Homography Estimation in the Realm of Deep Learning

Projektivitätsbestimmung im Zeitalter des Deep Learnings
Master thesis by Robin Hesse
Date of submission: September 24, 2020

1. Review: Prof. Stefan Roth, Ph.D.
2. Review: Joaquin Zepeda, Ph.D.
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

vi
visual inference

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Robin Hesse, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 24. September 2020

R. Hesse

In cooperation with Amazon Rekognition

Abstract

Even though deep learning has caused a paradigm shift in most computer vision applications, traditional handcrafted methods are still used extensively for image registration. In this thesis, we show that deep learning also caused a paradigm shift in image registration and that the excessive use of traditional methods is only partly justified. In the process of doing so, we establish new benchmarks by providing novel metrics and updated datasets. Additionally, we propose DeCo-Net, a learning-based keypoint extraction method. Existing detect-then-describe approaches, e.g. RF-Net [68], obtain keypoint features by processing all patches around the extracted keypoints individually. DeCo-Net, on the other hand, forms keypoint features by computing a dense feature map for the entire image that is sampled at locations encoded in the local geometry of the keypoints. This way, redundant feature computation for close keypoints is avoided, and scalability improves. Furthermore, we decrease dependence between the keypoint detector and descriptor by redefining keypoints as locations with discriminative local geometries instead of discriminative features. To make optimal use of our keypoint-based approaches, we further propose C-RANSAC as a method to constrain RANSAC and enhance its homography estimation performance significantly. Lastly, we show that traditional methods can be surpassed by learning-based methods without the need for any human supervision by utilizing domain adaption and self-supervised training.

Acknowledgments

My deepest gratitude goes to everyone who made this thesis possible. This applies in particular to my supervisors Joaquin Zepeda, Ph.D., Luis Goncalves, Ph.D., and Prof. Stefan Roth, Ph.D.

Further, I would like to thank all the inspiring people from the *Visual Inference Lab* at the *Technical University of Darmstadt* and *Amazon Rekognition* who helped me in various forms and made the time of working on this project unforgettable.

Contents

1 Introduction

Imagine you have just spent hours hiking to reach the top of a beautiful mountain. You are stunned by the view and want to capture the moment. Fortunately, your smartphone offers the functionality to take a panoramic photograph by horizontally moving your phone. Under the hood, multiple shifted photographs of the panorama are aligned and stitched together to form a single panoramic photograph [51], as illustrated in Figure 1.1. This process of finding correspondences to align images into the same coordinate system is called image registration and is generally used to compare or integrate images obtained from different measurements. The different measurements may vary in time, viewpoint, object, or modality [8]. Image registration is particularly used in remote sensing and medical imaging [72]. Remote sensing applications include landscape planning, weather forecasting, registration of satellite and aerial data into maps, image mosaicing, and integration of information into geographic information systems [72, 58]. In the medical field, it is used to fuse computed tomography (CT) and nuclear magnetic resonance (NMR) data for more detailed diagnosis, to monitor tumor evolution, to multimodally analyze diseases where the therapy incorporates anatomical magnetic resonance imaging (MRI) and functional electroencephalogram (EEG) data, and to verify the progress of a healing bone from X-ray images taken at different points in time [72, 58]. Image registration is also used in many traditional computer vision challenges. For example, in super-resolution imaging by aligning multiple low-resolution frames of the same scene. Due to the relative motion between the camera and the scene, each frame contains different information that can be aggregated to reconstruct the true scene in a higher resolution [6]. In computer stereo vision, 3D information is extracted from two images by estimating the relative displacement of objects in images taken from two vantage points. Using the estimated displacement and information about the two vantage points, the distance of the objects can be computed. Furthermore, image registration is used as a pre-processing step in change detection, where the image under inspection is registered to the reference scene [22]. Afterwards, the superimposed scenes can be compared to detect changes. In addition to the applications just mentioned, there have been several attempts to use image registration as a solution to more fundamental computer vision problems. It can, for example, be used in object detection and classification by aligning the image under inspection to predefined object templates [2]. Or, it can be used for segmentation by aligning a reference image, often called atlas, to the object of interest [3]. As you can see, many problems in computer vision can be reduced to the problem of finding image correspondences, and thus, having a general solution to image registration would solve a multitude of related problems.

Probably the most prominent approach to image registration is to find distinctive points, so-called keypoints, in an image that are described with a feature vector. The keypoint features can then be used to find corresponding keypoints from two overlapping images. Finally, two images

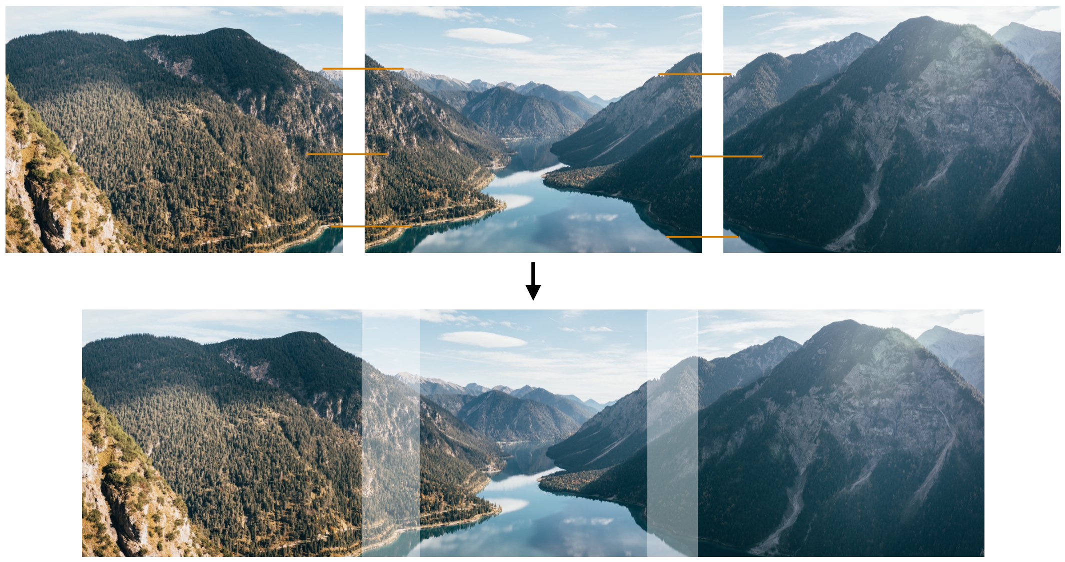


Figure 1.1: Illustration of the panorama stitching algorithm. Corresponding points in overlapping images are found (orange lines) to stitch the images together to a larger panoramic photography. The white shading denotes the overlap of the image patches.

are registered by applying the transformation that maps the keypoints from one image to the corresponding keypoints from the other image. Another common method for image registration is to minimize the dissimilarity of two superimposed images. When two images of the same scene are perfectly aligned, the intensity values at corresponding locations should be similar, and hence, the dissimilarity of the images is minimal. The local minimum can be found using a *gradient descent* algorithm.

Given the recent success of end-to-end deep learning approaches, it is surprising that those two methods are more commonly mentioned than any learning-based approach. To this day, there is an excessive use of handcrafted features available in public libraries like *OpenCV* [49], and keypoint extraction methods, e.g. SIFT [46], are still considered state-of-the-art and serve as strong baselines [68]. Since deep learning approaches improved performance in most other computer vision problems [50], it seems natural that it also had a positive impact on image registration. This assumption is supported by a growing interest in learning-based image registration methods, as can be seen in Figure 1.2. This thesis aims to verify if our belief is true or if the use of handcrafted methods is still rational in the context of image registration. In the process of doing so, we present a novel, fully learned keypoint extraction method and evaluate traditional and learning-based image registration approaches in a multitude of experiments. To be more precise, our main contributions in this thesis are as follows:

- We survey methods and the history of image registration to put this thesis into a context.
- We establish a new benchmark for image registration algorithms by providing a novel metric and two datasets consisting of one real and one synthetic dataset.

- We compare different image registration approaches and outline their advantages and drawbacks. Here, we focus on learning-based approaches to verify if recent success in deep learning had a positive impact on image registration. Furthermore, we account for a potential lack of annotated data by evaluating methods in a self-supervised domain adaption setting.
- We propose *C-RANSAC* as a method to incorporate prior knowledge about the target homography of an image registration problem into *RANSAC*. Our experiments show that even simple constraints can increase performance significantly.
- We propose a novel local keypoint extraction algorithm *DeCo-Net*. To our knowledge, it is the first fully learned detect-then-describe method that decouples the detector from the descriptor by redefining keypoints as points with a discriminative local geometry. Furthermore, we present a novel keypoint descriptor where a keypoint feature is obtained by sampling a dense feature map at locations encoded in the local geometry of the keypoint. The sampling method is inspired by *ROIAlign* implementations of *Faster RCNN* [29], however, unlike those methods we allow for more flexible modeling of local geometries.

To thoroughly investigate the effects of deep learning on image registration, we structure this thesis as follows: In Chapter 2 we provide needed background knowledge by surveying image registration, its history, and existing methods used as baseline methods in this thesis. Chapter 3 introduces the novel methods developed or adapted particularly for this thesis. This includes an algorithm to improve the performance of *RANSAC*, a novel learned keypoint extraction method, and an adapted synthetic dataset generation pipeline. In Chapter 4 we evaluate our proposed and baseline methods in a variety of different experiments. Furthermore, we give details about the implementation and specify hyperparameters to ensure reproducibility of our results. Finally, Chapter 5 summarizes and discusses our main findings and provides ideas for possible next steps.

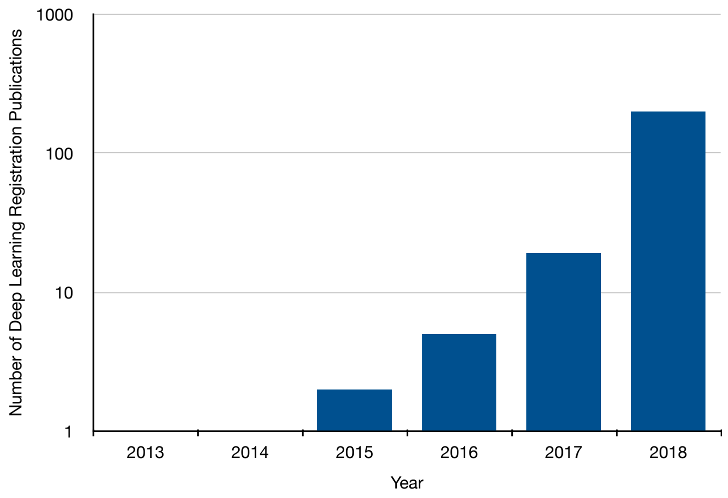


Figure 1.2: Overview of the number of learning-based image registration publications over the years 2013 to 2018 [26]. The seemingly exponential growth shows a rapidly growing interest.

2 Background and Related Work

Image registration, i.e. the process of transforming two images into the same coordinate system, is a fundamental problem in computer vision. The following chapter provides a framework for classifying image registration methods, surveys prominent image registration methods with a particular focus on learning-based approaches, introduces algorithms used in our experiments, and gives a theoretical background to concepts that serve as a basis for the methods described in Chapter 3.

2.1 Image Registration

Image registration denotes the process of finding a transformation T that maps the coordinate system of a source image I_s to the coordinate system of a target image I_t such that $I_t(\boldsymbol{x}) = I_s(T(\boldsymbol{x}))$, with $\boldsymbol{x}$ denoting any point (u, v) in the overlap, i.e. the area with information present in both I_s and I_t. An example of image registration can be seen in Figure 2.1.

Since there is a multitude of different approaches to image registration and the images to be aligned can have different modalities, resulting in different requirements for the needed transformation type, this section aims to shed some light on the topic of image registration. Additionally, we will state assumptions that will apply throughout the remainder of this thesis to simplify the problem at hand.

Image registration methods can be classified by the following three properties:

- Firstly, by the class of problem the images to be aligned fall into. Brown et al. propose the following classes [8]: *Multimodal Registration* denotes the registration of images from the same scene that are captured with different sensors, e.g. X-ray and MRI. *Template Registration* encompasses the class of problems where a template image is matched to a scene, as in atlas registration. *Viewpoint Registration* is the registration of images of the same scene taken from different viewpoints. *Temporal Registration* denotes the registration of images from the same scene taken at different time points or under different conditions.
- Secondly, by the transformation model that is used to register the source image space to the target image space [8]. The first broad class of transformations are projective transformations that map lines to lines and include translation, scaling, rotation, and other affine transformations. Since projective transformations cannot model local distortions, the second class of

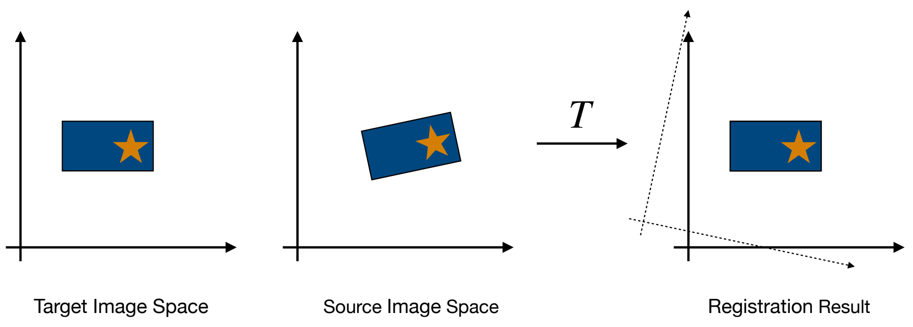


Figure 2.1: Two images with different orientations that are registered into the same coordinate system. Left is the target image, in the middle the source image and on the right the result after applying the transformation T to the source image. Note that the position of the target image remains unchanged.

transformations are more complex transformation models that allow to align elastic objects. An example of such are radial basis functions [69].

- Thirdly, image registration methods can be categorized into intensity-based approaches and feature-based approaches [24, 76]. Intensity-based algorithms aim to minimize a similarity measure of superimposed images. A simple yet prominent setting is the pixelwise *mean squared error (MSE)* metric and *gradient descent* for optimization. Let $\boldsymbol{\mu}$ be the transformation parameters that describe the transformation from the source image I_s to the target image I_t. We aim to find the transformation $T_{\boldsymbol{\mu}}$ with parameters $\boldsymbol{\mu}$ that correctly aligns I_s to I_t. The problem of finding this transformation can be formulated as a minimization problem with

$$\hat{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{\mu}} \mathcal{L}(\boldsymbol{\mu}, I_t, I_s),$$

where the loss $\mathcal{L}$ is given by the used metric, and $\hat{\boldsymbol{\mu}}$ are the parameters that correctly align the two images. Using the *MSE* metric, the loss can be calculated as

$$\mathcal{L}(\boldsymbol{\mu}, I_t, I_s) = \frac{1}{WH} \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} \left(I_t((u,v)) - I_s(T_{\boldsymbol{\mu}}((u,v)))\right)^2,$$

with W and H denoting the image width and height. To find the parameters $\hat{\boldsymbol{\mu}}$ that minimize the loss $\mathcal{L}$, a *gradient descent* algorithm is used. The algorithm computes the direction of the derivative of the loss function at its current point and takes a step towards that direction. Repeating this process iteratively will eventually find a local minimum of the function. The updated parameters can be calculated by

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i - \alpha \nabla \mathcal{L}(\boldsymbol{\mu}_i),$$

with α denoting the step size and i the iteration [31].
Feature-based algorithms, on the other hand, find distinctive points in the images, so-called keypoints, to compute a transformation that maps the points from the source image space

to the corresponding points in the target image space. To find keypoint correspondences, feature descriptors are extracted alongside the keypoints. Ideally, the distance of the features for corresponding keypoints from the two images is close while the distance to all other descriptors is high. Thus, the descriptor must be discriminative and repeatable considering the existing transformation [31].

