table 5.16.: Number of matches generated by different variants for image block 3 and 4. $N_{match}^{initial}$: number of initial matches. $N_{match}^{inl_2view}$: number of inlier matches after running two view geometry estimation using RANSAC. $N_{match}^{surv_BA}$: number of matches after running bundle adjustment. $R_{initial}^{inl_2view}$: ratio of inlier matches/initial matches. $R_{surv_BA}^{inl_2view}$: ratio of matches after bundle adjustment/inlier. Except for $N_{match}^{inl_2view}$, the highest three numbers for each term are indicated in bold type.




Figure 5.18.: Number and ratio of matches for different blocks. From top to bottom: number of matches after bundle adjustment ($N_{match}^{surv_BA}$), number of two view inlier matches ($N_{match}^{inl_2view}$), ratio of two-view inlier matches ($R_{initial}^{inl_2view}$) and ratio of matches after bundle adjustment ($R_{surv_BA}^{inl_2view}$). All four of the bar figures share the same label for different variants, as shown in the top figure. For the two related ratios (3rd and 4th row), the mean values over the four groups are also provided at the end of each bar figure.

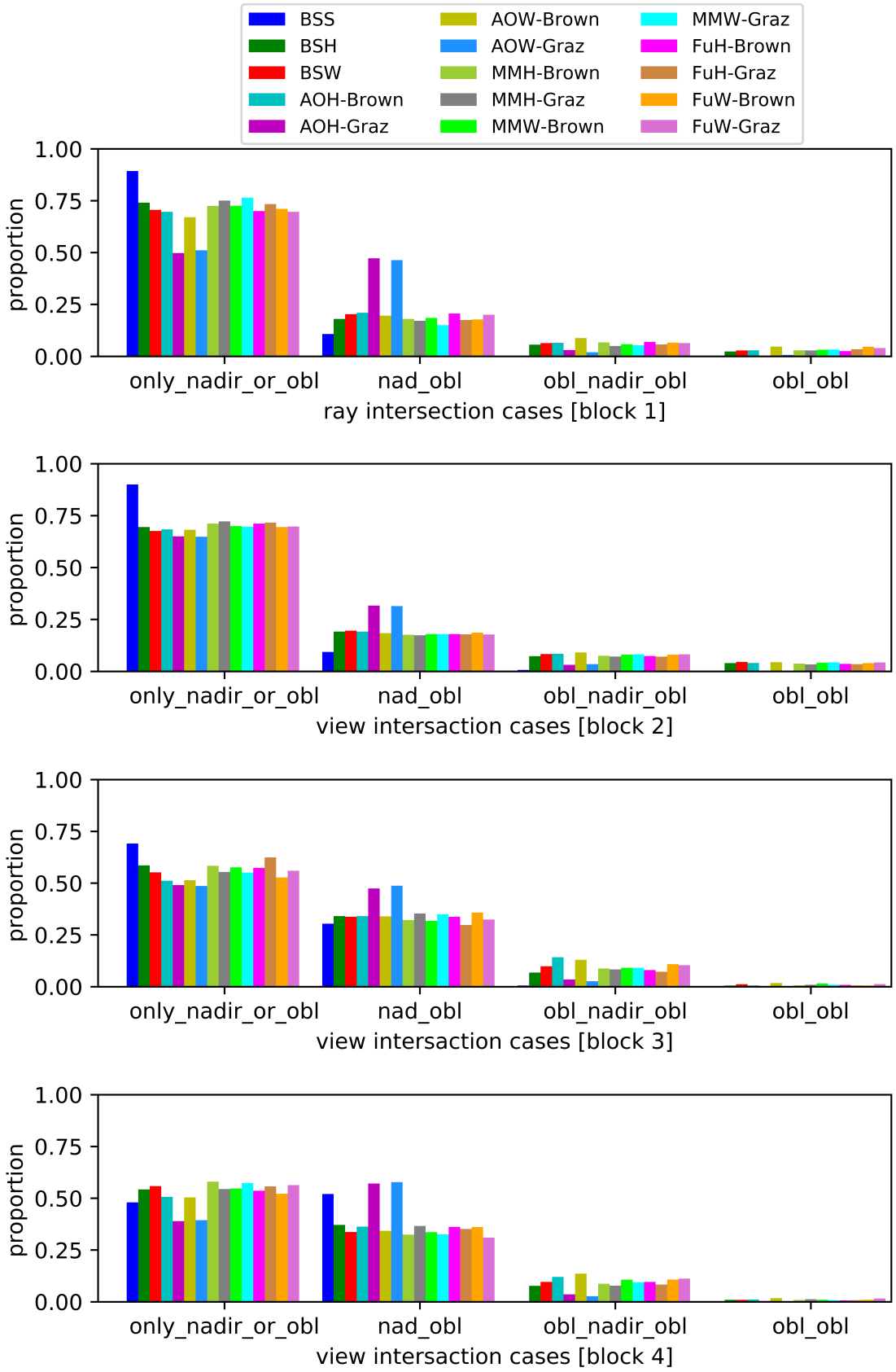
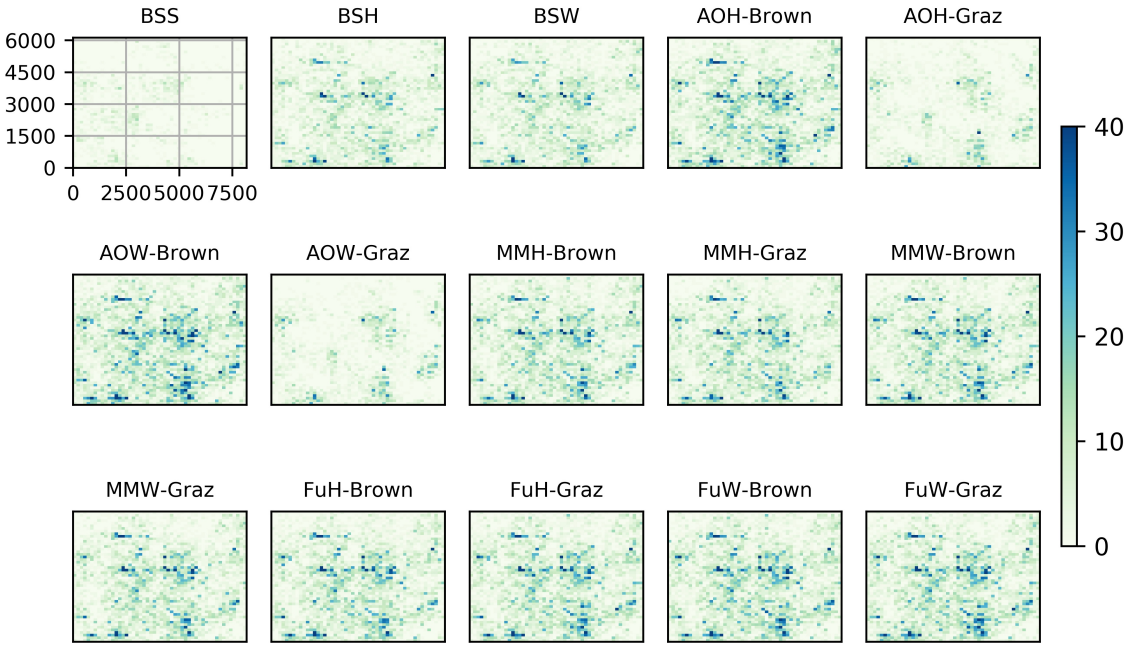
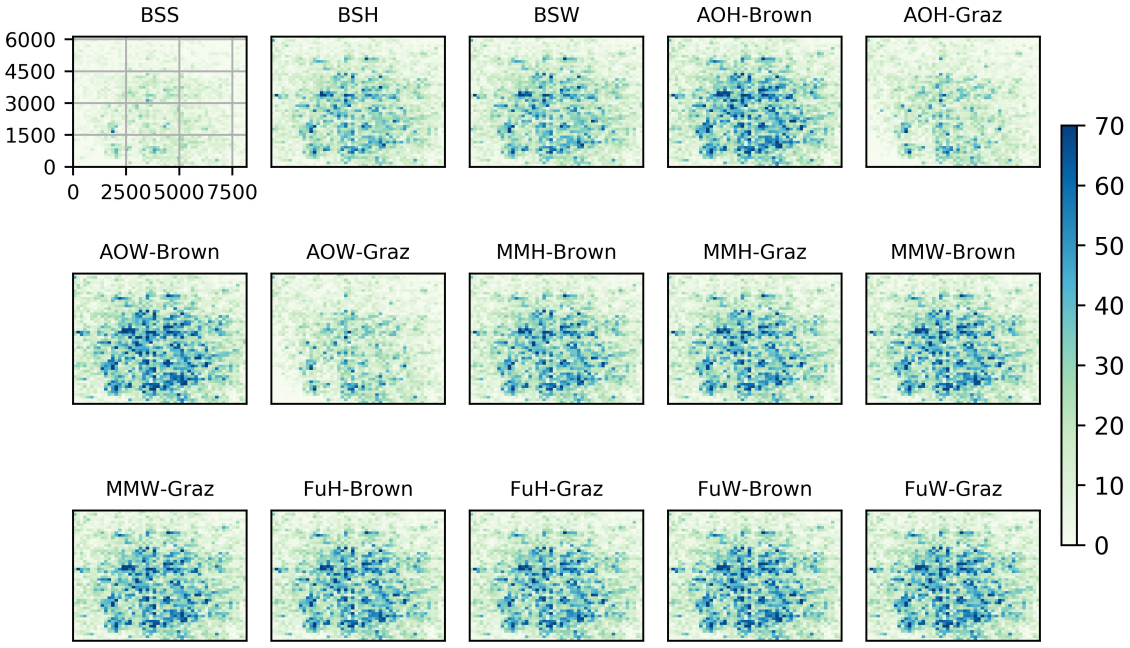



Figure 5.19.: Ray intersection for the match track of each 3D object point. From top to bottom the results are given for blocks 1 through 4. All four of the bar figures share the same label for the different variants as shown above the results for the first block.

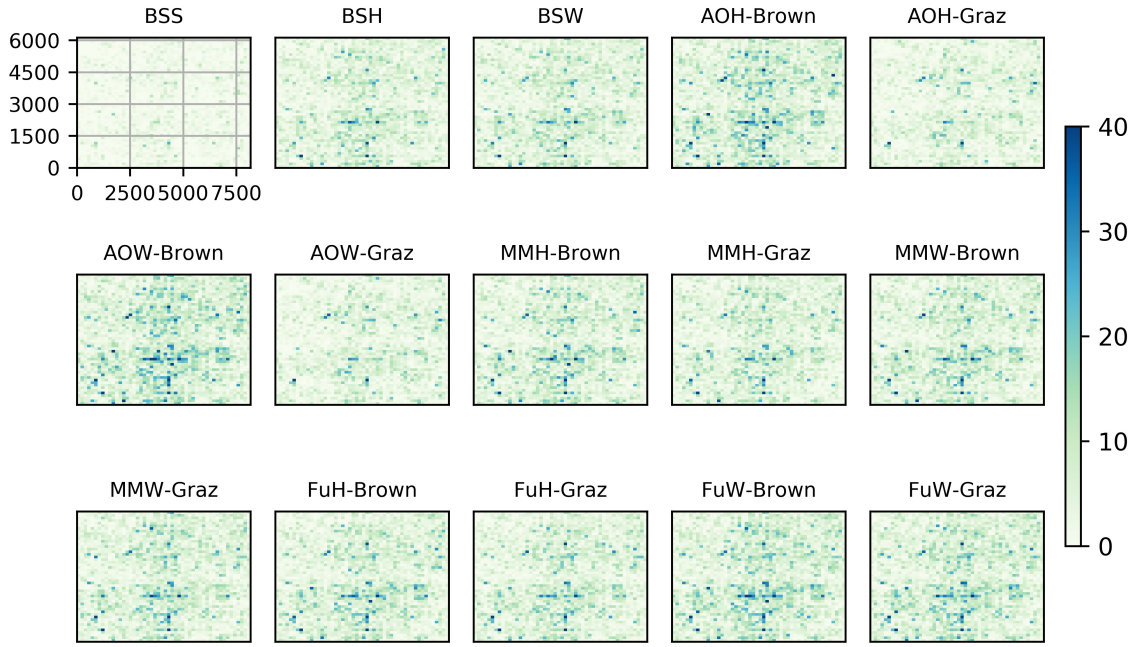



(a) block1

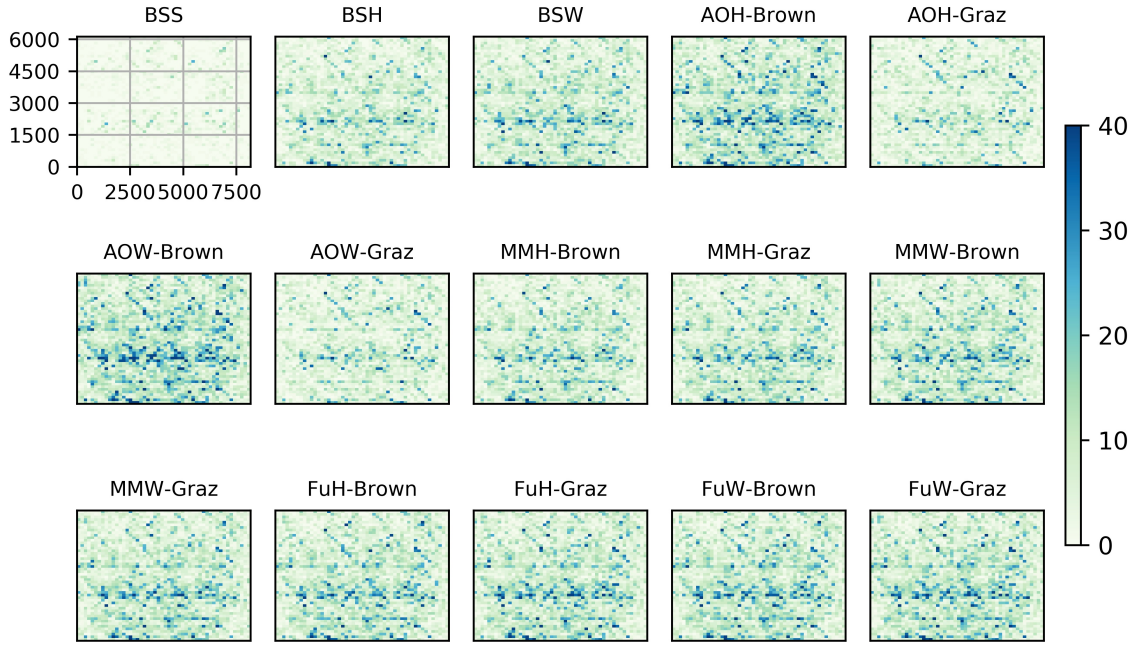



(b) block2

Figure 5.20.: The distribution of matching points after bundle adjustment applying different matching variants on blocks 1 (a) and 2 (b). For better visualization, the axes and grid are only shown in the HBSS result of each block and it is identical for all variants. The colour bars used in the two blocks show different ranges (number of matched point per grid), due to the fact that block 1 delivers a significantly smaller number of matches and reconstructed 3D points.




(a) block3




(b) block4

Figure 5.21.: The distribution of matching points after bundle adjustment using different matching method variants in blocks 3 (a) and 4 (b). For better visualization, the axes and grid are only shown in the HBSS result of each block, and it is identical for other variants. The colour bars used in blocks 3 and 4 are identical.

# 6. Discussion

This section first provides the discussion based on the experimental results presented in the previous chapter. For the sake of simplicity, the discussion is structured according to the investigations in chapter 5 which depicts the experiments and results. Some fundamental issues, such as the correctness of parameter study policy and the implications of the achieved experimental results, are discussed. Also, the strengths and limitations of the different modules proposed in this thesis are analysed. In accordance with those findings and an analysis of limitations, possible directions of future work are outlined.

## 6.1. Descriptor Learning and Patch Based Image Matching

### 6.1.1. Parameter Study

The parameter study for the proposed WeMNet involves the relative weight of weak match loss ($\lambda_{wm}$). The results are provided in table 5.4 and figure 5.9. The weak match branch contributes to the invariance of descriptors when $\lambda_{wm}$ is larger than 0.01. However, when $\lambda_{wm}$ is larger than 5.0, this improvement starts to diminish for "hard" and "tough" patches. In the case that $\lambda_{wm}$ is larger than 5, the whole descriptor loss is governed by the weak match part. Due to the fact that the weakly matched patches are sampled from the central part of the input feature patches, those patches used for training the weak match branch have smaller context windows. Assigning high weights to the descriptor trained on those smaller context windows could decrease the discriminative power of the resulting trained descriptor again.