For the remainder of this thesis, we will assume planarity and rigidity of the images to be aligned and that they fall into the class of *Viewpoint Registration*. This way, we only need to consider projective transformations and can simplify the transformation T to a homography $\boldsymbol{H} \in \mathbb{R}^{3\times 3}$.

Intensity-based image registration goes back at least to 1970 where image correlations were used for translational image registration [1]. Eleven years later Moravec et al. proposed the first local interest point algorithm for image matching [57]. In 2004, David G. Lowe published the *Scale Invariant Feature Transform (SIFT)* algorithm [46], which is regarded as state-of-the-art for keypoint extraction to this day and described in Section 2.4.1. With *AlexNet's* [40] success in the *ImageNet Large Scale Visual Recognition Challenge* [67], in 2012, deep learning had a breakthrough in computer vision and a huge impact on the community. Since then, several image registration approaches utilizing deep learning have emerged, in particular in the context of medical imaging [26].
First, deep learning was used to increase the performance of iterative, intensity-based methods, by learning a similarity measure between multimodal images [9, 27, 70]. This modification can easily be incorporated into the intensity-based framework presented above by setting

$$\mathcal{L}(\boldsymbol{\mu}, I_t, I_s) = M(I_t, T_{\boldsymbol{\mu}}(I_s)),$$

with M being a learned metric. Simonovsky et al. use a convolutional neural network for M to estimate the dissimilarity between two images [70]. Their network takes two grayscale images, stacked along the channel dimension to produce a 2-channel 3D image where each channel represents a different modality as input, and outputs a heatmap that indicates the dissimilarity for each pixel in the images. More precisely, the network learns to classify patch pairs X_i as correctly aligned (labeled $y_i = -1$) or not aligned ($y_i = 1$). To do so, a dataset with correctly aligned pairs of training images is augmented with random transformations to produce incorrect alignments. At train time, correct and incorrect alignments are sampled with equal probability and are fed into the neural network, which is then trained to minimize the hinge loss of the predicted and true labels. Later, the problem of image registration was formulated as a reinforcement learning problem [43, 47, 53]. According to Ma et al., in deep reinforcement learning, the environment E is organized as a stochastic finite state machine that takes an agent's action as input and outputs states and rewards [47]. Here, the agent can only observe states and rewards, and thus, has no knowledge about the internal model of the environment. Ma et al. propose a method for 3D image registration but for the sake of understandability, we will simplify their approach to the case of 2D image registration. A state $\boldsymbol{s}$ in E is represented as a tensor consisting of the two images to align. An action a is a transformation that is applied to the source image. The reward is designed to reflect the value of an action a_t of the agent in the current state $\boldsymbol{s}_t$. If the action results in a state that is closer to the ground truth state than the previous state, the reward is positive. If the opposite is the case, the reward is negative. For explorational steps, the magnitude of the reward is smaller compared to the reward triggered by the correct state. The agent is derived from the *dueling network* [80] and trained to approximate the optimal action-value function by maximizing the cumulative future reward [55]. The need for faster approaches motivated the emergence of end-to-end transformation estimation techniques, which take two images as input and directly output the estimated homography [14, 61]. Approach [14] proposed by DeTone et al. is described in more detail in Section 2.4.2. The lack of annotated data motivated the development of unsupervised methods [79, 42, 60]. The unsupervised image registration framework [79] proposed by De Vos et al. shares many similarities with conventional iterative intensity-based image registration and end-to-end transformation estimation techniques [14]. However, instead of directly optimizing the transformation parameters as in iterative approaches, the transformation parameters are optimized indirectly by optimizing the convolutional neural network parameters. At train time, the network takes two images, in a siamese [7] manner, as input, and outputs the transformation parameters to align the two images. The estimated parameters can then be used to transform the source image and compute the similarity between the target image and the transformed source image. The loss must be backpropagated through the transformation, and hence, a differentiable module like *Spatial Transformer Networks* [35] must be used to transform the source image. Since the previously mentioned unsupervised image registration approaches come with the familiar problem of quantifying image similarity [30, 78], more recent approaches try to address this problem by using weakly-supervised methods that utilize sparse labeling of anatomical structures [32] and *generative adversarial network (GAN)* [23] like frameworks [20]. In the method proposed by Fan et al., a *GAN* like architecture is used to combine end-to-end transformation estimation with learned similarity measures [20]. As in the traditional *GAN* [23] framework, they use two counterparts: A registration network R and a discrimination network D. D aims to determine if an input image pair is registered correctly (P^+) or not (P^-). To do so, D is trained to minimize the loss

$$\mathcal{L}_D(p) = \begin{cases} \log(1-p), & p \in P^+ \\ \log(p), & p \in P^- \end{cases},$$

where p is the output of D indicating the probability of a correct registration. R, on the other hand, aims to predict transformation parameters that trick D into thinking the registration is correct. Consequentially, R is supervised by the image similarity learned from D and its loss function can be formulated as

$$\mathcal{L}_R(p) = \log(1-p), \;\; p \in P^-.$$

D and R are trained in alternating order and convergence occurs when D cannot discriminate between positive and negative cases [20]. In feature-based image registration deep learning was first used to learn descriptors for image patches [81]. Later, the full keypoint detection and description pipeline was learned in a supervised setting as described in Section 2.4.3 [68, 63, 15, 18]. Figure 2.2 shows the chronology of important image registration branches. In this thesis, we concentrate on supervised transformation estimation and learned keypoint estimation. More detail on some of the above-mentioned methods is given in the following sections.

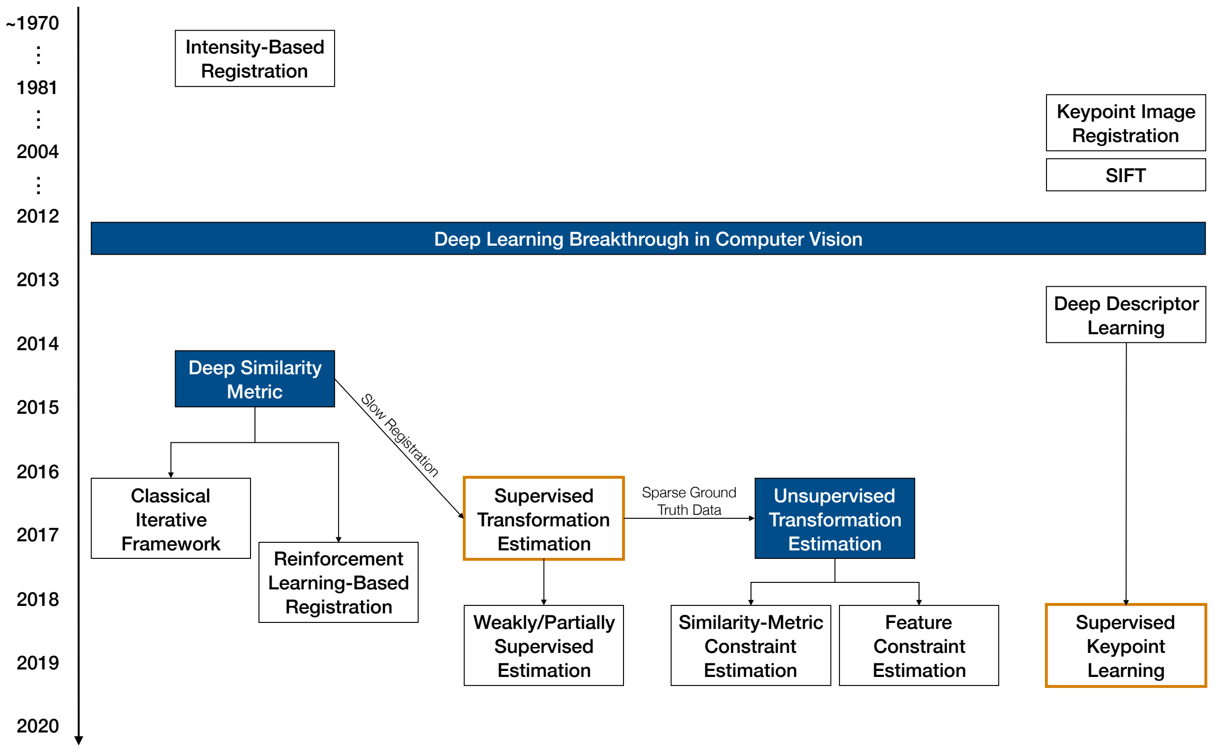

~1970
1981
2004
2012
2013
2014
2015
2016
2017
2018
2019
2020
Intensity-Based Registration
Keypoint Image Registration
SIFT
Deep Learning Breakthrough in Computer Vision
Deep Descriptor Learning
Deep Similarity Metric
Slow Registration
Classical Iterative Framework
Reinforcement Learning-Based Registration
Supervised Transformation Estimation
Sparse Ground Truth Data
Unsupervised Transformation Estimation
Weakly/Partially Supervised Estimation
Similarity-Metric Constraint Estimation
Feature Constraint Estimation
Supervised Keypoint Learning

Figure 2.2: Chronology of important milestones in image registration research (inspired by [26]). The boxes shaded in blue denote abstract classes while the orange outline indicates approaches examined in this work.

2.2 Direct Linear Transform (DLT)

The *DLT* algorithm can be used to compute a transformation matrix $\boldsymbol{H}$ from a sufficient set of point correspondences [74] and is used in this thesis to estimate a homography from keypoint-based approaches. The relation of two corresponding points $\boldsymbol{x} = (u, v, 1)^{\boldsymbol{T}}$ and $\boldsymbol{x}' = (s, t, 1)^{\boldsymbol{T}}$ in homogeneous coordinates can be expressed as

$$c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \boldsymbol{H} \begin{pmatrix} s \\ t \\ 1 \end{pmatrix}, \tag{2.1}$$

where c is a non-zero constant and

$$\boldsymbol{H} = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix}.$$

Dividing the first and second row of Equation 2.1 by the third row we get

$$-h_1 s - h_2 t - h_3 + (h_7 s + h_8 t + h_9) u = 0,$$

and

$$-h_4 s - h_5 t - h_6 + (h_7 s + h_8 t + h_9) v = 0,$$

which can be rewritten as

$$\boldsymbol{A}_i \boldsymbol{h} = 0,$$

with

$$\boldsymbol{A}_i = \begin{pmatrix} -x & -y & -1 & 0 & 0 & 0 & ux & uy & u \\ 0 & 0 & 0 & -x & -y & -1 & vx & vy & v \end{pmatrix}$$

and

$$\boldsymbol{h} = \begin{pmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{pmatrix}^T.$$

For each point correspondence, we get two equations, and thus, four correspondences are sufficient to solve for the eight degrees of freedom of a full homography, given that no three points are collinear. Four $\boldsymbol{A}_i \in \mathbb{R}^{2\times 9}$ matrices can be stacked row wise to get a single matrix $\boldsymbol{A} \in \mathbb{R}^{8\times 9}$. $\boldsymbol{A}\boldsymbol{h} = 0$ can then be solved to obtain $\boldsymbol{h}$ [17].

2.3 Random Sample Consensus (RANSAC)

Fischler and Bolles proposed *RANSAC* as an iterative method to fit a mathematical model to data containing outliers [21]. The algorithm does so by randomly sampling a subset of the data multiple times. Under the assumption that the data contains a sufficient amount of inliers to estimate the model parameters and that we repeat the process sufficiently often, the algorithm will find a subset of inliers that serve as data points for the parameter estimation. In the context of image registration, *RANSAC* can be used to estimate a homography from a set of matched keypoints from two images. Naturally, the keypoint detection algorithm will find true and false matches in the two images, and thus, one cannot estimate a homography using all matches. Using *RANSAC* allows to estimate a homography from a subset of correspondences that are likely to be correct matches. *RANSAC* can be divided into two steps that are iteratively repeated. First, a minimal amount of data to estimate the model parameters is randomly sampled from the dataset and the model is estimated using this subset. Secondly, the estimated model is evaluated on the entire dataset and the total number of inliers is computed. A data point is considered an inlier if it fits the model with some error thresholded by $\tau \in \mathbb{R}$. Those two steps are repeated until the model exceeds a predefined amount of inliers or a predefined number of iterations. Finally, the model that has led to the most inliers is returned. In an optional step, the model parameters may be re-estimated using all inliers under the best model. A more detailed description of the *RANSAC* algorithm is given in Section 3.2, where we present our improved *C-RANSAC* method.

2.4 Approaches in Detail

This section presents the methods evaluated in Chapter 4, namely *SIFT* [46], *Deep Image Homography Estimation* [14], and *RF-Net* [68]. Additionally, a short overview of other learned keypoint extraction methods is given.

2.4.1 Scale-Invariant Feature Transform (SIFT)

The *SIFT* algorithm [46] is used to first detect and then describe local interest points in images. For keypoint detection, the local extrema of the *Difference of Gaussians (DoG)* that occur at multiple scales are computed. Let $L(x, y, \sigma_i)$ be the convolution of the image $I(x, y)$ with the Gaussian kernel $G(x, y, \sigma_i)$ at scale σ_i:

$$L(x, y, \sigma_i) = G(x, y, \sigma_i) * I(x, y).$$

The *DoG* image $D(x, y, \sigma_i)$ can then be formulated as

$$D(x, y, \sigma_i) = L(x, y, \sigma_i) - L(x, y, \sigma_{i+1}).$$

Since the local extrema detection produces potentially unstable keypoint candidates, the algorithm rejects all candidates with low contrast or poor localization along an edge. Next, one or more orientations are computed for each keypoint based on local image gradient directions. This is an essential step to achieve invariance to rotation since the keypoint descriptor is computed relative to the orientation. For a Gaussian smoothed image sample $L(x, y, \sigma_i)$ at scale σ_i, the gradient magnitude $m(x, y, \sigma_i)$ and orientation $\theta(x, y, \sigma_i)$ can be computed, using pixel differences, by

$$\begin{aligned} m(x, y, \sigma_i) &= \sqrt{(L(x+1, y, \sigma_i) + L(x-1, y, \sigma_i))^2 + (L(x, y+1, \sigma_i) + L(x, y-1, \sigma_i))^2}, \\ \theta(x, y, \sigma_i) &= atan2\left(L(x, y+1, \sigma_i) - L(x, y-1, \sigma_i), L(x+1, y, \sigma_i) - L(x-1, y, \sigma_i)\right). \end{aligned}$$

The gradient magnitude and direction is computed for each pixel in a 16×16 pixel neighborhood around a keypoint. Finally, the keypoint descriptor is formed by weighting the previously computed gradients with a Gaussian window centered in the keypoint and accumulating them into orientation histograms summarizing the contents over 4×4 pixel subregions. The computed keypoint descriptors are highly distinctive and partly invariant to rotation, scale, and other possible variations [46].