In theory, the thresholds used in the descriptor distance based loss, i.e., $\beta$ and $\beta_{weak}$ (the margins between the matched and unmatched feature pairs in descriptor space), should also be studied in a systematic way. In an ideal case, a grid search over all three hyper-parameters, $\lambda_{wm}$, $\beta$ and $\beta_{weak}$, should be conducted. To keep the computation burden tractable in this research, however, the parameter study for $\beta$, $\beta_{weak}$ is ignored. Instead, $\beta$ and $\beta_{weak}$ are set empirically. For all the experiments in this thesis, $\beta$ is set to 1.0 for triplet loss and $\beta_{weak}$ to 0.8. Considering the fact that descriptors are normalized and the possible distances between any two descriptors thus lies in the range of $[0, 2.0]$, setting the distance margin threshold to 1.0 seems reasonable. As more challenging weakly matched patches are included for the weak match branch in descriptor learning, $\beta_{weak}$ should have a lower value that forms a looser constraint for harder training data. Therefore, $\beta_{weak}$ is set to 0.8. Although $\beta$ and $\beta_{weak}$ are set with meaningful values, there is a risk that the

values for those two parameters for the proposed WeMNet are not optimal. It is to be expected that with a more complete study of parameters the performance of the descriptor proposed in this thesis will improve slightly.

### 6.1.2. Comparison to Related Works

The descriptor is compared to its counterparts in closely related works on the matching task using the Hpatches benchmark [Balntas et al., 2017]. As can be seen from figure 5.10, the matching results of different variants on the illumination set are relatively stable. However, the result for the set View changes dramatically. The dataset used in *HardNet_PS* [Mitra et al., 2018] is deliberately designed to simulate challenging viewpoint and viewing direction change. Due to the fact that the amount of training data[1] used in *HardNet_PS* is at least ten times the size of the dataset used for other variants, it does not come as a surprise that a better performance in the view set is achieved.

By restricting the comparison of variants to cases where the same network and training data are used, i.e., *HardNet_Lib*, *SOSNet_Lib* and *WeMNet_Lib*, the advantage of including weak matches becomes obvious, as observed from the result for the sets View and Full. Even with a smaller dataset (Liberty), the variant *WeMNet_Lib* achieves better performance than *HardNet_Brown* which is trained with the entire Brown dataset. As discussed in section 3.5.2, the intra-variance in the matched patch pairs, i.e., including enough changes of appearance for matched pairs during the training process, was not fully explored and used in descriptor learning. In this thesis, however, the involvement of the proposed weak match branch helps the network to discover the potential of including enough intra-variance for matched pairs and thus achieving higher invariance under the condition that a limited quantity of training data is available.

However, when a larger dataset is used, the improvement for WeMNet is marginal, as can seen in *WeMNet_Lib* vs *WeMNet_Brown6*. This may imply a risk of over-fitting caused by the mechanism used in WeMNet. To be precise, the network may put too much emphasis on differentiating matched features whose feature support windows are slightly transformed to each other, and thus the trained network may be overly dependent on the training data. Consequently, the generalization power of the trained network could be restricted.

## 6.2. Descriptor Distance Analysis

To explore the issue how descriptor distance change by geometric transformations, two different groups of data and three different typical geometric transformations are chosen for analysis. The results are shown in section 5.4.

[1] A more detailed description of the dataset used in training *HardNet_PS* can be found here: https://github.com/rmitra/PS-Dataset

### 6.2.1. Translation

The sensitivity of descriptor distance towards translation of the local image context window is high. This observation is relatively independent of the type of descriptor and data, as illustrated in figure 5.11. On the one side, this observation suggests that the invariance of the involved local descriptors against localization errors of feature points is only bound to a limited range, e.g., not larger than half of the characteristic scale of underlying features. On the other side, this observation is consistent with applications, in which local descriptors are employed as similarity measures differentiating proposals (e.g., object proposal for the purpose of visual object detection, pixels in stereo matching) with different localizations. For instances, the SIFT descriptor is used as descriptor for object tracking [Zhou et al., 2009] and the DAISY descriptor is used as similarity measure in stereo matching [Tola et al., 2009].

### 6.2.2. Rotation

The sensitivity analysis of descriptor distance against rotation contains a lot of information. First, as mentioned already in section 5.4.2, when one patch is fixed and the other one is rotated in a pair, i.e., FixA_RotP, different descriptors deliver different results. The relevant issue here is that descriptor distance can be a less discriminative measure for learning the orientation of local features when the relative rotation difference lies beyond the sensitivity range of the descriptors employed. For instance, the range lies beyond $-60°$ to $60°$ for HardNet. However, in the real orientation network training stage, the patches used in each mini-batch are randomly simulated with different rotation angles, therefore a larger gradient, computed by the derivatives of loss with regard to the predicted angles, can normally be obtained by feature pairs which are simulated with a small amount of rotation difference. Second, the descriptor distance for the case RotA_RotP stays constant. This indicates that once local patches have been aligned, the descriptor distance is fixed, no matter which angles are used for alignment. Thus, no single solution for the alignment of feature patches is guaranteed. This leads to the over-parameterization problem mentioned in section 3.5.1. In this thesis, the involvement of mean gradient loss guarantees that the canonical unique solution can be found, as long as no symmetric pattern is contained in the feature context window.

### 6.2.3. Affine Shape Transformation

For the affine shape transformation, it can be observed that the changing trends of descriptor distance in the direction of $\phi$ (skewness) and stretch are very different. This suggests that optimization might prove difficult if the descriptor distance computed using HardNet or SIFT is employed as loss for training affine shape. The elongated shape of the contours suggests that a very different scale of gradients can be obtained for skewness and stretch. In turn, this may lead to a failure of convergence. Actually, when HardNet is used for training AffNet, the descriptor distance loss is significantly higher than the one obtained for descriptor training itself. This may serve as evidence that the derived affine shape is not globally optimized in the parameter space

formed by $\phi$ and stretch.

In case the same amount of affine shape of transformation is used for two patches in a pair, the distance is independent of the amount of transformation once the patch pair is aligned. This again confirms the over-parametrization problem mentioned in section 3.5.1 exists. In this thesis, joining the stretch and skew loss based on the second moment matrix is much better suited to compute a unique solution for the affine shape estimation.

## 6.3. Feature based Image Matching

In this part, the parameter $\lambda_{skew}$ is studied first in order to choose a good value so that a good feature matching performance is attained. Then, the trained modules are assessed with a rotation set of images and a set including affine transformations.

### 6.3.1. Parameter Study

The parameter study for $\lambda_{skew}$ is conducted in section 5.5.1. According to the results for which either HardNet or WeMNet is employed as descriptor, the proposed affine network is rather insensitive to the choice of $\lambda_{skew}$. The parameters, $\lambda_{skew}$ and $\lambda_{ori}$ in the full affine shape estimation module are adjusted during training, thus the three different losses can drop simultaneously. Based on the parameters set in this thesis, it can be observed that the stretch loss is made to decrease first, and the importance of the orientation loss increases after the stretch has been brought to a lower value. As the scale of skew loss is marginal, a small value for it is employed for all epochs. It is worth to note this strategy setting in full affine shape network training works fine for both the Brown and the Aerial-Graz dataset.

### 6.3.2. Rotation Set

As shown in figure 5.15, all variants that are combined with different feature orientation and descriptors generally perform better when the relative rotation of two images is smaller than they do in case of a larger relative rotation. Compared to the matching performance for smaller relative rotation (e.g., $[-45°, 45°]$), a performance drop of nearly 50% is observed for all variants in which the relative rotation between two images is moved to the maximum value ($\pm 180°$). This implies that not all of the involved feature orientation variants achieve a full rotation invariance in $[-180°, 180°)$.

An observation beyond expectation is that the variants of FuH, FuW , MH, when training with Brown or Aerial-Graz, and OH-Brown show a notable performance drop when the relative rotation changes from $\pm 45°$ to $\pm 15°$. It was found that this drop is mainly caused by the drop in 4 of the 59 used subsets. For these subsets, the performance drop from $\pm 45°$ to $\pm 15°$ is in range of 0.25 to 0.31 (in terms of AuC). This might

be partly due to the fact that WeMNet is less sensitive to rotation change in a small range (see Fig 5.12c and 5.12f) .

### 6.3.3. Affine Set

As shown in figure 5.16, the different variants involved are very comparable to each other, except for BSS, BSH, BSW, AOH-Graz and AOW-Graz. The difference between the variants is affected by the change of the ground truth matching thresholds used by said variants. For nearly all cases, it can be observed that HardNet performs nearly identical to WeMNet. In contrast, when the ratio and number of inliner matches are used in the oblique aerial image datasets in Task D, WeMNet performs slightly better than HardNet. This suggests that the result is data dependent.

Another positive aspect stems from the fact that MoNet, MGNet and Full-AffNet trained using the aerial image dataset (Aerial-Graz) perform well on close range images in Hpatches. For the other training dataset, i.e. Brown, both AffNet + OriNet and the networks (MGNet + MoNet and Full-AffNet) proposed in this thesis perform well. This implies that the generalization of the trained affine shape and orientation network is good enough for matching images from different domains.