2.4.2 Deep Image Homography Estimation (DIHE)

DeTone et al. have been the first to introduce a fully learned end-to-end approach for homography estimation [14]. The network adapted to our input data size is illustrated in Figure 2.3. It takes a two-channel grayscale image sized $W \times H \times 2$ that consists of two images related by a homography as input, and outputs the eight parameters of a homography. The conventional way to parameterize a homography is a 3×3 matrix with a fixed scale h_9. The homography maps points $(s, t, 1)^{\boldsymbol{T}}$ in the source image, to points $(u, v, 1)^{\boldsymbol{T}}$ in the target image, with

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix} \begin{pmatrix} s \\ t \\ 1 \end{pmatrix}.$$

Here, the submatrix $[[h_1, h_2], [\, h_4, h_5]]^{\boldsymbol{T}}$ represents, among other things, a rotational term, while the vector $[h_3,\ h_6]^{\boldsymbol{T}}$ is the translational offset. DeTone et al. found it hard to balance the rotational and translational terms as part of an optimization problem [14], and thus, proposed to use the

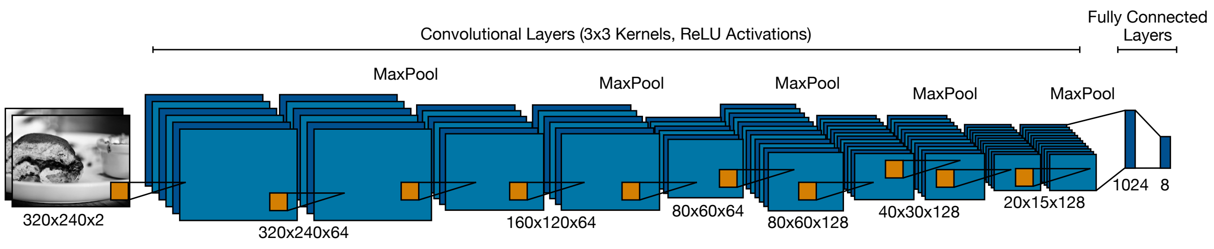


Figure 2.3: Architecture of the Deep Image Homography Estimation network implemented in this thesis. A VGG-style architecture with ten convolutional layers is used to estimate the eight parameters of a homography, given two images that are related by a homography [14].

4-point parameterization that has been used in traditional homography estimation methods [4]. For a more general notation, we extend the 4-point parameterization and introduce the n-point parametrization with $n \in \{1, 2, 3, 4\}$. Let $\Delta u_i = u_i - x_i$ and $\Delta v_i = v_i - y_i$ denote the u- and v-offset of two corresponding points in the source image I_s and the transformed image $\boldsymbol{H}_{n\text{-}point} \circ I_s$, with $\circ$ denoting the warping of image I by applying a homography $\boldsymbol{H}$ and

$$\boldsymbol{H}_{n\text{-}point} = \begin{pmatrix} \Delta u_1 & \Delta v_1 \\ \vdots & \vdots \\ \Delta u_n & \Delta v_n \end{pmatrix}.$$

Given the points in I_s and the offsets in $\boldsymbol{H}_{n\text{-}point}$, one can obtain the corresponding points in $\boldsymbol{H}_{n\text{-}point} \circ I_s$ to compute a homography using the *DLT* algorithm [25]. The n-point parameterization uses $2n$ parameters describing n point correspondences. Hence, the 4-point parameterization can describe a full homography, the 3-point parameterization an affinity, the 2-point parametrization a similarity, and the 1-point parameterization a translation. By choosing an appropriate n for the parameterization, one can incorporate prior knowledge of the needed transformation into the training. Some datasets may only require a translation to align the images, so predicting a full homography potentially introduces errors by falsely rotating or shearing the images. The estimated homography $\hat{\boldsymbol{H}}_{n\text{-}point}$ is supervised by the *MSE* of the regressed parameters and the parameters of the ground truth homography $\boldsymbol{H}_{n\text{-}point}$. Using the n-point parameterization, this can be formulated as

$$\mathcal{L}(\boldsymbol{H}_{n\text{-}point}, \hat{\boldsymbol{H}}_{n\text{-}point}) = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{H}_{n\text{-}point} - \hat{\boldsymbol{H}}_{n\text{-}point})^2.$$

2.4.3 Learned Keypoints

LF-Net. Ono et al. were the first to propose a fully learned keypoint extraction pipeline that could learn meaningful features from data alone [63]. The model consists of two components: A detector that detects keypoints and a descriptor that produces a feature for each keypoint. At train time, the detector takes two images I_i and I_j that are related by a homography $\boldsymbol{H}^{ij} \in \mathbb{R}^{3\times3}$, such that

$I_j = \boldsymbol{H}^{ij} \circ I_i$, as input. Based on a *Siamese Network* [7] structure, one branch is used to generate the ground truth for the other branch, as illustrated in Figure 2.4. Each branch outputs three dense maps defining the keypoint saliency $\boldsymbol{S}$, scale $\boldsymbol{s}$, and orientation (rotation) $\boldsymbol{\theta}$. Since we want to compare corresponding keypoints in I_i and I_j, we denote $\hat{\boldsymbol{S}}_j = \boldsymbol{H}^{ij} \circ \boldsymbol{S}_i$ as the saliency map obtained from warping $\boldsymbol{S}_i$ to the coordinate system of I_j. Following the same logic, we denote $\hat{\boldsymbol{s}}_j = \boldsymbol{H}^{ij} \circ \boldsymbol{s}_i$ and $\hat{\boldsymbol{\theta}}_j = \boldsymbol{H}^{ij} \circ \boldsymbol{\theta}_i$ as the scale and orientation obtained from warping $\boldsymbol{s}_i$ and $\boldsymbol{\theta}_i$ to the coordinate system of I_j. The K largest values in $\boldsymbol{S}_i$ and $\hat{\boldsymbol{S}}_j$ define the locations $\boldsymbol{p}_i^k$, $\hat{\boldsymbol{p}}_j^k$ of K corresponding keypoints $\kappa_i^k = (\boldsymbol{p}_i^k, \boldsymbol{s}_i^k, \boldsymbol{\theta}_i^k, \boldsymbol{D}_i^k)$ and $\hat{\kappa}_j^k = (\hat{\boldsymbol{p}}_j^k, \hat{\boldsymbol{s}}_j^k, \hat{\boldsymbol{\theta}}_j^k, \hat{\boldsymbol{D}}_j^k)$, with $1 \leq k \leq K$ and $\boldsymbol{D}_i^k$, $\boldsymbol{D}_j^k$ being keypoint features. Repeatability of the selected keypoints from I_i, I_j is enforced by the image loss

$$\mathcal{L}_{im}(\boldsymbol{S}_i, \boldsymbol{S}_j) = |\boldsymbol{S}_i - g(w(\boldsymbol{S}_j))|^2,$$

with w being a warping module that projects $\boldsymbol{S}_j$ to $\boldsymbol{S}_i$ and g a non-maximum suppression followed by a Gaussian filter to produce a smoother target saliency. Additionally, the detector minimizes the pair loss

$$\mathcal{L}_{pair}(\boldsymbol{D}_i^k, \hat{\boldsymbol{D}}_j^k) = \frac{1}{K}\sum_k^K |\boldsymbol{D}_i^k - \hat{\boldsymbol{D}}_j^k|^2 \tag{2.2}$$

that enforces the features $\boldsymbol{D}_i^k$, $\hat{\boldsymbol{D}}_j^k$ of corresponding keypoints κ_i^k and $\hat{\kappa}_j^k$ to be close. The detected keypoints are used to extract image patches from the locations $\boldsymbol{p}_i^k$ and $\hat{\boldsymbol{p}}_j^k$ that are scaled and rotated according to $\boldsymbol{s}_i^k$, $\hat{\boldsymbol{s}}_j^k$ and $\boldsymbol{\theta}_i^k$, $\hat{\boldsymbol{\theta}}_j^k$. Those image patches are fed to the descriptor to obtain the feature vectors $\boldsymbol{D}_i^k$, $\hat{\boldsymbol{D}}_j^k$. The descriptor is supervised by a triplet loss

$$\mathcal{L}_{tri}(\boldsymbol{D}_i^k, \hat{\boldsymbol{D}}_j^k, \hat{\boldsymbol{D}}_j^{k'}) = \frac{1}{K}\sum_k^K max\left(0, |\boldsymbol{D}_i^k - \hat{\boldsymbol{D}}_j^k|^2 - |\boldsymbol{D}_i^k - \hat{\boldsymbol{D}}_j^{k'}|^2 + \gamma\right)$$

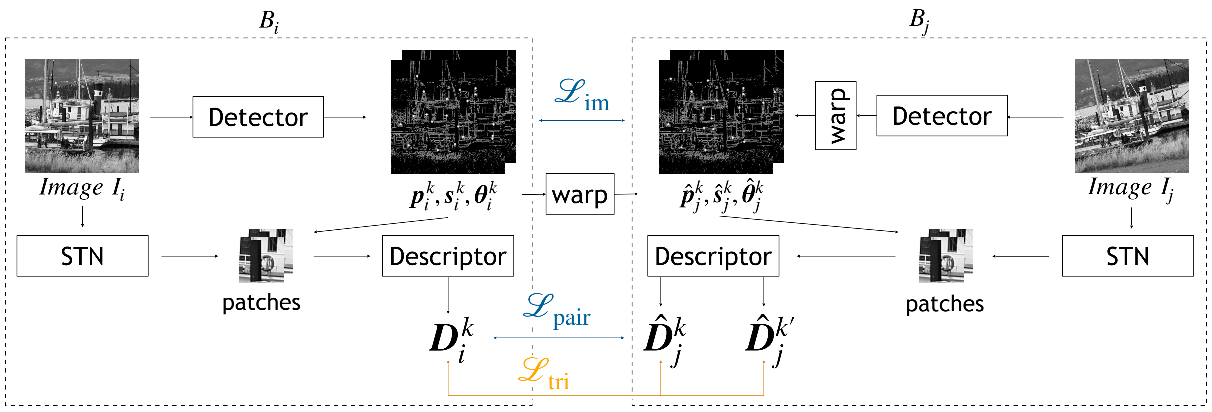
that minimizes the distance of features $\boldsymbol{D}_i^k$, $\hat{\boldsymbol{D}}_j^k$ from corresponding keypoints κ_i^k, $\hat{\kappa}_j^k$ and maximizes the distance between $\boldsymbol{D}_i^k$ and the feature $\hat{\boldsymbol{D}}_j^{k'}$ of the closest non-corresponding keypoint $\hat{\kappa}_j^{k'}$ from I_j, with γ being a margin between positive and negative pairs. Additionally, Ono et al. proposed to use a geometry loss for the detector to enforce geometrical consistency over the orientation of the detected and warped points with

$$\mathcal{L}_{geom}(\boldsymbol{s}_i^k, \boldsymbol{\theta}_i^k, \hat{\boldsymbol{s}}_j^k, \hat{\boldsymbol{\theta}}_j^k) = \lambda_{ori}\frac{1}{K}\sum_k^K |\boldsymbol{\theta}_i^k - \hat{\boldsymbol{\theta}}_j^k|^2 + \lambda_{scale}\frac{1}{K}\sum_k^K |\boldsymbol{s}_i^k - \hat{\boldsymbol{s}}_j^k|^2,$$

where λ_{ori}, λ_{scale} are weights. However, they report best numbers when omitting $\mathcal{L}_{geom}$. To detect keypoints at different scales, *LF-Net* resizes the same feature map, extracted from *ResNet* [28], to different resolutions. Here, each response in the abstract feature maps represents a high-level feature extracted from a large region in the image, while the low-level features are neglected [68].

RF-Net. Shen et al. proposed *RF-Net* which extends *LF-Net* by using a receptive field based detector [68]. This allows to extract low-level features at the first layers of a CNN and high-level features at the later layers, and thus, generates more effective scale-spaces and response maps. Additionally, they propose to remove spatially close keypoints from the triplet loss $\mathcal{L}_{tri}$ to stabilize training. Spatially close keypoints produce similar patches whose feature distances should not be maximized. Lastly, they successfully incorporate orientation (rotation) estimation that is indirectly supervised by the pair loss $\mathcal{L}_{pair}$ defined in Equation 2.2.

D^2-Net. Dusmanu et al. propose a describe-then-detect approach as opposed to the detect-then-describe approaches of previous methods [18]. They use a single network to produce a dense feature map $\boldsymbol{F} \in \mathbb{R}^{W\times H\times d}$ that plays a dual role. Firstly, this map yields features for each pixel in the image. Secondly, keypoints can be detected at local maxima in $\boldsymbol{F}$ along the spatial dimensions. This allows to eliminate the detector and locate keypoints based on high-level features instead of low-level image structures. The authors report increased stability at the cost of less well-localized keypoints.



B_i
B_j
Detector
Detector
Image I_i
Image I_j
$\boldsymbol{p}_i^k, \boldsymbol{s}_i^k, \boldsymbol{\theta}_i^k$
$\hat{\boldsymbol{p}}_j^k, \hat{\boldsymbol{s}}_j^k, \hat{\boldsymbol{\theta}}_j^k$
warp
warp
$\mathcal{L}_{\text{im}}$
STN
STN
patches
patches
Descriptor
Descriptor
$\boldsymbol{D}_i^k$
$\mathcal{L}_{\text{pair}}$
$\mathcal{L}_{\text{tri}}$
$\hat{\boldsymbol{D}}_j^k$
$\hat{\boldsymbol{D}}_j^{k'}$

Figure 2.4: Illustration of the LF-Net [63] training scheme. The network is trained by processing two images that are related by a homography with two identical instances of the network. The right branch B_j, starting from image I_j, is used to generate a supervision signal for the left branch B_i, starting from image I_i. By warping the results of B_i into the coordinate system of B_j corresponding image patches from I_i and I_j can be processed. As the warping is not differentiable, the authors optimize only over B_i and update the network copy for B_j in the next iteration. The blue losses belong to the detector loss while the orange triplet loss belongs to the descriptor.

3 Methods

The following chapter presents the methods specifically developed or adapted for this thesis. First, we will outline our data generation process which is closely related to [14] with the difference that we provide high-resolution image patches. Next, we will present *Constrained RANSAC*, a method to incorporate prior knowledge into *RANSAC* for homography estimation problems. Lastly, we will introduce our novel, fully learned keypoint extraction method *DeCo-Net*. Throughout this chapter, we keep the formulation of all algorithms general. Specific hyperparameter choices are stated in Chapter 4.

3.1 Synthetic Dataset

Since publicly available homography estimation datasets are sparse and data acquisition is expensive and time-consuming, we use a synthetically generated dataset. Using a synthetic dataset further allows us to remove potential noise, control the amount of distortion, and evaluate self-supervised training. However, synthetic datasets do not necessarily reflect all the challenges of a real dataset and should therefore be used with caution, especially during evaluation.
The dataset is generated by applying random projective transformations to images from *MS COCO 2017* [45]. Our method is similar to the method presented in *Deep Image Homography Estimation* [14] with the difference that we additionally provide high-resolution image patches. One sample of the dataset consists of two images I_i, I_j of size $W \times H$, the homography $\boldsymbol{H}^{ji} \in \mathbb{R}^{3\times 3}$ that maps I_j to I_i, and two high-resolution images I_i', I_j' of size $\lambda W \times \lambda H$, with $\lambda \geq 1$. The data generation process is illustrated in Figure 3.1 and described in the following. We start with a single image I' of size $\lambda(W + 2\rho) \times \lambda(H + 2\rho)$, with $\rho \in \mathbb{N}$ controlling the amount of distortion present in the dataset. We downscale the image by a factor of $\frac{1}{\lambda}$ to obtain I of size $W + 2\rho \times H + 2\rho$. Next, a patch I_i of size $W \times H$ is cropped from I at position $\boldsymbol{p} = (\rho, \rho)$. Then, the four vertices of I_i are randomly perturbed by values within the range $[-\rho, \rho]$. The four corresponding vertices define a homography $\boldsymbol{H}^{ji}$. The inverse of this homography $\boldsymbol{H}^{ij} = (\boldsymbol{H}^{ji})^{-1}$ is applied to I to produce the distorted image I_d. A second patch I_j is cropped from I_d at position $\boldsymbol{p}$. To obtain the high-resolution images, we first compute the homography $\boldsymbol{H}^{ij'} = \boldsymbol{H}^{s^{-1}} \boldsymbol{H}^{ij} \boldsymbol{H}^{s}$, with $\boldsymbol{H}^{s}$ being a scaling homography $[[\frac{1}{\lambda}, 0, 0], [0, \frac{1}{\lambda}, 0], [0, 0, 1]]^{\boldsymbol{T}}$. $\boldsymbol{H}^{ij'}$ is applied to I' to get I_d'. Analogously to before, patches I_i', I_j' of size $\lambda W \times \lambda H$ at position $(\lambda\rho, \lambda\rho)$ are cropped from I' and I_d'. Finally, one sample is constructed from I_i, I_j, I_i', I_j', and $\boldsymbol{H}^{ji}$. This data generation pipeline can be customized

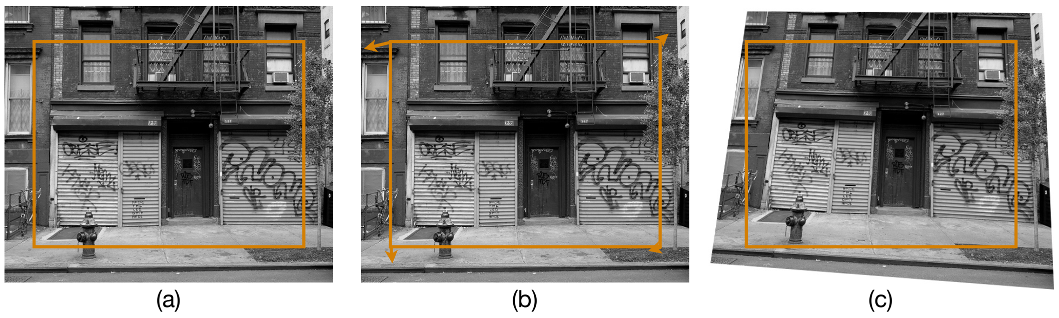


Figure 3.1: Illustration of the data generation pipeline for a low-resolution patch of the synthetic dataset. (a) A crop is taken from an image. (b) The vertices of the crop are randomly perturbed by values within the range $[-\rho, \rho]$. (c) The four resulting vertex correspondences are used to compute a homography $\boldsymbol{H}^{ji}$. The inverse of this homography $\boldsymbol{H}^{ij} = (\boldsymbol{H}^{ji})^{-1}$ is applied to the image and a crop is taken from the same position as before.

to incorporate specific transformations or visual artifacts like lightning changes. Example image pairs of size 320×240 from the dataset with $\rho \in \{16, 32, 64\}$ can be seen in Figure 4.1.

3.2 Constrained RANSAC (C-RANSAC)

Oftentimes, there might be prior knowledge about the target transformation in an image registration problem. For example, one could have a brief pre-alignment or know the maximum amount of distortion present in a dataset. Inspired from existing approaches that constrain the solution space of *RANSAC* [56, 12, 41], we propose a simple method to incorporate prior knowledge about the target transformation into *RANSAC*. Given that the constraint is reasonable, homographies computed from false positives can be eliminated, and thus, accuracy increases. Let I_i and I_j be two images of size $W \times H$ that are related by a homography $\boldsymbol{H} \in \mathbb{R}^{3\times3}$, such that $I_i = \boldsymbol{H} \circ I_j$, for which we aim to estimate a homography $\hat{\boldsymbol{H}} \in \mathbb{R}^{3\times3}$ that minimizes the *average projection error*

$$\texttt{APE}(\boldsymbol{H}, \hat{\boldsymbol{H}}) = \frac{1}{WH} \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} \sqrt{\left(\boldsymbol{H}\boldsymbol{H}^{-1}\boldsymbol{x}_{uv} - \hat{\boldsymbol{H}}\boldsymbol{H}^{-1}\boldsymbol{x}_{uv}\right)^2}, \tag{3.1}$$

with $\boldsymbol{x}_{uv}$ denoting the point $(u, v, 1)^{\boldsymbol{T}}$ in homogeneous coordinates. Further, let κ_i^k and κ_j^k, with $1 \leq k \leq K$, denote K keypoints extracted from I_i and I_j, and $C = \{(\kappa_i^k, \kappa_j^{n_k})\}$ a set of correspondences that assigns the closest keypoint $\kappa_j^{n_k}$ from κ_j^k to each keypoint in κ_i^k. In practice, C contains true and false correspondences, and hence, the objective of *(C-)RANSAC* is to find a subset $\hat{C}$ of true correspondences to estimate $\hat{\boldsymbol{H}}$. To do so, *(C-)RANSAC* keeps track of the largest set $\hat{C}$ of inliers that is produced from an estimated homography $\hat{\boldsymbol{H}}$, in an iterative manner. In one iteration t, *(C-)RANSAC* randomly samples four correspondences from C to compute a homography $\hat{\boldsymbol{H}}_t$ using the *DLT* algorithm introduced in Section 2.2. Next, the subset $\hat{C}_t \subseteq C$ of correspondences that are considered inliers under $\hat{\boldsymbol{H}}_t$ is computed by

$$\hat{C}_t = \{c \in C \mid \|c_i - \hat{\boldsymbol{H}}_t \circ c_j\| \leq \tau_{in}\},$$

with c_i and c_j denoting the I_i and I_j keypoint locations of a correspondence $(\kappa_i^k, \kappa_j^{n_k}) = c \in C$, $\circ$ denoting that a transformation is applied, and $\tau_{in} \in \mathbb{R}$ specifying how close two corresponding keypoints must be located to count as an inlier. In traditional *RANSAC*, the largest set of inliers $\hat{C}$ would be updated to $\hat{C}_t$ if $|\hat{C}_t| > |\hat{C}|$. In *C-RANSAC* it must additionally hold that the estimated homography is sufficiently similar to a given reference homography $\boldsymbol{H}_{ref} \in \mathbb{R}^{3\times3}$ that is a brief estimate of the correct transformation, i.e. $\texttt{APE}(\boldsymbol{H}_{ref}, \hat{\boldsymbol{H}}_t) \leq \tau_{ref}$, with $\tau_{ref} \in \mathbb{R}$ being a reference threshold that defines how much the estimated homography may differ from the reference homography. We use the *average projection error* instead of directly comparing the parameters of the homographies because the similarity of the parameters does not reflect the actual similarity of the transformation [14]. Consequently, the reference threshold τ_{ref} is specified in pixels. If $|\hat{C}_t| \geq \tau_n$, with $\tau_n \in \mathbb{N}$ denoting a sufficient amount of inliers, the loop is stopped early in iteration t. After finishing the loop, a final estimate $\hat{\boldsymbol{H}}$ is computed from all inliers in $\hat{C}$. If $\texttt{APE}(\boldsymbol{H}_{ref}, \hat{\boldsymbol{H}}) \geq \tau_{ref}$ the final estimate is rejected and the homography leading to the most inliers using only four correspondences is returned. If there are less than four correspondences or none of the estimated homographies met the conditions, the identity matrix is returned. The full algorithm is outlined in Algorithm 1. The constraint could, for example, be used to incorporate a preliminary estimate from *Deep Image Homography Estimation* [14] or by setting the reference homography to the identity map and the reference threshold to the maximum amount of distortion present in the dataset.

Algorithm 1 Constrained RANSAC

Pseudo code of our *Constrained RANSAC* implementation. The variable `corrs` denotes all found correspondences, `threshN` the amount of inliers needed to break the loop early, `Href` the reference homography, and `threshRef` the reference threshold.

```
function C-RANSAC(corrs, threshN, Href, threshRef)
    maxInliers = []
    finalH = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
    for i = 1 to N do
        fourCorrs = get4correspondences(corrs)
        H = computeH(fourCorrs)
        inliers = getInliers(corrs, H)
        if inliers.length > maxInliers.length then
            offset = APE(H, Href)
            if offset ≤ threshRef then
                maxInliers = inliers
                finalH = H
            end if
        end if
        if inliers.length ≥ threshN then
            break
        end if
    end for
    H = computeH(maxInliers)
    offset = APE(H, Href)
    if offset ≤ threshRef then
        finalH = H
    end if
    return finalH
end function
```

3.3 DeCo-Net: Deep Covariant Local Image Description

DeCo-Net is our novel method for local keypoint extraction, conceptually similar to the approaches introduced in Section 2.4.3 [68, 18, 63]. We aim to eliminate two downsides of existing approaches that utilize a detect-and-describe scheme. Firstly, we get rid of redundant descriptor computations from overlapping keypoint patches by densely extracting features for each pixel in the image. A keypoint feature is then formed by sampling features at specific locations proposed by the detector. This feature generation scheme reduces memory usage when increasing the number of keypoints. Secondly, we aim to make the training scheme less heuristic by decreasing the dependence between the detector and descriptor. The authors of *RF-Net* [68] report that the detector's training is greatly influenced by the descriptor's performance, and thus, propose to train the descriptor twice and the detector once in each training iteration. To reduce the dependence of the two parties, we define keypoints as locations with a discriminative local geometry instead of locations with a discriminative feature. Since the local geometry is estimated by the detector this reformulation allows to train the detector independently of the descriptor. Additionally, we allow for a more flexible modeling of local geometries compared to existing approaches that only model scale and rotation. A detailed illustration of *RF-Net* [68] to highlight our novelties is given in Figure 3.5.

3.3.1 Architecture

DeCo-Net consists of a detector that extracts keypoint locations and local geometries from an image and a descriptor that computes features for each keypoint. Our detector relies on two regressors derived from the same feature map $\boldsymbol{F}$ using two separate heads, as illustrated in Figure 3.2:

- The feature map $\boldsymbol{F} \in \mathbb{R}^{W \times H \times f}$ is produced by propagating the input image through a convolutional neural network.
- The keypoint location head consists of a single convolutional layer with one 1×1 kernel, followed by instance normalization [77] and a spatial softmax function. The regressor takes $\boldsymbol{F}$ as input and produces a saliency map $\boldsymbol{S} \in \mathbb{R}^{W \times H \times 1}$ with entries $\boldsymbol{S}^k$ for each pixel location $\boldsymbol{p}^k \in \mathbb{R}^{2 \times 1}$, with $k \in \mathbb{N}$ denoting the index of the location. Letting $\boldsymbol{S}'$ denote the output of the instance normalization and $\mathcal{N}_k$ denote the $n \times n$ spatial neighborhood of $\boldsymbol{p}^k$, the spatial softmax function can be written as

$$\boldsymbol{S}^k = \frac{\exp(\boldsymbol{S'}^k)}{\sum_{l \in \mathcal{N}_k} \exp(\boldsymbol{S'}^l)} \in [0, 1].$$

- The local geometry head takes $\boldsymbol{F}$ as input and consists of a convolutional neural network. The geometry regressor outputs $\boldsymbol{G} \in \mathbb{R}^{W \times H \times 8}$ containing, for each pixel location $\boldsymbol{p}^k$, a parametrization $\boldsymbol{G}^k \in \mathbb{R}^8$ of the local geometry of a region around $\boldsymbol{p}^k$ so that $\boldsymbol{G}^k$ can be mapped to a homography matrix. Inspired by DeTone et al. and illustrated in Figure 3.3 we use a point-based parametrization [14], where four points are placed in a canonical constellation $\boldsymbol{C} = [\boldsymbol{p}_l \in \mathbb{R}^2]_{l=1}^4$ around $\boldsymbol{p}^k$, and $\boldsymbol{G}^k$ encodes displacements relative to $\boldsymbol{C}$. We

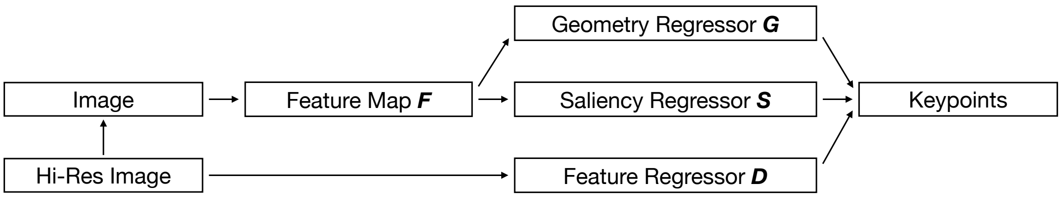


Figure 3.2: High-level illustration of our DeCo-Net architecture. The high-resolution image of size $\lambda W \times \lambda H$ is downscaled to $W \times H$. The downscaled image is propagated through a convolutional neural network to produce a feature map $\boldsymbol{F} \in \mathbb{R}^{W\times H\times f}$. From this feature map $\boldsymbol{F}$, two regressor heads output the saliency map $\boldsymbol{S} \in \mathbb{R}^{W\times H}$ and the local geometry map $\boldsymbol{G} \in \mathbb{R}^{W\times H\times 8}$. The feature regressor takes the high-resolution image as input and produces feature vectors $\boldsymbol{D} \in \mathbb{R}^{W\times H\times d}$. Finally, a keypoint is formed at locations $\boldsymbol{p}^k$ proposed by $\boldsymbol{S}$ with orientation $\boldsymbol{G}^k \in \boldsymbol{G}$ and descriptor $\boldsymbol{D}^k \in \boldsymbol{D}$.

use the four vertices of an axis-aligned square centered in $\boldsymbol{p}^k$ with an edge length of $2e$ as canonical constellation $\boldsymbol{C} = [\pm e, \pm e]^{\boldsymbol{T}} \in \mathbb{R}^{2\times 4}$. The point-based parametrization allows easy feature sampling as described in the following paragraph and makes it possible to constrain the transformation type to a sub-group of homographies by using $L \in \{1, 2, 3\}$ points for $\boldsymbol{C} = [\boldsymbol{p}_l \in \mathbb{R}^2]_{l=1}^{L}$.

The descriptor consists of a convolutional neural network that takes a high-resolution version of the input image of size $\lambda W \times \lambda H$ and produces a feature map $\boldsymbol{D} \in \mathbb{R}^{W\times H\times d}$. We use a high-resolution image to capture more image details and obtain more discriminative features. To form a feature $\boldsymbol{D}^k$ for a single keypoint $\kappa^k = (\boldsymbol{p}^k, \boldsymbol{G}^k, \boldsymbol{D}^k)$, $\boldsymbol{D}$ is sampled at five locations encoded in the keypoint location $\boldsymbol{p}^k$ and its local geometry $\boldsymbol{G}^k$. The five locations are the keypoint location itself and the offsets in $\boldsymbol{G}^k$ relative to the canonical sampling constellation around the keypoint. Formally, given a canonical sampling constellation

$$\boldsymbol{C} = [\pm e, \pm e]^{\boldsymbol{T}} \in \mathbb{R}^{2\times 4} \tag{3.2}$$

at location $\boldsymbol{p}^k \in \mathbb{R}^{2\times 1}$

$$\boldsymbol{C}' = [\ \boldsymbol{C} + \boldsymbol{p}^k\ ,\ \boldsymbol{p}^k\ ]^{\boldsymbol{T}} \in \mathbb{R}^{2\times 5},$$

we can apply a local geometry estimate $\boldsymbol{G}^k$ to compute the five sampling locations

$$\boldsymbol{C}^k = \boldsymbol{C}' + [\boldsymbol{G'}^k, [0, 0]^{\boldsymbol{T}}]^{\boldsymbol{T}} \in \mathbb{R}^{2\times 5}, \tag{3.3}$$

with $\boldsymbol{G'}^k \in \mathbb{R}^{2\times 4}$ being a reshape of $\boldsymbol{G}^k \in \mathbb{R}^8$. The actual feature $\boldsymbol{D}^k$ for a single keypoint κ^k at location $\boldsymbol{p}^k$ can then be computed by taking the mean of the five features obtained from sampling $\boldsymbol{D}$ at the locations in $\boldsymbol{C}^k$:

$$\boldsymbol{D}^k = \frac{1}{5}\sum_{n=1}^{5} b(\boldsymbol{D}, \boldsymbol{C}_n^k) \in \mathbb{R}^d, \tag{3.4}$$

where $b(\boldsymbol{M}, \boldsymbol{C}_n^k)$ denotes bilinear sampling of feature map $\boldsymbol{M}$ at location $(\boldsymbol{C}_{mn}^k)_{\substack{m\in\{1,2\}\\ n}} \in \mathbb{R}^2$.