One possible risk lies in the type of scene contained in the used data, i.e., planar dominant contents or pure rotation camera poses. On the one hand, this simplifies the process of generating ground truth correspondences because only the homography matrix is needed to model the pixel-wise geometric relationship between two test images. On the other hand, the scene type is not fully representative. Therefore, extending this evaluation task to images containing large viewpoint and viewing change while imaging real 3D scene is a meaningful direction for future work on this topic.

## 6.4. Image Orientation

In order to evaluate the performance of the proposed feature matching modules when they are applied to real image orientation tasks, small blocks of images that cover different types of images are utilized. The result changes from block to block. In the city area, mainly owing to the block geometry and the fact that an increased proportion of cross camera matches was found by the deep learning based feature matching variants, the object coordinate precision improves significantly (results of block 3 and 4 compared to block 1 and 2) and therefore the stability of the reconstructed block is improved accordingly.

Although the proposed MoNet, MGNet and Full-AffNet are trained with one branch and do not require matching relationships of features as a label, the variants based on these proposed modules achieve result comparable to those of AffNet and OriNet trained on the Brown dataset, for which Siamese CNN are used and a matching relationship is needed to supervise the training process. This result suggests that the proposed method can be used to solve real tasks where challenging viewpoint and viewing direction change is included.

Despite the fact that the proposed method is tested on images of rural and urban areas, this only covers applications on plain landscape, as the both imaging cities, Dortmund and Zeche Zollern, are located in the North German Plain. The performance of the proposed method for other types of landscapes, e.g., mountain areas, still remains unknown. This restriction comes from the lack of proper oblique aerial image benchmarks which would be representative enough for varying types of landscapes. Therefore, the result for Task D in this thesis can be seen as a pilot study for applying the proposed matching algorithm to real image orientation tasks. In this pilot study, promising results with regard to real image orientation are achieved.

Each of the blocks 1 through 4 contains small image blocks made up from only 3 to 5 images from each camera view in the penta-camera system. Some of the adjacent images, as mentioned in section 5.1.2, were excluded in order to create more challenging viewpoint and viewing direction change. Once the approach is verified to be efficient for smaller blocks, the extension to large and complete image blocks should work as well.

To make the comparison between AffNet and OriNet of [Mishkin et al., 2018] and MoNet, MGNet and Full-AffNet proposed in this thesis as fair as possible, identical datasets and an equal number of data are used in training. In particular, 10M patch pairs for AffNet and OriNet and 10.2M patch for MoNet, MGNet and Full-AffNet are used. Regarding the training dataset, all the network are trained on both Brown and Aerial-Graz. For AffNet and OriNet trained on Brown, as already mentioned, the version trained by the author of Mishkin et al. [2018] is used, while the version using Aerial-Graz is trained by the author of this thesis. As observed in task C and D, the AffNet and OriNet trained on Aerial-Graz experienced a significant performance drop. This is shown in task C.3 and task D. This drop may stem from two aspects:

- The hard negative constant loss proposed in [Mishkin et al., 2018]may be data dependent and thus this loss cannot cope with Aerial-Graz dataset as well as with Brown;
- The inconsistency among the training data for descriptor, affine shape and orientation related networks. In particular, the Brown dataset is the training data for HardNet, as well as AffNet and OriNet published in [Mishkin et al., 2018]. In this thesis, HardNet trained on Brown is used for training of AffNet and OriNet on both Brown and Aerial-Graz dataset. When Aerial-Graz is used to train AffNet and OriNet, the dataset consistency among the training sets of descriptor, affine shape and orientation networks is violated.

For MGNet, MoNet and Full-AffNet, the versions trained on Brown and Aerial-Graz datasets performs equally well in task C and D, only small differences between the two versions are observed. Those images in the training data, i.e. Brown and Aerial-Graz and in the test datasets, i.e. Hpatches and Aerial-Dortmund, respectively, clearly come from different image domains. Therefore, a performance drop or decreasing generalization can be expected. However, the experimental result of Task D (as well as Task C) does not suggest an obvious performance drop or domain gap. This is probably due to the fact that only local image regions, are relied on for training the affine shape, orientation and full affine shape as well for descriptor learning. In other words, large context windows are not necessary, so that the reliance on data “domains” is

considerably weakened and thus contributes to a smaller domain gap compared to applications where larger context windows are indispensable for discriminating among different categories of objects. The result in this thesis shed some light on the generalization of the trained modules. However, this conclusion is still far from comprehensive. To find a complete and stable answer to the generalization issue, a systematic study using different training/testing dataset combinations is needed, which, in turn, suggests a direction for future work.

One of the issues which remains unexplored so far but is also essential to feature matching is feature detection. All of the variants in this thesis rely on the Hessian detector, as it is more invariant against viewpoint and viewing direction change than other hand crafted feature detectors. In the chapter presenting the results of the experiment conducted for this thesis, it is verified that the Hessian can detect repeatable features despite challenging viewpoint and viewing direction change, otherwise matching would not have been successful and the image blocks would not have been orientated. However, two closely related issues are still not explored. First, the repeatability of features remains unknown when large viewpoint and viewing direction change is included in the data. Consequently, it also remains unknown when the feature detection starts to become unreliable. The feature detector can be applied to image patches that are simulated with strong relative geometric transformation, then a repeatability constraint can be employed to force the location of detected features to be as consistent as possible. This leaves - second - the question whether the localization precision of detected features can be improved. This is critical to applications for which high precision for image orientation (camera pose, 3D object point coordinates) is required. Either forming a localization loss or even solving this in a larger context, e.g., bundle adjustment, can be issues to be dealt with in future work.

# 7. Conclusion and Outlook

Following the goal of improving feature based image matching for images containing large viewpoint and viewing direction change, this thesis addressed the closely related issues for several steps in the feature based image matching framework. Starting from improving the descriptor invariance against large viewpoint and viewing direction change, orientation and affine shape estimation using geometric measures as well as the process of simultaneously learning the affine shape and orientation module were discussed. Also, the proposed method was comprehensively assessed in the image orientation of aerial oblique image blocks. This chapter draws the conclusion based on the presented investigation and suggests the most promising future directions for further work on this topic.

By incorporating a weak match branch that actively finds weak matches in descriptor learning using a Siamese CNN architecture, the learned descriptor has been shown to be more invariant against viewpoint and viewing direction change. Through the weak match network, the appearance of the matched features is explored in a more comprehensive way, therefore the intra-variance of the appearance for the matched features is better considered and covered. As only a simple predefined geometric transformation model is provided for the weak match network to find the hardest matched feature pairs during descriptor learning, the influence of the types and range of geometric transformations on the descriptor performance remains unknown. This issue needs to be addressed in future work to increase the understanding of the influence caused by the choice of geometric transformations allowed between matched feature pairs.

A canonical feature patch is critical to the feature orientation assignment and affine shape estimation problem. With the help of the proposed mean gradient based orientation loss and second moment based skewness and stretch loss, the orientation and affine shape network are able to find the rotation and affine shape transformation so that the output patches have canonical forms. The proposed loss for both feature orientation and feature affine shape estimation is independent of descriptor distance, which makes possible to use a single branch CNN instead of the Siamese CNN for training. In a further step, a full affine network that simultaneously learns the orientation and affine shape of an input feature patch from scratch is also proposed. To the best of the author's knowledge, this is the first time that orientation and affine shape are reported to have been successfully trained simultaneously from scratch. To make it possible that the three separate losses regarding the orientation, skew and stretch of a local patch decrease at the same time, a policy of dynamically adjusting the relative weights of those three different losses during training is utilized. As a result, a plausible solution for the full affine shape estimation network is achieved. However, this strategy should be more systematically studied, so that a better weight adjustment strategy can be found to bring the

final losses for all the three parts to a minimum. Correspondingly, further performance improvement for the full affine shape estimation network can then be expected.

The proposed feature matching method is assessed in an image orientation task using oblique aerial image blocks in which challenging viewpoint and viewing direction change is included. Compared to the existing benchmarks, the dataset used for assessment in this thesis emphasizes the challenging case with a real application problem, i.e., image orientation, by varying the types of covered landscape in the image blocks as well as the combination of orientation, affine shape and feature description module. Through the detailed analysis of the results of this image orientation task, it is found in this thesis that the proposed descriptor leads to a slightly higher number of matching points. At the same time the proposed orientation and affine shape network achieve a result comparable to the one yielded by the network trained by a Siamese CNN which employs descriptor distance based loss. However, for the image pairs used in the orientation task, its ground truth matches still remained unknown. In the future, a strategy that can automatically provide the ground truth matches for image pairs containing large viewpoint and viewing direction change should be developed. As a consequence, the recall and precision of the developed matching method can be derived and then used as complementary performance measures for the measures used in the current work.