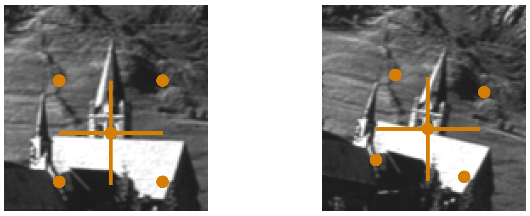


Figure 3.3: Two corresponding local geometries. The orange cross indicates the keypoint position while the orange dots indicate the associated local geometry. Left is the canonical constellation $\boldsymbol{C}$ and on the right the correct corresponding local geometry. Note how the points are at corresponding positions in the two images.

3.3.2 Training and Losses

Similar to *LF-Net* [63] and *RF-Net* [68], *DeCo-Net* is trained based on a *siamese network* structure [7] where one branch B_j generates the ground truth for the other branch B_i, as illustrated in Figure 3.4. At train time, *DeCo-Net* takes two images I_i, I_j of size $W \times H$, two homographies $\boldsymbol{H}^{ji}$, $\boldsymbol{H}^{ij}$ mapping I_j to I_i and I_i to I_j, respectively, and two high-resolution images I'_i, I'_j of size $\lambda W \times \lambda H$ as input. The detector takes I_i, I_j to produce the keypoint saliency maps $\boldsymbol{S}_i$, $\boldsymbol{S}_j$ and the local geometry maps $\boldsymbol{G}_i$, $\boldsymbol{G}_j$. Since we want to compare corresponding keypoints in I_i and I_j, we denote $\hat{\boldsymbol{S}}_j = \boldsymbol{H}^{ij} \circ \boldsymbol{S}_i$ as the saliency map obtained from warping $\boldsymbol{S}_i$ to the coordinate system of I_j. Following the same logic, we denote $\hat{\boldsymbol{G}}_j = \boldsymbol{H}^{ij} \circ \boldsymbol{G}_i$ as the local geometry map obtained from warping $\boldsymbol{G}_i$ to the coordinate system of I_j. Note that warping $\boldsymbol{G}_i$ consists of a global transformation, as for I_i and $\boldsymbol{S}_i$, followed by a local warping where the points encoded in $\boldsymbol{G}_i$ are mapped to corresponding points in I_j. The descriptor takes I'_i, I'_j to produce dense feature maps $\boldsymbol{D}_i$, $\boldsymbol{D}_j$. The K largest values in $\boldsymbol{S}_i$ and $\hat{\boldsymbol{S}}_j$ define the positions $\boldsymbol{p}_i^k$, $\hat{\boldsymbol{p}}_j^k$ of K corresponding keypoints in I_i and I_j, with $1 \leq k \leq K$. Using the keypoint positions $\boldsymbol{p}_i^k$ and local geometries $\boldsymbol{G}_i^k$ we can sample keypoint features $\boldsymbol{D}_i^k$ from $\boldsymbol{D}_i$, as described in Equation 3.3 and 3.4. Similarly, we can sample $\boldsymbol{D}_j$ at locations encoded in $\hat{\boldsymbol{p}}_j^k$, $\boldsymbol{G}_j^k$ and $\hat{\boldsymbol{p}}_j^k$, $\hat{\boldsymbol{G}}_j^k$ to obtain corresponding features $\hat{\boldsymbol{D}}_j^k$ and $\hat{\hat{\boldsymbol{D}}}_j^k$, respectively. Finally, we can form keypoints $\kappa_i^k = (\boldsymbol{p}_i^k, \boldsymbol{G}_i^k, \boldsymbol{D}_i^k)$, $\hat{\kappa}_j^k = (\hat{\boldsymbol{p}}_j^k, \boldsymbol{G}_j^k, \hat{\boldsymbol{D}}_j^k)$, and $\hat{\hat{\kappa}}_j^k = (\hat{\boldsymbol{p}}_j^k, \hat{\boldsymbol{G}}_j^k, \hat{\hat{\boldsymbol{D}}}_j^k)$ using the keypoint positions, local geometries, and features.
Repeatability of the K selected keypoints κ_i^k and κ_j^k, i.e. the positions of the K largest values in $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$, is enforced by the image loss

$$\mathcal{L}_{im}(\boldsymbol{S}_i, \boldsymbol{S}_j) = |\boldsymbol{S}_i - g(\boldsymbol{H}^{ji} \circ \boldsymbol{S}_j)|^2,$$

with g being a Gaussian filter to produce a smoother target saliency. The local geometries are trained to be covariant with the local transformations undergone by the region around the keypoint in the two images, giving *DeCo-Net* its name. In order to enforce this property, we need a way to compare two local geometries $\boldsymbol{G}_i^k$, $\boldsymbol{G}_j^k$ from I_i, I_j. Following the notation in Equation 3.2, we first let $\boldsymbol{C}_{1:4}^k = (\boldsymbol{C}_{mn}^k)_{\substack{m \in \{1,2\} \\ 1 \leq n \leq 4}}$ denote the four points in the constellation $\boldsymbol{C}^k$ that are not at location $\boldsymbol{p}^k$. Next, we let $r(\boldsymbol{C}^k)$ denote the minimum ℓ_1 distance of the points in the constellation $\boldsymbol{C}_{1:4}^k \in \mathbb{R}^{2\times 4}$

having entries c_{mn} to the centroid $\overline{\boldsymbol{C}^k_{1:4}} \in \mathbb{R}^2$ of the constellation:

$$r(\boldsymbol{C}^k) = \min_n(|c_{1n} - (\overline{\boldsymbol{C}^k_{1:4}})_1| + |c_{2n} - (\overline{\boldsymbol{C}^k_{1:4}})_2|),$$

where $\overline{\boldsymbol{C}^k_{1:4}} = (\frac{\sum_n^4 c_{1n}}{4}, \frac{\sum_n^4 c_{2n}}{4})$. Using this, we define a measure that allows us to compare points from two corresponding constellations $\boldsymbol{C}^k_i$ and $\boldsymbol{C}^k_j$ of κ^k_i and $\hat{\kappa}^k_j$ as follows:

$$h(\boldsymbol{C}^k_i, \boldsymbol{H}^{ji} \circ \boldsymbol{C}^k_j) = 1 - \frac{d\left(\boldsymbol{C}^k_i, \boldsymbol{H}^{ji} \circ \boldsymbol{C}^k_j\right)}{\left(r(\boldsymbol{C}^k_i) r(\boldsymbol{H}^{ji} \circ \boldsymbol{C}^k_j)\right)^{\frac{1}{p}} + \epsilon},$$

where $d(\cdot,\cdot)$ can denote the squared ℓ_2 distance ($p = 1$), or the (smooth) ℓ_1 distance ($p = 2$), ϵ is a small number ensuring that the division is valid, and $\boldsymbol{H}^{ji} \circ \boldsymbol{C}^k_j$ denotes the transformation of the constellation $\boldsymbol{C}^k_j$ into the coordinate system of I_i. This transformation is necessary to compare the constellations in the same coordinate system. Taking the p-th root ensures that the numerator and denominator both contain linear or squared measures of distance. The normalization in the denominator is important to avoid the trivial solution $\boldsymbol{C}^k_i = \boldsymbol{H}^{ji} \circ \boldsymbol{C}^k_j = [\boldsymbol{p}^k_i, \cdots, \boldsymbol{p}^k_i]^{\boldsymbol{T}} \in \mathbb{R}^{2\times5}$, where all points collapse to the origin $\boldsymbol{p}^k_i$. Further, let $c(\boldsymbol{G}, \boldsymbol{p}^k) = \boldsymbol{C}^k$ denote the constellation at $\boldsymbol{p}^k$ after applying the local geometry $\boldsymbol{G}^k$ as described in Equation 3.3. Using the above expressions, we define a loss that enforces region repeatability while jointly learning keypoint positions and local geometries as

$$\mathcal{L}_{geo}(\boldsymbol{S}_i, \boldsymbol{G}_i, \boldsymbol{G}_j) = -\frac{\sum_{\boldsymbol{p}^k_i \in \Omega} \boldsymbol{S}^k_i h(c(\boldsymbol{G}_i, \boldsymbol{p}^k_i), \boldsymbol{H}^{ji} \circ c(\boldsymbol{G}_j, \boldsymbol{H}^{ij} \circ \boldsymbol{p}^k_i))}{\sum_{\boldsymbol{p}^k_i \in \Omega} \boldsymbol{S}^k_i + \epsilon},$$

where $\boldsymbol{S}^k_i$ is the salience score at position $\boldsymbol{p}^k_i$ and Ω denotes the set of all locations $\boldsymbol{p}^k_i$ in the overlap of I_i and I_j, i.e. the area with information present in both I_i and I_j. The detector additionally minimizes the pair loss

$$\mathcal{L}_{pair}(\boldsymbol{D}^k_i, \hat{\boldsymbol{D}}^k_j) = \frac{1}{K}\sum_k^K |\boldsymbol{D}^k_i - \hat{\boldsymbol{D}}^k_j|^2$$

that enforces the features D^k_i, $\hat{D}^k_j$ of corresponding keypoints κ^k_i, $\hat{\kappa}^k_j$ to be close. Combining the three losses gives us the final detection loss

$$\mathcal{L}_{det} = \lambda_1 \mathcal{L}_{im} + \lambda_2 \mathcal{L}_{geo} + \lambda_3 \mathcal{L}_{pair},$$

with $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}$ being weight factors. The descriptor is supervised by a triplet loss $\mathcal{L}_{tri}$ that minimizes the distance of features $\boldsymbol{D}^k_i$, $\hat{\hat{\boldsymbol{D}}}^k_j$ of corresponding keypoints κ^k_i, $\hat{\hat{\kappa}}^k_j$ and maximizes the distance between $\boldsymbol{D}^k_i$ and the feature $\hat{\hat{\boldsymbol{D}}}^{k'}_j$ of the closest non-corresponding keypoint:

$$\mathcal{L}_{des} = \lambda_4 \mathcal{L}_{tri}(\boldsymbol{D}^k_i, \hat{\hat{\boldsymbol{D}}}^k_j, \hat{\hat{\boldsymbol{D}}}^{k'}_j) = \lambda_4 \frac{1}{K}\sum_k^K \max\left(0, \beta|\boldsymbol{D}^k_i - \hat{\hat{\boldsymbol{D}}}^k_j|^2 - |\boldsymbol{D}^k_i - \hat{\hat{\boldsymbol{D}}}^{k'}_j|^2 + \gamma\right),$$

with $\lambda_4 \in \mathbb{R}$ being a weight factor, $\beta \in \mathbb{R}$ controlling the weight of positive pairs and $\gamma \in \mathbb{R}$ being a margin between positive and negative pairs. Using the features $\hat{\hat{\boldsymbol{D}}}_j^k$ that have been computed from the warped geometry $\hat{\boldsymbol{G}}_j^k$ reduces the dependence on the actual geometry estimation of the detector, and therefore, stabilizes the training of the descriptor. To reduce label ambiguity we use the *neighbor mask algorithm* [68] where keypoint locations $\boldsymbol{p}^{k'}$ are only considered as non-corresponding if their Euclidean distance $e(\cdot,\cdot)$ to $\boldsymbol{p}^k$ is larger than τ_Q

$$e(\boldsymbol{p}^k, \boldsymbol{p}^{k'}) > \tau_Q.$$

Intuitively, if keypoint positions $\boldsymbol{p}^k$ and $\boldsymbol{p}^{k'}$ are very close in space we consider them as matching, and thus, do not want to maximize their feature distance.

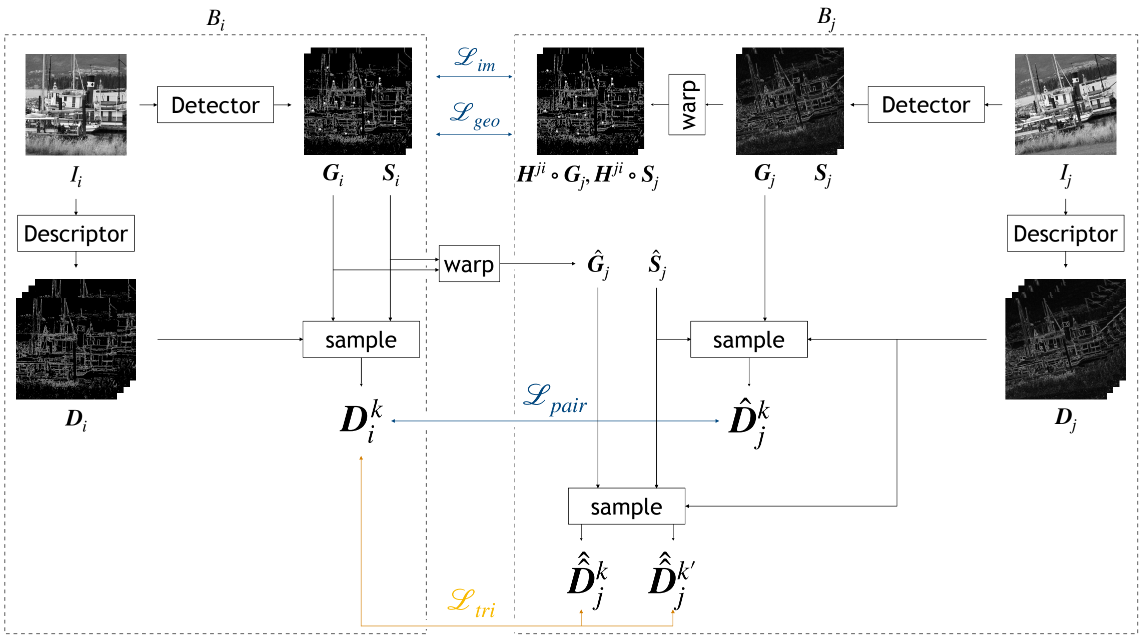


Figure 3.4: Illustration of our DeCo-Net training scheme. The network is trained by processing two images that are related by a homography with two identical instances of the network. The right branch B_j, starting from image I_j, is used to generate a supervision signal for the left branch B_i, starting from image I_i. By warping the regressions from B_i into the coordinate system of B_j, or vice versa, corresponding regressions from I_j and I_i can be compared. The blue losses belong to the detector while the orange triplet loss belongs to the descriptor.

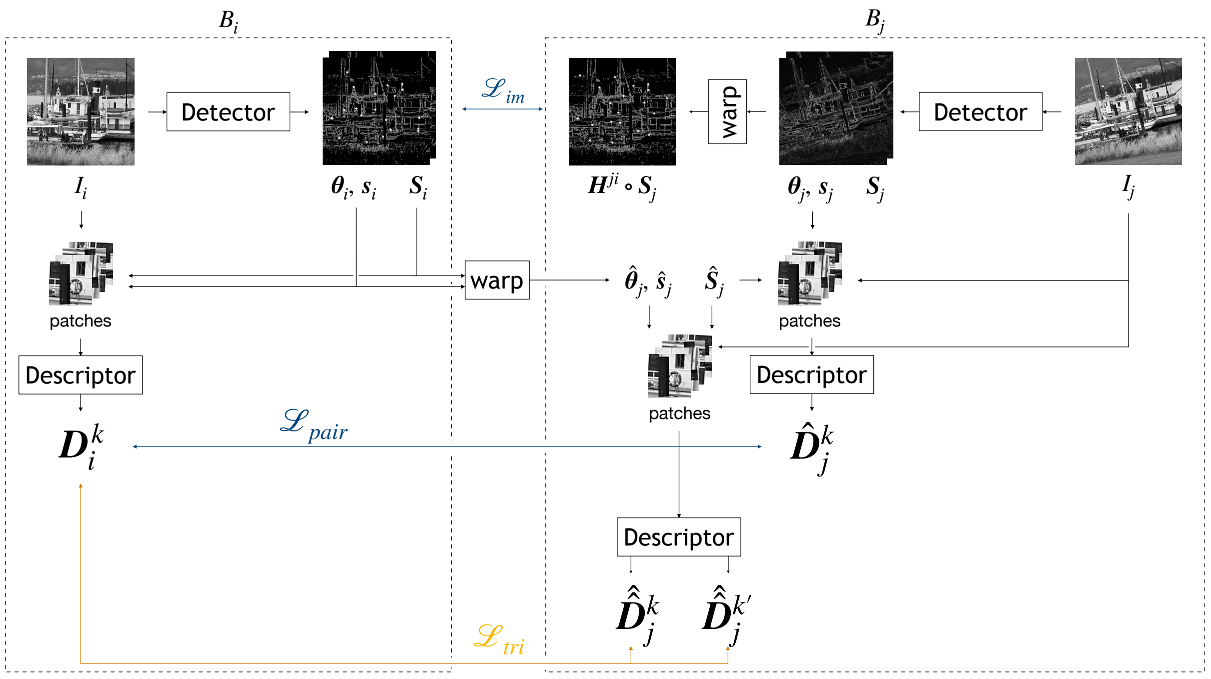


Figure 3.5: Illustration of the RF-Net [68] training scheme. The network is trained by processing two images that are related by a homography with two identical instances of the network. The right branch B_j, starting from image I_j, is used to generate a supervision signal for the left branch B_i, starting from image I_i. By warping the regressions from B_i into the coordinate system of B_j, or vice versa, corresponding regressions from I_j and I_i can be compared. The blue losses belong to the detector while the orange triplet loss belongs to the descriptor. In contrast to DeCo-Net, RF-Net uses normalized image patches to produce features for each keypoint, and thus, needs no feature sampling strategy. This comes at the cost of higher memory requirements when increasing the number of keypoints. Additionally, RF-Net uses a simpler geometry normalization that only accounts for scale s and rotation θ, and that affects the input of the descriptor instead of its output. Furthermore, there is no direct way to compare local geometries since RF-Net uses no geometry loss.

4 Experiments

In this chapter, we conduct an in-depth evaluation of *SIFT*, *DIHE*, *RF-Net*, and *DeCo-Net*. The algorithm's homography estimation performance is tested in different settings on two datasets. All models are trained on a GeForce RTX 2080 Ti [62] with 11GB memory. First, we give details about the implementation of the methods under inspection. Next, we present the used datasets and evaluation metrics. Afterwards, we report keypoint matching scores and homography projection errors in our default environment. Lastly, we conduct experiments to evaluate domain adaption, the dependence of detectors and descriptors, and *Constrained RANSAC*. To simplify notation we omit *(C-)RANSAC* when talking about a homography estimated from a keypoint-based approach and *(C-)RANSAC*. So instead of saying the homography was estimated from *DeCo-Net* + *(C-)RANSAC*, we simply say that the homography was estimated from *DeCo-Net*.

4.1 Implementation

This section aims to ensure the self-containment of the thesis. We state implementation details and specify hyperparameter choices to guarantee that all results are reproducible. Further, we concretize ideas that were kept abstract in previous chapters. All methods are implemented using *Python 3.6.9* and *PyTorch* [65]. *OpenCV 3.4.2* [49] is used to extract *SIFT* keypoints.

4.1.1 DIHE

Our implementation of *DIHE* takes a two-channel image of size $320 \times 240 \times 2$ as input and outputs the eight parameters (for $n = 4$ in the n-point parameterization) of a full homography. The network uses 3×3 convolutional blocks with *BatchNorm* [34] and *ReLU* activation functions [59], and is architecturally similar to *VGG Net* [37]. Ten convolutional layers are used with five max-pooling layers (2×2), as illustrated in Figure 2.3. The convolutional layers are followed by two fully connected layers with 1024 and eight nodes and a dropout layer with $p = 0.5$ after the last convolutional layer and the first fully connected layer. We found the training to be more stable if we constrain the output of the network and the ground truth homography parameters to be in the range of $[-1, 1]$. To do so, the estimated parameters are put into a *Tanh* activation function and the ground truth homography parameters are divided by the maximum amount of corner distortion d_{max} existent in the dataset. We set $d_{max} = 500$. For inference, the estimated parameters are multiplied by d_{max} to reflect the actual displacement. When randomly initialized, the network was not able to learn meaningful features for the more challenging datasets *HPatches* and *MS COCO 2017* with $\rho = 64$. To resolve this issue, the network was instead initialized with the model parameters obtained from the training on *MS COCO 2017* with $\rho = 32$. We train the network for 5760 epochs using a *stochastic gradient descent (SGD)* optimizer with an initial learning rate of 0.005, a momentum of 0.9, and a batch size of 32. After every third of the training, the learning rate is decreased by a factor of 0.1. The model parameters that performed best on the evaluation set are used for inference in the following experiments.

4.1.2 RF-Net

We keep *RF-Net* as it is described in [68]. All implementation details can be taken from the original paper with the exception that we set the triplet loss balancing factor $\beta = 0.8$ for *MS COCO 2017* and that the network is trained with a batch size of one and 512 keypoints for 200 epochs on *Hpatches* and for 100 epochs on the nearly two times larger *MS COCO 2017* datasets.

4.1.3 DeCo-Net

For a fair comparison, we keep the implementation of shared components of *DeCo-Net* and *RF-Net* [68] the same. Our detector relies on two regressors derived from the same feature map $\boldsymbol{F} \in \mathbb{R}^{W \times H \times 16}$ using two separate heads, as illustrated in Figure 3.2. The feature map $\boldsymbol{F}$ is produced by propagating the input image through ten convolutional layers with sixteen 3×3 kernels followed by an instance normalization [77] and leaky ReLU activations [48]. Additionally, shortcut connections [28] were added between all consecutive layers to simplify the training of the network [68]. The resulting feature map $\boldsymbol{F}$ has a receptive field of 21×21. The keypoint location head takes $\boldsymbol{F}$ as input and consists of a single convolutional layer with one 1×1 kernel followed by instance normalization [77], a spatial softmax function with a receptive field of 5×5, and a Gaussian blurring with a kernel size of 15×15 and $\sigma = 0.5$ in branch B_j. The local geometry head takes $\boldsymbol{F}$ as input and consists of five convolutional layers with 32, 64, 32, 16, and eight 3×3 kernels, each followed by batch normalization [34] and leaky ReLU activations [48]. The last layer is put into a *Tanh* activation function and multiplied by $\alpha = 5$. The canonical reference square for the local geometry estimation has an edge length of $2e$ with $e = 10$ for *HPatches* and $e = 5$ for *MS COCO 2017*. To compare local geometries we use the ℓ_1 distance for $d(\cdot, \cdot)$ and set $p = 2$. We set $\lambda_1 = 1.0$, $\lambda_2 = 0.1$, and $\lambda_3 = 1.0$ to weight the losses accordingly.
Our descriptor consists of seven convolutional layers with 32, 32, 64, 64, 128, and 128 3×3 kernels and 128 9×9 kernels in the last layer. Each layer is followed by batch normalization [34] and all but the last layer are followed by a ReLU [59] activation function. Layers three and five have a stride of two, such that the descriptor produces a feature map $\boldsymbol{D} \in \mathbb{R}^{W \times H \times 128}$ taking the original image of size $4W \times 4H$ as input. We set the triplet loss balancing factor $\beta = 1$ for *HPatches* and $\beta = 0.8$ for *MS COCO 2017* and the margin $\gamma = 1$. The neighbor mask threshold τ_Q is set to 10 and the loss is weighted with $\lambda_4 = 1.0$.
Training is augmented by exchanging branches B_i and B_j and repeating the process in the inverse direction. We train the network with a batch size of one and 128 keypoints for 200 epochs on

Hpatches and for 100 epochs on the nearly two times larger *MS COCO 2017* datasets. The training routine is copied from *RF-Net* [68]: In each epoch, the detector is trained once while the descriptor is trained twice. The detector uses an Adam [38] optimizer with an initial learning rate of 0.01 for *HPatches* and 0.1 for *MS COCO 2017*. The learning rate is decayed by a factor of 0.9 every five epochs. The descriptor uses an *SGD* optimizer with an initial learning rate of 1.0 for *HPatches* and 10.0 for *MS COCO 2017*. The learning rate is decayed by a factor of $1.0 - \frac{t}{230N}$, where t is the current step and N the total number of epochs [68]. The model parameters that performed best on the evaluation set are used for inference in the following experiments.

4.1.4 (C-)RANSAC

Our *(C-)RANSAC* implementation follows the algorithm outlined in Section 3.2. We compute a homography from point correspondences using the *DLT* algorithm. The resulting equation $\boldsymbol{Ah} = 0$ is solved using *singular value decomposition* [73]. The loop is executed 1000 times or stopped early if 62.5% of all correspondences are aligned correctly. Given our images of size 320×240, two corresponding points are considered correctly aligned if their Euclidean distance in the same coordinate system is below a 5-pixel threshold. The final homography $\hat{\boldsymbol{H}}$ that is computed from all inliers also needs to satisfy our constraint $\texttt{APE}(\hat{\boldsymbol{H}}, \boldsymbol{H}_{ref}) \leq \tau_{ref}$ with $\boldsymbol{H}_{ref}$ and τ_{ref} being the reference homography and reference threshold. If the condition is not met, the homography leading to the most inliers using only four correspondences is returned. If there are less than four correspondences or none of the computed homographies met the conditions, the identity matrix is returned. We set $\tau_{ref} = \infty$ when using regular *RANSAC*.

4.2 Datasets

Throughout this thesis, we will use two datasets. The first, *HPatches* [5], consists of 116 sequences of up to six images from the same scene that are related by a given homography. Here, 57 sequences contain only photometric changes while 59 sequences show geometric variations due to viewpoint changes. We restrict our attention to the 59 sequences showing geometric variations. Those sequences are split into a training set, evaluation set, and test set with 230, 30, and 30 samples, respectively.

The second synthetic dataset is obtained from *MS COCO 2017* [45] by applying the method described in Section 3.1. To find appropriate distortion values ρ for the synthetic dataset we conduct a brief qualitative study. We randomly pick an image of size 320×240 from *MS COCO 2017* and, following the process in Section 3.1, warp the image such that the vertices move $[[\rho, -\rho], [-\rho, -\rho], [\rho, \rho], [-\rho, \rho]]^{\boldsymbol{T}}$ with $\rho \in \{16, 32, 64\}$. Those distortions can be considered as examples with one of the most extreme distortions present in the synthetic dataset. The results can be seen in Figure 4.1. Visually, $\rho = 32$ yields a good amount of distortion for our experiments, $\rho = 16$ gives too little distortions, and $\rho = 64$ gives too extreme distortions. Consequently, we will use $\rho = 32$, and $\rho = 64$ to see how the methods under inspection perform on more difficult data. The training set consists of the first 20000 samples in the *MS COCO 2017* training split, while the

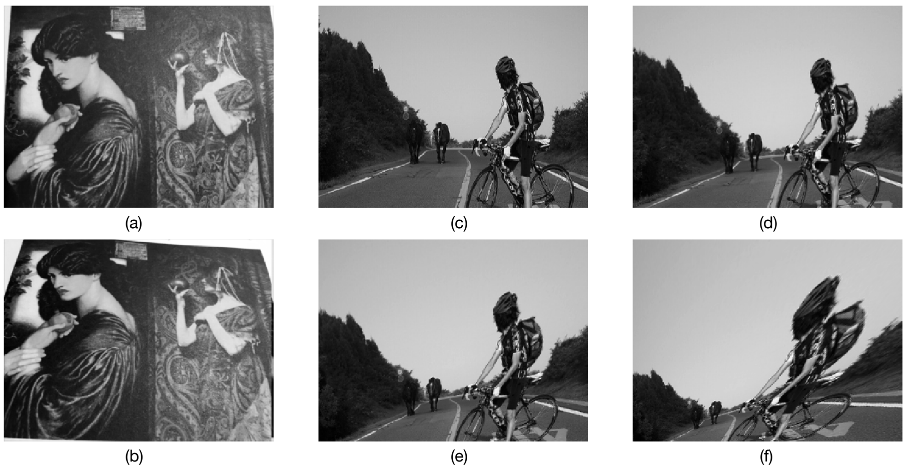


Figure 4.1: Example images from our datasets. (a) and (b) are a random sample from the HPatches [5] dataset. (c) is a randomly selected image from the MS COCO 2017 [45] dataset. (d), (e), and (f) are the same image warped such that the four vertices are shifted by $[[\rho, -\rho], [-\rho, -\rho], [\rho, \rho], [-\rho, \rho]]^{\boldsymbol{T}}$ with $\rho \in \{16, 32, 64\}$.

evaluation set consist of the first 200 and the test set of the next 2000 samples in the *MS COCO 2017* evaluation split. In each epoch, only 500 of the 20000 training samples are randomly selected to keep the number of samples that are processed per epoch similar to *HPatches*.

In both datasets, the images are converted to grayscale and normalized by subtracting the training set mean and dividing by the training set standard deviation. The image size of 320×240 is chosen as in *RF-Net* [68] and *LF-Net* [63], with the high-resolution images being four times larger, i.e. 1280×960. Example images from the two datasets can be seen in Figure 4.1.

4.3 Metrics

This section outlines the metrics used to evaluate the methods under inspection. First, we present metrics that can be used to compare keypoint estimates. Secondly, we provide metrics that can be used to evaluate homographies in the context of image registration. To our knowledge, the metrics for homography estimation have not been used in existing work, and thus, are a further contribution of this thesis.