In this thesis, the ability to transfer learned modules to images that are different from the training data is preliminarily answered. As different datasets are used in the training and evaluation tasks, an acceptable generalization is observed for the trained modules. As mentioned before, the possible reasons for this generalization come from two aspects. First, all of the learned modules rely on a simple network composed of only several layers and a large quantity of training data is used to train them. Second, the feature patch relies mainly only on local image contents because the characteristic scales of most detected features are within the range of several pixels, therefore a larger context window is not required. As a result, for one local image pattern contained in the image domain used for training, a similar local image pattern can probably be found in the image domain where the trained networks are applied. However, the ability to transfer the learned modules is not systematically investigated in the current work. A more systematic and comprehensive study, which employs more diverse datasets and compares different combinations of training and testing dataset, may bring a clearer vision to the generalization issue of the proposed method.

So far, feature detection still relies on a hand-crafted detector, i.e., the Hessian detector. In the future, when feature detection is revisited, the question whether its performance can be improved by deep learning should be explored. Some existing research has already shed light on the promising aspects of learning feature detectors. For images containing large viewpoint and viewing direction change, the feature detection network can follow the line of achieving a higher repeatability and localization accuracy. Consequently, further improvement on the image orientation quality in terms of the accuracy of image and object points over this work can be expected.

# Bibliography

Aanaes, H., A. Dahl, and K. Steenstrup Pedersen. 2012. "Interesting interest points." *International Journal of Computer Vision* 97 (1): 18–35.

Agarwal, S., Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. 2011. "Building Rome in a day." *Communications of the ACM* 54 (10): 105–112.

Alahi, A., R. Ortiz, and P. Vandergheynst. 2012. "Freak: Fast retina keypoint." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 510–517.

Balntas, V., E. Johns, L. Tang, and K. Mikolajczyk. 2016a. "PN-Net: Conjoined triple deep network for learning local image descriptors." *arXiv preprint arXiv:1601.05030*.

Balntas, V., E. Riba, D. Ponsa, and K. Mikolajczyk. 2016b. "Learning local feature descriptors with triplets and shallow convolutional neural networks." Paper presented at *the Proceedings of the British Machine Vision Conference*, pp. 1–11.

Balntas, V., Lenc, K., Vedaldi, A., and Mikolajczyk, K. 2017. "Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5173–5182.

Barnard, S. T. and W. B. Thompson. 1980. "Disparity analysis of images." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2 (4): 333–340.

Baumberg, A. 2000. "Reliable feature matching across widely separated views." Paper presented at *the Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 774–781.

Bay, H., A. Ess, T. Tuytelaars, and L. Van Gool. 2008. "Speeded-up robust features (SURF)." *Computer Vision and Image Understanding* 110 (3): 346–359.

Bay, H., T. Tuytelaars, and L. Van Gool. 2006. "SURF: Speeded up robust features." Paper presented at *the Proceedings of the European Conference on Computer Vision*, pp. 404–417.

Beis, J. S. and D. G. Lowe. 1997. "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1000–1006.

Blott, G., Yu, J., and Heipke, C. (2019). "Multi-view person re-identification in a fisheye camera network with different viewing directions." *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 87(5-6):263–274.

Bromley, J., I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. 1994. "Signature verification using a "Siamese" time delay neural network." Paper presented at *the Proceedings of the Advances in Neural Information Processing Systems*, pp. 737–744.

Brown, M., G. Hua, and S. Winder. 2011. "Discriminative learning of local image descriptors." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33(1): 43–57.

Brown, M., R. Szeliski, and S. Winder. 2005. "Multi-image matching using multi-scale oriented patches." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Volume I, pp. 510–517.

Calonder, M., V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. 2012. "BRIEF: Computing a local binary descriptor very fast." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34 (7): 1281–1298.

Calonder, M., V. Lepetit, C. Strecha, and P. Fua. 2010. "BRIEF: Binary robust independent elementary features." Paper presented at *the Proceedings of the European Conference on Computer Vision*, pp. 778–792.

Carlevaris-Bianco, N. and R. M. Eustice. 2014. "Learning visual feature descriptors for dynamic lighting conditions." Paper presented at *the Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2769–2776.

Chechik, G., V. Sharma, U. Shalit, and S. Bengio. 2010. "Large scale online learning of image similarity through ranking." *Journal of Machine Learning Research* 11 (3): 1109–1135.

Chen, L., F. Rottensteiner, and C. Heipke. 2014. "Learning image descriptors for matching based on Haar features." In *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XL-3, pp. 61–68.

Chen, L., F. Rottensteiner, and C. Heipke. 2016. "Invariant descriptor learning using a Siamese convolutional neural network." In *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume III-3, pp. 11–18.

Chen, L., F. Rottensteiner, and C. Heipk. 2020a. "Deep learning based feature matching and its application in image orientation." In *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume V-2, pp. 25–33.

Chen, L., F. Rottensteiner, and C. Heipk. 2020b. Feature detection and description for image matching: from hand-crafted design to deep learning. *Geo-spatial Information Science*, 24(1):58–74. DOI:https://doi.org/10.1080/10095020.2020.1843376

Chung, D., Tahboub, K., and Delp, E. J. 2017. "A two stream siamese convolutional neural network for person re-identification." Paper presented at *the Proceedings of the IEEE International Conference on Computer Vision*. pp. 1983-1991.

Dreschler, L. and H.-H. Nagel. 1981. "On the frame-to-frame correspondence between greyvalue characteristics in the images of moving objects." Paper presented at *the Proceedings of the German Workshop on Artificial Intelligence*, pp. 18–29.

Fan, B., Kong, Q., Wang, X., Wang, Z., Xiang, S., Pan, C., and Fua, P. 2017. "A performance evaluation of local features for image based 3d reconstruction." *arXiv preprint arXiv:1712.05271*.

Fan, B., Q. Kong, X. Wang, Z. Wang, S. Xiang, C. Pan, and P. Fua. 2019. "A performance evaluation of local features for image-based 3D reconstruction." *IEEE Transactions on Image Processing 28*(10): 4774–4789.

Förstner, W. and E. Gülch. 1987. "A fast operator for detection and precise location of distinct points, corners and centres of circular features." In *Proceedings of ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data*, pp. 281–305.

Föstner, W. 1991. "*Statistische Verfahren für die automatische Bildanalyse und ihre Bewertung bei der Objekterkennung und -vermessung*." Habilitation Thesis, Deutsche Geodätische Kommission bei der Bayerischen Akademie der Wissenschaften, Nr. 370.

Goesele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S. M. 2007. Multi-view stereo for community photo collections. Paper presented at *the Proceedings of the IEEE International Conference on Computer Vision*, pp. 1–8.

Hadsell, R., S. Chopra, and Y. LeCun. 2006. "Dimensionality reduction by learning an invariant mapping." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1735–1742.

Han, X., T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. 2015. "Matchnet: Unifying feature and metric learning for patch-based matching." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3279–3286.

Harris, C. and M. Stephens. 1988. "A combined corner and edge detector." Paper presented at *the Proceedings of Alvey Vision Conference*, Volume 15, pp. 147–151.

Hoffer, E. and N. Ailon. 2015. "Deep metric learning using triplet network." Paper presented at *the Proceedings of the International Workshop on Similarity-Based Pattern Recognition*, pp. 84–92.

Ioffe, S. and Szegedy, C. 2015. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167*.

Jacobsen, K. and M. Gerke. 2016. "Sub-camera calibration of a penta-camera." In *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XL-3/W4, pp. 35–40.

Jahrer, M., M. Grabner, and H. Bischof. 2008. "Learned local descriptors for recognition and matching." Paper presented at *the Proceedings of the Computer Vision Winter Workshop*, pp. 39–46.

Jin, Y., D. Mishkin, A. Mishchuk, J. Matas, P. Fua, K. M. Yi, and E. Trulls. 2020. "Image matching across wide baselines: From paper to practice." *arXiv preprint arXiv:2003.01587*.

Ke, Y., R. Sukthankar, and I. C. Society. 2004. "PCA-SIFT: A more distinctive representation for local image descriptors." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 506–513.

Keller, M., Z. Chen, F. Maffra, P. Schmuck, and M. Chli. 2018. "Learning deep descriptors with scale-aware triplet networks." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2762–2770.

Kim, D.-G., W.-J. Nam, and S.-W. Lee. 2019. "A robust matching network for gradually estimating geometric transformation on remote sensing imagery." Paper presented at *the Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 3889–3894.

Kumar, B., G. Carneiro, I. Reid, et al. 2016. "Learning local image descriptors with deep Siamese and triplet convolutional networks by minimising global loss functions." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5385–5394.

LeCun, Y.. 1989. "Generalization and network design strategies." Technical Report CRG-TR-89-4, University of Toronto Connectionist Research Group, June 1989.

Lenc, K. and A. Vedaldi. 2016. "Learning covariant feature detectors." Paper presented at *the Proceedings of the European Conference on Computer Vision Workshops*, pp. 100–117.

Lepetit, V. and P. Fua. 2006. "Keypoint recognition using randomized trees." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(9): 1465–1479.

Leutenegger, S., M. Chli, and R. Y. Siegwart. 2011. "BRISK: Binary robust invariant scalable keypoints." Paper presented at *the Proceedings of the IEEE International Conference on Computer Vision*, pp. 2548–2555.

Lindeberg, T. 1994. "Scale-space theory: A basic tool for analyzing structures at different scales." *Journal of Applied Statistics* 21 (1-2):225–270.

Lindeberg, T. 1998. "Feature detection with automatic scale selection." *International Journal of Computer Vision* 30 (2): 79–116.

Lindeberg, T. 2015. "Image matching using generalized scale-space interest points." *Journal of Mathematical Imaging and Vision*, 52(1): 3–36.

Lindeberg, T. and J. Garding. 1997. "Shape-adapted smoothing in estimation of 3-D shape cues from affine deformations of local 2-D brightness structure." *Image and Vision Computing* 15 (6): 415–434.

Lowe, D. G. 1999. "Object recognition from local scale-invariant features." Paper presented at *the Proceedings of the International Conference on Computer Vision*, pp. 1150–1157.

Lowe, D. G. 2004. "Distinctive image features from scale-invariant keypoints." *International Journal of Computer Vision* 60 (2): 91–110.

Lucas, B. D., T. Kanade, et al. 1981. "An iterative image registration technique with an application to stereo vision." Paper presented at *the Proceedings of the DARPA Image Understanding Workshop*, pp. 121–130.

Luo, Z., T. Shen, L. Zhou, S. Zhu, R. Zhang, Y. Yao, T. Fang, and L. Quan. 2018. "Geodesc: Learning local descriptors by integrating geometry constraints." Paper presented at *the Proceedings of the European Conference on Computer Vision*, pp. 168–183.

Matas, J., O. Chum, M. Urban, and T. Pajdla. 2004. "Robust wide-baseline stereo from maximally stable extremal regions." *Image and Vision Computing* 22 (10): 761–767.

Mikolajczyk, K. 2002. "Interest point detection invariant to affine transformations." PhD diss., Institut National Polytechnique de Grenoble, France.

Mikolajczyk, K. and Schmid, C. 2001. "Indexing based on scale invariant interest points." Paper presented at *the Proceedings of the International Conference on Computer Vision*, pp. 525–531.

Mikolajczyk, K. and C. Schmid (2004). "Scale & affine invariant interest point detectors." *International Journal of Computer Vision* 60 (1): 63–86.

Mikolajczyk, K. and C. Schmid. 2005. "A performance evaluation of local descriptors." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (10): 1615–1630.

Mikolajczyk, K., T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. 2005. "A comparison of affine region detectors." *International Journal of Computer Vision* 65 (1-2): 43–72.

Mishchuk, A., D. Mishkin, F. Radenovic, and J. Matas. 2017. "Working hard to know your neighbor's margins: Local descriptor learning loss." Paper presented at *the Proceedings of the Advances in Neural Information Processing Systems*, pp. 4826–4837.

Mishkin, D., Radenovic, F., and Matas, J. 2017. "Learning discriminative affine regions via discriminability." *arXiv preprint arXiv:1711.06704*.

Mishkin, D., F. Radenovic, and J. Matas. 2018. "Repeatability is not enough: Learning affine regions via discriminability." Paper presented at *the Proceedings of the European Conference on Computer Vision*, pp. 284–300.

Mitra, R., Doiphode, N., Gautam, U., Narayan, S., Ahmed, S., Chandran, S., and Jain, A. 2018. "A large dataset for improving patch matching." *arXiv:1801.01466v3*.

Moravec, H. P. 1979. "Visual mapping by a robot rover." Paper presented at *the Proceedings of the International Joint Conference on Artificial Intelligence*, Volume 1, pp. 598–600.

Morel, J.-M. and G. Yu. 2009. "ASIFT: A new framework for fully affine invariant image comparison." *SIAM Journal on Imaging Sciences* 2 (2): 438–469.

Nair, V. and Hinton, G. E. 2010. "Rectified linear units improve restricted boltzmann machines." Paper presented at *the Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814.

Nesterov, Y. 2013. "Gradient methods for minimizing composite functions." *Mathematical Programming*, 140(1):125–161.

Nex, F., F. Remondino, M. Gerke, H.-J. Przybilla, M. Bäumker, and A. Zurhorst. 2015. "ISPRS benchmark for multi-platform photogrammetry." In *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume II-3/W4, pp. 135–142.

Nguyen, U. and C. Heipke 2020. "3D Pedestrian tracking using local structure constraints." *ISPRS Journal of Photogrammetry and Remote Sensing*, 166: 347–358.

Onyango, F., F. Nex, M. Peter, and P. Jende. 2017. "Accurate estimation of orientation parameters of uav images through image registration with aerial oblique imagery." In *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XLII-1/W1, pp. 599–605.

Osendorfer, C., J. Bayer, S. Urban, and P. Van Der Smagt. 2013. "Convolutional neural networks learn compact local image descriptors." Paper presented at *the Proceedings of the International Conference on Neural Information Processing*, pp. 624–630.

Ozuysal, M., M. Calonder, V. Lepetit, and P. Fua. 2010. "Fast keypoint recognition using random ferns." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(3): 448–461.

Perd'och, M., Chum, O., and Matas, J. 2009. "Efficient representation of local geometry for large scale object retrieval." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9–16.

Rodehorst, V. and A. Koschan. 2006. "Comparison and evaluation of feature point detectors." Paper presented at *the 5th International Symposium Turkish-German Joint Geodetic Days*, Technical University of Berlin, Berlin, Germany.

Rosten, E., R. Porter, and T. Drummond. 2010. "Faster and better: A machine learning approach to corner detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (1): 105–119.

Rublee, E., V. Rabaud, K. Konolige, and G. Bradski. 2011. "ORB: An efficient alternative to SIFT or SURF." Paper presented at *the Proceedings of the IEEE International Conference on Computer Vision*, pp. 2564–2571.

Savinov, N., A. Seki, L. Ladicky, T. Sattler, and M. Pollefeys. 2017. "Quad-networks: unsupervised learning to rank for interest point detection." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3929–3937.

Schönberger, J. L. and J.-M. Frahm. 2016. "Structure-from-motion revisited." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4104–4113.

Shi, J. and C. Tomasi . 1994. "Good features to track." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600.

Simo-Serra, E., E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. 2015. "Discriminative learning of deep convolutional feature point descriptors." Paper presented at *the Proceedings of the IEEE International Conference on Computer Vision*, pp. 118–126.

Simonyan, K., A. Vedaldi, and A. Zisserman. 2014. "Learning local feature descriptors using convex optimisation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (8): 1573–1585.

Smith, M., N. Kokkas, A. Hamruni, D. Critchley, and A. Jamieson. 2008. "Investigation into the orientation of oblique and vertical digital images." Paper presented at *the European Calibration and Orientation Workshop (EUROCOW)*. Barcelona, Spain.

Smith, S. M. and J. M. Brady. 1997. "SUSAN - A new approach to low level image processing." *International Journal of Computer Vision* 23 (1): 45–78.

Snavely, N., S. M. Seitz, and R. Szeliski. 2006. "Photo tourism: exploring photo collections in 3D." Paper presented at *the Proceedings of the International Conference on Computer Graphics and Interactive Techniques (ACM Siggraph)*, pp. 835–846.

Snavely, N., S. M. Seitz, and R. Szeliski. 2008. "Modeling the world from internet photo collections." *International Journal of Computer Vision* 80 (2): 189–210.

Szeliski, R. 2010. "*Computer vision: algorithms and applications*." Springer Science and Business Media, London. DOI:https://doi.org/10.1007/978-1-84882-935-0

Tian, Y., B. Fan, and F. Wu. 2017. "L2-net: Deep learning of discriminative patch descriptor in euclidean space." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 661–669.

Tian, Y., X. Yu, B. Fan, F. Wu, H. Heijnen, and V. Balntas. 2019. "Sosnet: Second order similarity regularization for local descriptor learning." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11016–11025.

Tola, E., V. Lepetit, and P. Fua. 2009. "Daisy: An efficient dense descriptor applied to wide-baseline stereo." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (5): 815–830.

Trzcinski, T., M. Christoudias, P. Fua, and V. Lepetit. 2013. "Boosting binary keypoint descriptors." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2874–2881.