4.3.1 Local Keypoint Estimation

For quantitative comparison of keypoint estimates, we report the precision, i.e. the proportion of correct matches. Following [54, 68] we use three different matching strategies to determine a match. Let κ_i^k and κ_j^k, with $1 \leq k \leq K$, denote K keypoints from two images I_i and I_j that are related by a homography $\boldsymbol{H}$, such that $I_i = \boldsymbol{H} \circ I_j$. Further, let κ_i^m and κ_j^n be two arbitrary keypoints from κ_i^k and κ_j^k at locations $\boldsymbol{p}_i^m$ and $\boldsymbol{p}_j^n$ with corresponding features $\boldsymbol{D}_i^m$ and $\boldsymbol{D}_j^n$. The different matching strategies can be formulated as follows:

- In *nearest neighbor (NN)* based matching, two keypoints κ_i^m and κ_j^n are matched if they have the closest descriptor features $\boldsymbol{D}_i^m$ and $\boldsymbol{D}_j^n$.
- In *nearest neighbor with a threshold (NNT)* based matching, two keypoints κ_i^m and κ_j^n are matched if they have the closest descriptor features $\boldsymbol{D}_i^m$ and $\boldsymbol{D}_j^n$ and their feature distance is below a threshold τ_d.
- In *nearest neighbor distance ratio (NNR)* based matching, two keypoints κ_i^m and κ_j^n are matched if $||\boldsymbol{D}_i^m - \boldsymbol{D}_j^n|| \ / \ ||\boldsymbol{D}_i^m - \boldsymbol{D}_j^o|| < \tau_r$, with $\boldsymbol{D}_j^n$ and $\boldsymbol{D}_j^o$ being the nearest and the second nearest neighbor to $\boldsymbol{D}_i^m$.

The three matching score metrics are combined into a single *mean matching score*

$$\texttt{MMS} = \frac{1}{N}\sum_{n=1}^{N}\frac{\texttt{NN} + \texttt{NNT} + \texttt{NNR}}{3},$$

with N denoting the dataset size. Contrary to the overlap measure used in [54], we consider two keypoints κ_i^m and κ_j^n as a true match if their distance $||\boldsymbol{p}_i^m - (\boldsymbol{H} \circ \boldsymbol{p}_j^n)||$ is below a 5-pixel threshold [68, 63, 66], given our images of size 320×240. Additionally, the learned descriptors are $L2$-normalized to ensure a fair comparison.

4.3.2 Homography Estimation

To compare an estimated homography $\hat{\boldsymbol{H}}$ to a ground truth homography $\boldsymbol{H}$ that maps image I_s to image I_t, we introduced the *average projection error* in Equation 3.1. This measure is more meaningful than directly comparing the parameters of the homographies because it reflects the actual similarity of the transformations applied to the images. The APE over the entire dataset, i.e. the *mean average projection error* can be computed by

$$\texttt{MAPE} = \frac{1}{N}\sum_{n=1}^{N}\texttt{APE}_n,$$

with n denoting the n-th image pair in the dataset. We found that a few, very extreme outliers can increase the MAPE strongly, so we exclude outliers and introduce the *thresholded mean average projection error*

$$\texttt{TMAPE} = \frac{1}{N-M}\sum_{n=1}^{N}\begin{cases}\texttt{APE}_n & \texttt{APE}_n \leq \tau \\ 0 & \text{else}\end{cases},$$

with M being the number of outliers for which $\texttt{APE}_n > \tau$. Further, we can compute the proportion of samples that were aligned correctly, i.e. has an APE less or equal than τ, as

$$\texttt{CorrH} = \frac{1}{N}\sum_{i=1}^{N}\left[\texttt{APE}_n \leq \tau\right].$$

The scores range between 0 and 1, with higher values being better.

4.4 Hyperparameter Search

Previous publications suggest that the performance of a deep learning algorithm is sensitive to the chosen hyperparameters [33, 82]. To find appropriate parameter settings for *RF-Net* and *DeCo-Net* we conducted a brief parameter search. Starting with the hyperparameters reported in [68] and additionally testing a smaller and larger value, will allow us to find potentially better setups and gives an estimate for the algorithm's stability in regard to the hyperparameter under inspection. For comparison, we report the *mean matching score* introduced in Section 4.3.1 on the evaluation sets with 128 keypoints.

4.4.1 Learning Rate and Number of Keypoints

			Learning Rate		
			0.01 & 1	**0.1 & 10**	**1.0 & 100**
DeCo-Net	**Keypoints**	**128**	0.526	0.526	0.464
		512	**0.561**	0.439	0.263
		2048	0.220	0.231	0.224
RF-Net		**128**	0.530	0.582	0.579
		512	**0.633**	0.630	0.627
		2048	N/A	N/A	N/A

Table 4.1: Mean matching scores on the evaluation set of HPatches for DeCo-Net and RF-Net with different learning rates and numbers of keypoints. N/A are setups that exceed our memory constraints.

First, we aim to find the optimal learning rate for our approaches. Since the batch size affects the quality of the computed gradients, and therefore, has an impact on the optimal learning rate, we test all combinations $(lr, \#\kappa)$ of learning rates $lr \in \{(0.01, 1), (0.1, 10), (1, 100)\}$ and numbers of keypoints $\#\kappa \in \{128, 512, 2048\}$. The learning rate tuple consists of the detector and descriptor learning rate. The *mean matching scores* for *HPatches* can be found in Table 4.1. We could not test *RF-Net* with 2048 keypoints since this setup exceeds our memory constraints. *RF-Net* achieves higher scores with more keypoints and has the best performance with a learning rate of $(0.01, 1)$. However, changing the learning rate affects the performance only marginally. *DeCo-Net*, on the other hand, is more sensitive to the learning rate, especially when training with 512 keypoints. Increasing the number of keypoints to 2048 decreases performance. The following argument suggests that the decrease in performance when increasing the number of keypoints is caused by poor keypoint localization. Let 2048 keypoints be arranged in a uniform grid on an image of size 320×240. The number of keypoints x per row and y per column can be computed by solving

$$xy = 2048 \quad \text{\textit{and}} \quad \frac{y}{x} = \frac{240}{320}, \tag{4.1}$$

resulting in $x \approx 53$ and $y \approx 39$. The distance d between two neighboring keypoints can then be computed by dividing the image edge lengths by x and y giving us $d \approx 6$. So, even in the case of a uniform distribution, there is a keypoint approximately every six pixels. Such a high density of keypoints implies that some of the keypoints are not located at meaningful positions or too close to each other, and thus, impair training. Reducing the number of keypoints to 128 decreases performance slightly but increases stability, and hence, will be used in all following experiments with a learning rate of $(0.1, 10)$.

4.4.2 Triplet Loss Balancing Factor

The triplet loss is known to suffer from slow convergence and poor local optima [52, 71]. In our keypoint-based methods, those poor local optima show by either producing very similar or very distinct features for all keypoints. To address this issue, we introduced the triplet loss balancing factor β that allows us to balance the impact of positive and negative tuples in the triplet loss. This way, we can control the loss values for the above-mentioned local optima, and thus, avoid them. If all features are similar, the positive tuples have too much impact, so β must be reduced. If all features are very distinct, the negative tuples have too much impact, so β must be increased. We test the performance of *DeCo-Net* and *RF-Net* on our synthetic *MS COCO 2017* dataset with $\rho \in \{32, 64\}$ for $\beta \in \{0.8, 1.0, 1.2\}$. The resulting *mean matching scores* on the evaluation set can be found in Table 4.2. Both methods achieve the highest performance for $\beta = 0.8$ with a significant lead over the second-best of up to 0.1, indicating that β must be chosen carefully.

		Balancing Factor		
		0.8	**1.0**	**1.2**
DeCo-Net	**MS COCO 32**	**0.40**	0.33	0.35
	MS COCO 64	**0.13**	0.12	0.12
RF-Net	**MS COCO 32**	**0.86**	0.79	0.78
	MS COCO 64	**0.81**	0.71	0.70

Table 4.2: Mean matching scores on the evaluation set of our synthetic MS COCO 2017 datasets for DeCo-Net and RF-Net with different balancing factors.

4.5 Keypoint Matching Scores

To compare the performance of the local keypoint estimation approaches, we report the matching scores introduced in Section 4.3.1 on the datasets presented in Section 4.2. We report performance on the test sets for *SIFT*, *RF-Net*, and *DeCo-Net* using 128 keypoints. For the matching score thresholds, we use $\tau_d = 1.0$ and $\tau_r = 0.7$. Further, we use the hyperparameters reported in Section 4.4 that achieved the best `MMS` on the evaluation sets. The results can be seen in Figure 4.2.

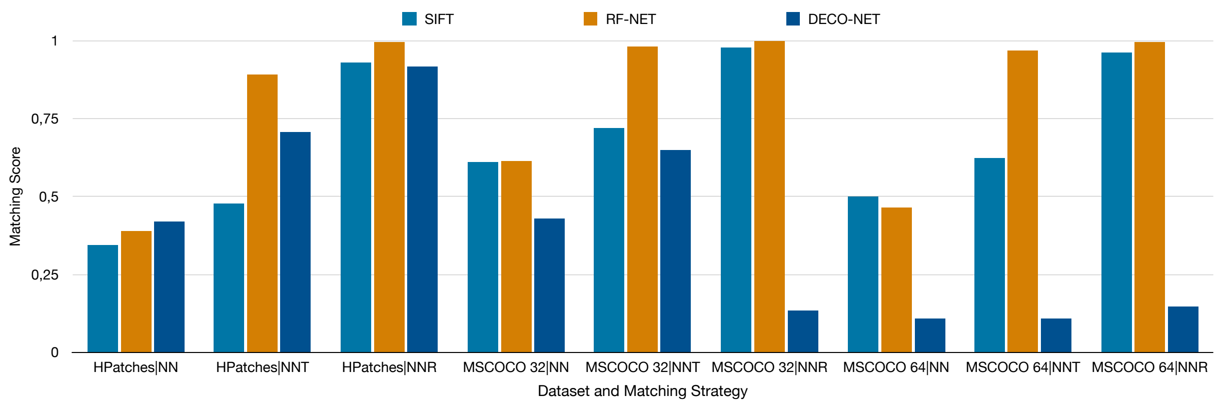


Figure 4.2: Matching scores for the methods under inspection. We report scores for the different matching strategies introduced in 4.3.1 on the three test sets from our datasets. The number after MS COCO indicates the ρ-value.

RF-Net outperforms the other two methods for all *NNT* and *NNR* based matching strategies as well as for the *NN* matching strategy on *MS COCO 2017* with $\rho = 32$. *SIFT* is the strongest for *NN* matching on *MS COCO 2017* with $\rho = 64$ and *DeCo-Net* has the highest *NN* matching score on *HPatches*. All methods are quite competitive except for *DeCo-Net* on *MS COCO 2017* with $\rho = 64$ and for *NNR* based matching on *MS COCO 2017* with $\rho = 32$. Note that the matching score thresholds τ_d and τ_r are borrowed from [68], and thus, might be advantageous for *RF-Net*.

Looking at the *NN* matching scores allows us to make some assumptions about the properties of the algorithms. Firstly, we see that the learned approaches outperform *SIFT* on *HPatches*. In *HPatches*, corresponding keypoints from two images must not only account for geometric changes but also for illumination and other artifacts introduced when taking a photo. The two learned approaches outperform *SIFT* since they can learn dataset-specific invariances, indicating that learning-based approaches are especially useful when working with images that show more than geometric variance, e.g. modal variance. Secondly, we see that *SIFT* and *RF-Net* outperform *DeCo-Net* on the synthetic *MS COCO 2017* dataset which contains exclusively geometric variations. The strong performance of *SIFT* can be explained by the handcrafted features that are optimized to be invariant to certain geometric changes. The discrepancy between *RF-Net* and *DeCo-Net* on *MS COCO 2017* can be explained by the way the two models account for geometric variations. *RF-Net* predicts the local geometries of two corresponding image patches and normalizes them before extracting their features. Given that the predicted geometries are correct and ignoring artifacts introduced by warping, the descriptor is fed with exactly the same input image for both patches, and thus, can produce the same features. *DeCo-Net*, on the other hand, first extracts features from the two differing image patches and then tries to compensate geometric variation by sampling at meaningful locations as described in Section 3.3. Since the features from the two images are likely to vary, this method is less powerful to account for geometric variation, and consequently, *DeCo-Net* is inferior to *SIFT* and *RF-Net* on *MS COCO 2017*. This issue is further discussed in the context of equivariance in Chapter 5.

4.6 Homography Estimation

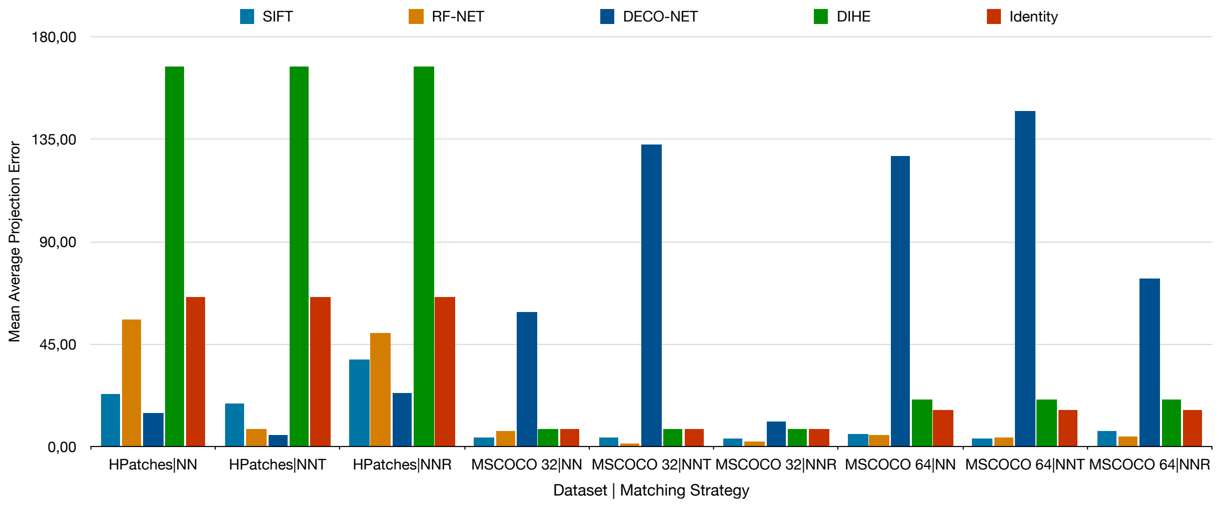


Figure 4.3: Mean average projection errors on our test sets. DIHE and Identity have no matching strategy, and thus, the same value is reported for all strategies.