Trzcinski, T., M. Christoudias, and V. Lepetit. 2015. "Learning image descriptors with boosting." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37(3): 597–610.

Tuytelaars, T. and Mikolajczyk, K. 2008. "Local invariant feature detectors: a survey." *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280.

Tuytelaars, T. and L. Van Gool. 2004. "Matching widely separated views based on affine invariant regions." *International Journal of Computer Vision 59*(1), 61–85.

Tuytelaars, T., L. Van Gool, L. D'haene, and R. Koch. 1999. "Matching of affinely invariant regions for visual servoing." Paper presented at *the Proceedings 1999 IEEE International Conference on Robotics and Automation*, pp. 1601–1606.

Tuytelaars, T. and L. J. Van Gool. 2000. "Wide baseline stereo matching based on local, affinely invariant regions." Paper presented at *the Proceedings of the British Machine Vision Conference*, pp. 38.1–38.14.

Verdie, Y., K. Yi, P. Fua, and V. Lepetit. 2015. "TILDE: a temporally invariant learned detector." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5279–5288.

Verykokou, S. and C. Ioannidis. 2016. "Automatic rough georeferencing of multiview oblique and vertical aerial image datasets of urban scenes." *The Photogrammetric Record* 31 (155): 281–303.

Verykokou, S. and C. Ioannidis. 2018. "Oblique aerial images: a review focusing on georeferencing procedures." *International Journal of Remote Sensing* 39 (11): 3452–3496.

Viola, P. and M. J. Jones. 2004. "Robust real-time face detection." *International Journal of Computer Vision* 57 (2): 137–154.

Wang, C., J. Chen, J. Chen, A. Yue, D. He, Q. Huang, and Y. Zhang. 2018. "Unmanned aerial vehicle oblique image registration using an ASIFT-based matching method." *Journal of Applied Remote Sensing* 12(2), 025002.

Winder, S. A. and M. Brown. 2007. "Learning local image descriptors." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Wu, C. 2013. "Towards linear-time incremental structure from motion." Paper presented at *the Proceedings of the International Conference on 3D Vision*, pp. 127–134.

Wu, C., S. Agarwal, B. Curless, and S. M. Seitz. 2011. "Multicore bundle adjustment." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3057–3064.

Yi, K. M., Y. Verdie, P. Fua, and V. Lepetit. 2016a. "Learning to assign orientations to feature points." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 107–116.

Yi, K. M., E. Trulls, V. Lepetit, and P. Fua. 2016b. "LIFT: Learned invariant feature transform." Paper presented at *Proceedings of the European Conference on Computer Vision*, pp. 467–483.

Zagoruyko, S. and N. Komodakis. 2015. "Learning to compare image patches via convolutional neural networks." Paper presented at *the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4353–4361.

Zhou, H., Yuan, Y., and Shi, C. 2009. "Object tracking using sift features and mean shift." *Computer Vision and Image Understanding*, 113(3): 345–352.

# A. Affine Shape Adaptation Theory

Affine shape adaptation theory is studied in works Lindeberg and Garding [1997]; Baumberg [2000]; Mikolajczyk and Schmid [2004]. Unlike rotation and uniform scaling in all directions, affine transformation introduce anisotropic scale change and it leads to the fact that isotropic measure like Gaussian with scales will not be invariant. Thus affine Gaussian space which can be invariant to affine transformation under deliberate design is introduced.

Let one image pattern $x_L$ on image $L$ is transformed to another image pattern $x_R$ on image $R$ through some invertible linear transformation $B$ in form of $x_L = Bx_R$. Furthermore, let two intensity patterns represent by $f_L, f_R$ respectively have the form of $f_L, f_R : \mathbb{R}^2 \to \mathbb{R}$, then under the transformation of $B$ it fulfils Lindeberg and Garding [1997]

$$f_L(x_L) = f_R(Bx_L)$$

## A.1. transformation of affine Gaussian scale-space

The affine Gaussian scale space is defined as:

$$\begin{aligned} L(\cdot;\Sigma_L) &= g(\cdot;\Sigma_L) * f_L(\cdot) \\ R(\cdot;\Sigma_R) &= g(\cdot;\Sigma_R) * f_R(\cdot) \end{aligned} \tag{A.1}$$

$g(\cdot;\Sigma)$ is Gaussian function with covariance matrix $\Sigma$. The covariance matrices $\Sigma_L, \Sigma_R$ are symmetric positive semi-definite matrix which deliver non-uniform Gaussian kernel for image $L$ and $R$. In this affine scale-space, the response value for $x_L$ and $x_R$ should be equal, thus it achieves invariance against the underline linear transformation $B$, i.e.,

$$L(x_L;\Sigma_L) = R(x_R;\Sigma_R) \tag{A.2}$$

In this case, $\Sigma_L$ and $\Sigma_R$ meet the following equation:

$$\Sigma_R = B\Sigma_L B^\top \tag{A.3}$$

$Proof$: first with the basic definition and considering that $d(B\xi) = \det B d\xi$

$$\begin{aligned} L(x_L; \Sigma_L) &= \int_{\xi \in \mathbb{R}^2} g(x_L - \xi; \Sigma_L) f_L(\xi) d\xi \\ &= (\det B)^{-1} \int_{\xi \in \mathbb{R}^2} g(B^{-1}(Bx_L - B\xi); \Sigma_L) f_R(B\xi) d(B\xi) \end{aligned} \tag{A.4}$$

The gaussian kernel transform under a linear transformation as

$$\begin{aligned} g(B^{-1}\zeta; \Sigma_L) &= \frac{1}{2\pi\sqrt{\det \Sigma_L}} e^{\frac{-(B^{-1}\zeta)^\top \Sigma_L^{-1}(B^{-1}\zeta)}{2}} \\ &= \sqrt{\det BB^\top} \frac{1}{2\pi\sqrt{\det B\Sigma_L B^\top}} e^{\frac{-\zeta^\top (B\Sigma_L B^\top)^{-1}\zeta}{2}} \\ &= \det B \frac{1}{2\pi\sqrt{\det B\Sigma_L B^\top}} e^{\frac{-\zeta^\top (B\Sigma_L B^\top)^{-1}\zeta}{2}} \\ &= \det B g(\zeta, B\Sigma_L B^\top) \end{aligned} \tag{A.5}$$

Further, we can convert the former one into the following form given $[\eta = B\xi]$:

$$\begin{aligned} L(x_L; \Sigma_L) &= (\det B)^{-1} (\det B) \int_{\xi \in \mathbb{R}^2} g(B^{-1}(Bx_L - B\xi); B\Sigma_L B^\top) f_R(B\xi) d(B\xi) \\ &= \int_{\eta \in \mathbb{R}^2} g(Bx_L - \eta; B\Sigma_L B^\top) f_R(\eta) d(\eta) \\ &= R(Bx_L; B\Sigma_L B^\top) \\ &= R(x_R, \Sigma_R) \end{aligned} \tag{A.6}$$

The above equations shows that when two image patterns are related by transformation $B$, then the affine Gaussian space of the two patterns is equal when $\Sigma_R = B\Sigma_L B^\top$. If we reverse this process and suppose the covariance matrices $\Sigma_L, \Sigma_R$ are known, we can further calculate the transformation $B$ between two image patterns.

## A.2. Local affine distortion measurement

For image $L : \mathbb{R}^2 \to \mathbb{R}$ and its gradient $\nabla L = (L_x, L_y)^\top$, the second momentum matrix of $L$ centred at $q$ defined with a window function $w$ is:

$$\begin{aligned} \mu_L(q) &= \int_{x \in \mathbb{R}^2} (\nabla L(x))(\nabla L(x))^\top w(q - x) dx \\ &= \int_{(x_1, x_2) \in \mathbb{R}^2} \begin{pmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{pmatrix} w(q_1 - x_1, q_2 - x_2) dx_1 dx_2 \end{aligned} \tag{A.7}$$

here $(q_1, q_2)$ is the image point under computation. $\mu_L(q)$ maps $\mathbb{R}^2$ to a symmetric positive semi-definite 2 x 2 matrix. Given transformation $[\eta = B\xi]$ and the fact that $\nabla L(\xi) = B^\top \nabla R(B\xi)$, $u_L(q)$ can be converted to:

$$\begin{aligned}\mu_L(q) &= \int_{\xi\in\mathbb{R}^2} B^\top(\nabla R(B\xi))(\nabla R(B\xi))^\top B w(q-\xi)d\xi \\ &= B^\top\left\{\int_{\eta\in\mathbb{R}^2}(\nabla R(\eta))(\nabla R(\eta))^\top w(B^{-1}(p-\eta))(\det B)^{-1}d\eta\right\}B\end{aligned} \tag{A.8}$$

Since $w(B^{-1}(p-\eta))(\det B)^{-1} = \det B w(p-\eta)(\det B)^{-1} = w(p-\eta)$, the above equation is further written as

$$\begin{aligned}\mu_L(q) &= B^\top\left\{\int_{\eta\in\mathbb{R}^2}(\nabla R(\eta))(\nabla R(\eta))^\top w(p-\eta))d\eta\right\}B \\ &= B^\top\mu_R(p)B\end{aligned} \tag{A.9}$$

This equation shows that the momentum matrix of two image patterns under transformation $B$ is linked by the above formula. If two eigenvalues of a second momentum matrix close to each other, it basically means that the computed feature has an isotropic shape.

The window function $w$ can be specified as a Gaussian function with an integration kernel $\Sigma_I$. The gradient parts $\nabla L, \nabla R$ is calculated with another differential kernel $\Sigma_D$. Thus it can be further written as:

$$\mu_L(q, \Sigma_I, \Sigma_D) = g(q, \Sigma_I) * ((\nabla L)(q; \Sigma_D)(\nabla L)(q; \Sigma_D)^\top) \tag{A.10}$$

## A.3. More affine transformation

Under a linear transformation $x_R = Bx_L$, the calculated momentum matrix at $x_L$ and $x_R$ transformed in the following way:

$$\begin{aligned}\mu_L(x_L, \Sigma_{I,L}, \Sigma_{D,L}) &= B^\top\mu(x_R, \Sigma_{I,R}, \Sigma_{D,R})B \\ &= B^\top\mu(Bx_L, B\Sigma_{I,L}B^\top, B\Sigma_{D,L}B^\top)B\end{aligned} \tag{A.11}$$

Here $\Sigma_{I,L}$ is the integration kernel and is $\Sigma_{D,L}$ the differentiation kernel used in the calculation of affine Gaussian scale space. With denote of $\mu_L(x_L, \Sigma_{I,L}, \Sigma_{D,L}) = M_L$ and $\mu_L(x_R, \Sigma_{I,R}, \Sigma_{D,R}) = M_R$, the following relationship can be obtained:

$$\begin{aligned}M_L &= B^\top M_R B \\ M_R &= B^{-\top} M_L B^{-1}\end{aligned} \tag{A.12}$$

Suppose $R$ is an arbitrary 2 x 2 rotation matrix, we can extend $M_L$ into the following term

$$\begin{aligned} M_L &= M_L^{1/2} R^\top M_R^{-1/2} M_R M_R^{-1/2} R M_L^{1/2} \\ &= (M_L^{1/2} R^\top M_R^{-1/2}) M_R (M_R^{-1/2} R M_L^{1/2}) \\ &= (M_R^{-1/2} R M_L^{1/2})^\top M_R (M_R^{-1/2} R M_L^{1/2}) \\ &= B^\top M_R B \end{aligned} \tag{A.13}$$

Thus the transformation $B$ can further represent by $M_L, M_R$ as:

$$B = M_R^{-1/2} R M_L^{1/2}$$

Substitute this $B$ into $x_R = Bx_L$, it is easily to get:

$$\begin{aligned} x_R = Bx_L = M_R^{-1/2} R M_L^{1/2} x_L \\ M_R^{1/2} x_R = R M_L^{1/2} x_L \end{aligned} \tag{A.14}$$

This shows that, when normalized frame $M_L^{1/2} x_L$ and $M_R^{1/2} x_R$ are related with a pure rotation $R$. The following Diagram shows the affine transformation diagram.

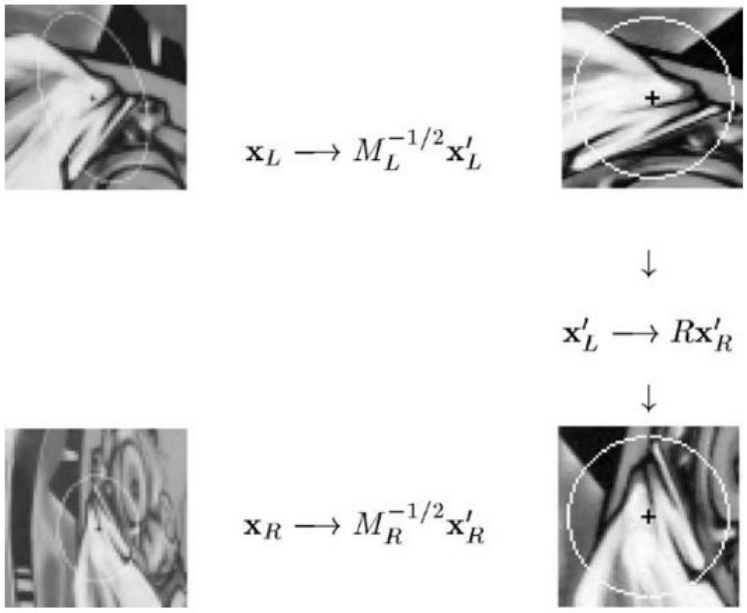



Figure A.1.: affine normalization based on 2nd momentum matrixMikolajczyk and Schmid [2004]

To calculate the affine transformation between images, iterative methods are used to estimate the affine corrected image patchesMikolajczyk and Schmid [2004]. It runs in the following steps:

1. Estimate the shape with second momentum matrix $M$ on detected feature points
2. Normalize the affine region with $M^{-1/2}$ and calculate the new second momentum matrix $M'$ for transformed region
3. Calculate the eigenvalue of $M'$ and go back to step 1 if the two eigenvalues of $M'$ are not close to each other

# Curriculum Vitae

## Personal Information

| | |
|---|---|
| Name | Lin Chen |
| Date / Place of Birth | 6th December 1987; Hubei, China |

## Work Experience

| | |
|---|---|
| Since October 2017 | Institute of Photogrammetry and GeoInformation,<br>Leibniz Universität Hannover<br>*Research Assistant* |
| Feb 2017 - April 2017 | Computer Vision Laboratory (CVLAB),<br>Swiss Federal Institute of Technology Lausanne (EPFL)<br>*Three-month Visiting PhD Student* |

## Education

| | |
|---|---|
| October 2013 - March 2021 | Institute of Photogrammetry and GeoInformation,<br>Leibniz Universität Hannover, Germany<br>*PhD study, China Scholarship Council (CSC) Doctoral Scholarship (2013-2017)* |
| October 2012 - June 2013 | School of Geodesy and Geomatics, Wuhan University, China<br>*One year of PhD study before exiting the program* |
| September 2010 - June 2012 | School of Geodesy and Geomatics, Wuhan University, China<br>*Master of Science* |
| September 2006 - June 2010 | School of Geodesy and Geomatics, Wuhan University, China<br>*Bachelor of Science* |
| September 2003 - June 2006 | High School, Badong No.1 High School, Hubei, China |

# Acknowledgement

I would like to thank the people who have supported me throughout my PhD journey.

My deepest gratitude goes to Prof. Christian Heipke for his great support during my studies. I appreciate the fruitful discussions, valuable suggestions and detailed corrections he provided. His support is not only in academia, but also in many aspects outside research.

I would like to thank Prof. Franz Rottensteiner for working as the referee, also the support he gave me during the past years, especially in the early stage of my PhD study at IPI. I would also like to thank Prof. Pascal Fua acting as referee and hosting me as a guest PhD student at computer vision lab for three months in 2017. Furthermore, I would like to thank Prof. Monika Sester acting as referee and Prof. Jürgen Müller for chairing the examination committee.

I would like to thank China Scholarship Council (CSC) for financially supporting my PhD study from 2013 to 2017. I would also like to thank Graduate Academy of Leibniz University Hannover for providing me a three-month scholarship during my visiting study at EPFL Computer Vision Lab. I thank NVIDIA Corp. for donating a Titan X GPU used in this research through its GPU grant program. Also, I would like to think Dr. Jens Kremer from Intergrated Geospatical Innovations (IGI) and Dr. Michael Gruber from Vexcel Imaging Austria for providing the aerial datasets used in the research related to this thesis.

I thank all the colleagues at IPI for sharing their knowledge and making the institute a pleasant place. In particular, I would like to thank Dr. Karsten Jocansen for sharing his knowledge during many coffee breaks, and Dennis Wittich for technically supporting the online defence of my Ph.D thesis. My thanks also go to Dr. Junhua Kang for the successful scientific collaborations we had during the past years and proof-reading my thesis.

I would like to thank all the laboratory members at EPFL Computer vision Lab for sharing their expertise and for a great time with the group in 2017. I thank Dr. Kwang Moo Yi (now Prof. Yi) and Dr. Eduard Trulls for supervising my work during my visiting study.

Finally, I would like to thank my family and friends for their continuous support. I especially thank my wife Jamin for her unconditional love and encouragement, as well as the great support during the time of writing and defending this thesis.