We quantify the performance of the image registration approaches *SIFT*, *RF-Net*, *DeCo-Net*, *DIHE*, and *Identity* using the metrics introduced in Section 4.3.2. We report performance on the test sets from *HPatches* and *MS COCO 2017* with $\rho \in \{32, 64\}$. For the keypoint-based approaches, 128 keypoints are used with matching score thresholds $\tau_d = 1.0$ and $\tau_r = 0.7$ and *RANSAC* to estimate a homography from a set of matched keypoints. Further, we use the hyperparameters reported in Section 4.4 that achieved the best MMS score on the evaluation sets. *DIHE* has no matching strategy, and thus, the same value is reported for all matching strategies. *Identity* means that we use the identity matrix as the predicted homography and can be interpreted as the dataset-specific projection error. The resulting *mean average projection errors* can be found in Figure 4.3. The MAPE for *Identity* is the highest on *HPatches* indicating that this is the hardest dataset. Further, it is noticeable that *DIHE* performs worse on *HPatches* than on *MS COCO 2017* and that *DeCo-Net* performs worse on *MS COCO 2017* than on *HPatches*. Lastly, we can see that *DeCo-Net* has the best performance on *HPatches* while *SIFT* and *RF-Net* have the best performance on *MS COCO 2017*.
Since single outliers with extremely large *average projection errors* can push the *mean average projection error* up, especially for the smaller *HPatches* test set, we also report the *thresholded mean average projection error* with a threshold $\tau = 39.9$ in Figure 4.4. As bad algorithms could achieve high scores in this metric, we further report the *CorrH* value for the same threshold $\tau = 39.9$ to put the TMAPE into context. If the *CorrH* value is close to 1.0 and the TMAPE strongly differs from the MAPE it is an indicator that there are single outliers that cause the MAPE to be large, as for *RF-Net* on *HPatches|NN* and for *SIFT* on *HPatches|NNT & NNR*. On the other hand, a *CorrH* value close to 0.0, as for *DeCo-Net* on *MS COCO 2017* with $\rho = 64$ and *DIHE* on *HPatches*, is an indicator that the TMAPE reflects only a small part of the dataset, and thus, is not meaningful.
The proportion of correctly aligned images, where a registration is considered correct if the *APE* is below 5.0, can be seen in Figure 4.5. *DeCo-Net* outperforms all other methods on *HPatches|NN* and

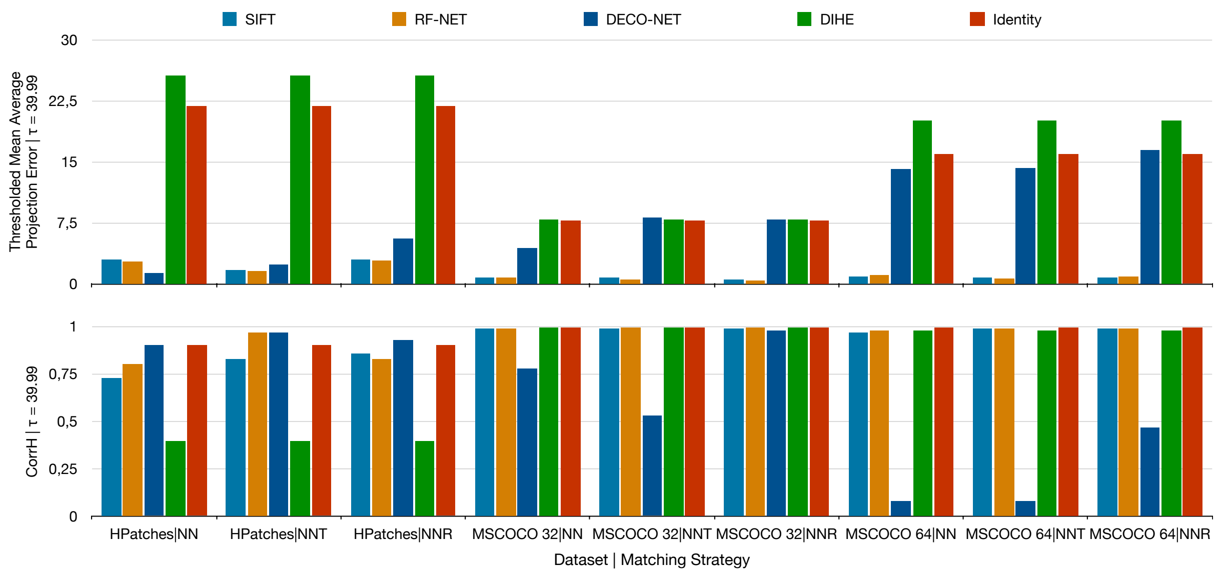


Figure 4.4: Thresholded mean average projection errors (top) and CorrH (bottom) with $\tau = 39.9$ for our test sets. DIHE and Identity have no matching strategy, and thus, the same value is reported for all strategies.

is comparable to *RF-Net* for the other matching strategies on *HPatches*. On the synthetic dataset, *SIFT* and *RF-Net* are the best methods with *RF-Net* having a small lead over *SIFT*. *DIHE* has low performance on all datasets. The discussion for the performance of *SIFT*, *RF-Net*, and *DeCo-Net* follows the arguments stated in the previous section. *DIHE* has a worse performance for the harder datasets *HPatches* and *MS COCO 2017* with $\rho = 64$. This is caused by the strong distortions and small dataset size of *HPatches*. Unlike the keypoint-based approaches, *DIHE* directly estimates a homography causing the target space to be larger when the transformations present in the dataset are larger. This also shows in unsuccessful training when not initializing with a pre-trained model.

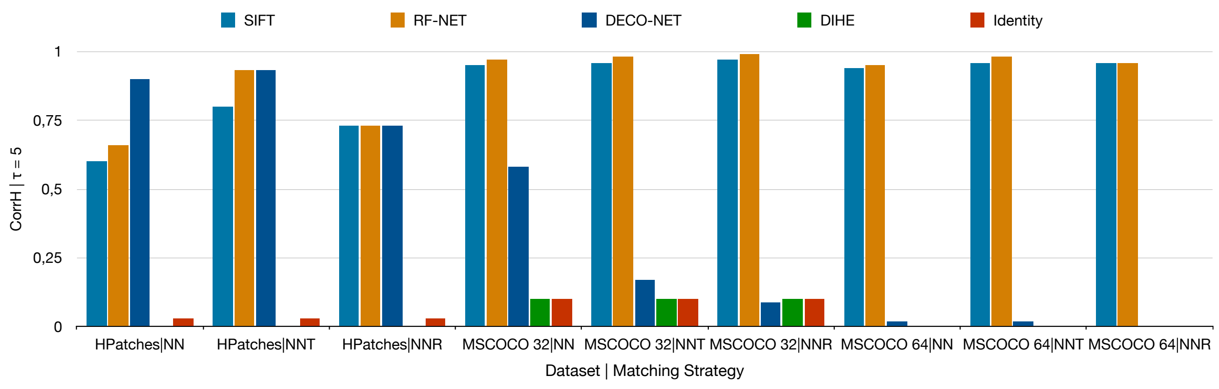


Figure 4.5: CorrH values with $\tau = 5$ on our test sets. DIHE and Identity have no matching strategy, and thus, the same value is reported for all strategies.

However, for *MS COCO 2017* with $\rho = 32$ the model is extremely robust in the sense that there is no $\texttt{APE} > 39.9$. Although it is not visible in the plots, none of the keypoint-based approaches could achieve such robustness. This indicates that, given training was successful and the amount of distortion can be handled, *DIHE* lacks exact alignments that keypoint-based approaches provide but is an extremely robust estimator that has no strong outliers as it is the case for *RANSAC* that produces up to 50% outliers on *MS COCO 2017* with $\rho = 32$. This means *DIHE* would be a good method to estimate a brief pre-alignment that can be used to constrain *C-RANSAC* as tested in Section 4.10.

4.7 Decoupling the Detector and Descriptor

One potential downside of *RF-Net* is the strong dependence of the detector on the descriptor, which results in a somewhat heuristic training scheme where the descriptor is trained twice while the detector is trained once. To evaluate if this dependence negatively affects the training and if our additional local geometry loss helps to reduce the dependence, we conduct the following experiments: Firstly, we change the number of times the descriptor is trained compared to the detector from two (*Regular*) to one (*Des Step = 1*), keeping all other hyperparameters the same. In a setting where the detector is strongly influenced by the descriptor, we expect the change to have a stronger impact on the matching scores. Secondly, we will remove all impact of the descriptor on the detector, i.e. remove the pair loss (*w/o pair loss*). Since the detection in *DeCo-Net* can be reformulated as locations in the image that have a predictable local geometry we expect *DeCo-Net* to perform better than *RF-Net* where keypoints are locations with discriminative features. We train *RF-Net* and *DeCo-Net* with the modifications on *MS COCO 2017* with $\rho = 32$ and report the matching scores in Figure 4.6. Changing the number of times the descriptor is trained compared to the detector decreases the performance of *RF-Net* marginally. *DeCo-Net* has a drop in performance for the *NN* and *NNT* matching strategies while performance increases for the *NNR* matching strategy. Completely removing the pair loss causes a drop in performance for *RF-Net*, especially for the

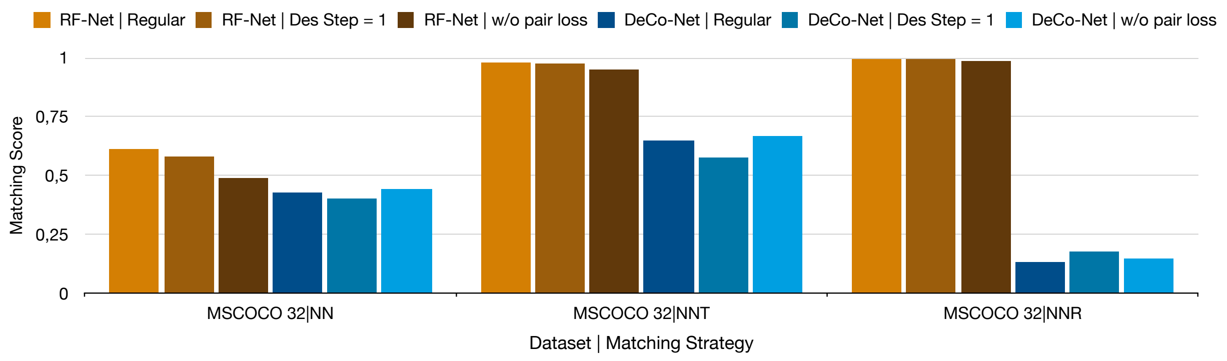


Figure 4.6: Matching scores on MS COCO 2017 with $\rho = 32$. RF-Net and DeCo-Net were trained in a regular way (Regular) as described in Section 4.1, with less descriptor steps (Des Step = 1), and without pair loss (w/o pair loss).

NN matching strategy, while slightly increasing performance for *DeCo-Net*. Those observations indicate that the detector in *RF-Net* is more dependent on the descriptor than in *DeCo-Net*. To be more precise, removing the pair loss impairs the keypoint detection in *RF-Net* in two ways. Firstly, the detector has no information about the keypoint features, and thus, cannot find meaningful keypoint locations with discriminative features. Secondly, the local geometry estimation is indirectly supervised by the pair loss. Consequently, the network cannot learn to estimate a meaningful canonical pose. *DeCo-Net*, on the other hand, has no functionality that is exclusively supervised by the pair loss, and therefore, shows no drop in performance when removing the pair loss. This property could be particularly useful when fine-tuning only the detector of *DeCo-Net* or when developing application-specific descriptors for the same detector.

4.8 Domain Adaption

The lack of task-specific annotated training data is an ever-present problem in deep learning, motivating the development of unsupervised and semi-supervised approaches [19, 64]. Two alternative approaches to solve the problem are self-supervised learning [36] and domain adaption [39]. Self-supervised training denotes a supervised task where no manual data labeling is required [16]. Hence, training on our synthetic *MS COCO 2017* dataset can be considered as self-supervised training. Domain adaption is a technique where a network is trained on a large corpus of data D_{train} and tested on the actual dataset of interest D_{target} that comes from another related domain [39]. This section aims to evaluate how well our methods can adapt to a new domain D_{target} when trained on a different and automatically labeled training domain D_{train}. To our knowledge, we are the first to train fully learned keypoint extraction methods in a self-supervised setting, allowing us to remove the need for any human supervision. We train our methods on *MS COCO 2017* with $\rho = 32$ and evaluate the performance on the *HPatches* test set. Different metrics for the three learning-based approaches *DeCo-Net*, *RF-Net*, and *DIHE* are reported in Table 4.3. For the keypoint-based approaches, the *NN* matching strategy is used. We report numbers on the test sets for regular training on *MS COCO 2017* with $\rho = 32$ and *HPatches* in the first two columns and numbers for the domain adaption setup in the third column. *MS COCO 2017* with $\rho = 32$ is used for pre-training since it is the only dataset where all learning-based approaches could successfully be trained.

All methods could transfer some of the learned knowledge to the new domain and perform reasonable well on *HPatches*. However, *DeCo-Net* has a significant drop in performance in all metrics compared to when it was trained on *HPatches*. *RF-Net* shows only a slight decrease in performance while performance for *DIHE* even increases. We assume performance for *DIHE* increases because it was not able to learn anything useful on the small *HPatches* dataset. *RF-Net* outperforms *DeCo-Net* in the domain adaption setting because it was able to learn more meaningful features on *MS COCO 2017*, as shown by the better scores for regular training. When comparing the results of the strongest method in this experiment, *RF-Net*, with the results from *SIFT* in Section 4.6 we see that performance is almost identical. Therefore, our results suggest that a learned method can compete with *SIFT* on a real dataset without the need for any human supervision, increasing practicability drastically.

			Regular Training		Domain Adaption
			MS COCO 32	HPatches	MS COCO 32 → HPatches
DeCo-Net	Metric	MAPE	59.06	14.65	58.92
		TMAPE | τ = 39.9	4.36	1.37	3.5
		CorrH | τ = 39.9	0.78	0.9	0.57
		CorrH | τ = 5.0	0.58	0.9	0.43
RF-Net		MAPE	6.65	55.78	63.10
		TMAPE | τ = 39.9	0.75	2.73	2.52
		CorrH | τ = 39.9	0.99	0.8	0.7
		CorrH | τ = 5.0	0.97	0.66	0.63
DIHE		MAPE	7.89	166.71	45.18
		TMAPE | τ = 39.9	7.89	25.56	21.2
		CorrH | τ = 39.9	1.0	0.4	0.87
		CorrH | τ = 5.0	0.8	0.0	0.03

Table 4.3: Homography estimation metrics for our learning-based approaches. The first two columns show numbers on the test sets for regular training. The third column shows numbers for the domain adaption setup, where the network is trained on MS COCO 2017 and evaluated on HPatches. All methods were able to adapt to the new domain.

4.9 Memory Usage

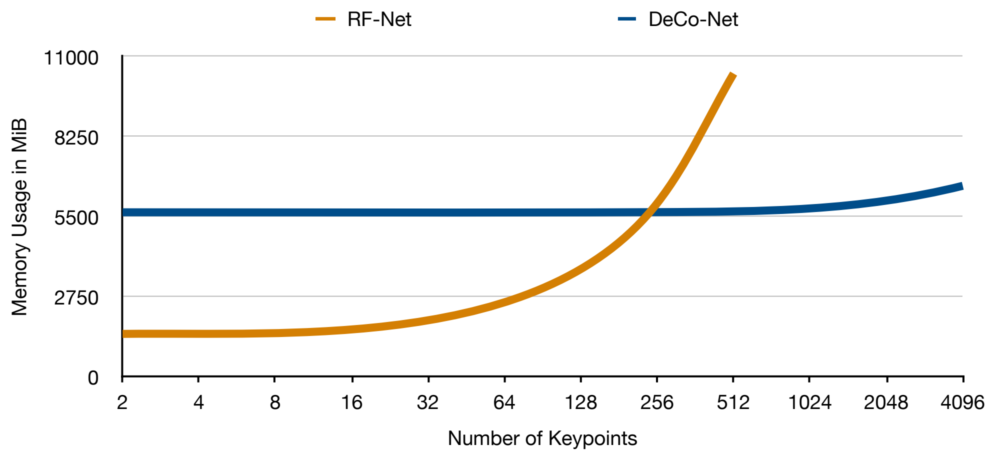


Figure 4.7: Memory usage of RF-Net and DeCo-Net versus the number of keypoints used at train time. We cannot report numbers for RF-Net with more than 512 keypoints due to the 11GB memory constraint of our GPU. However, we expect the curve to continue to grow semi-linear.

A major motivation to develop *DeCo-Net* was the high increase of required memory when training *RF-Net* with more keypoints. The following experiment aims to evaluate if our efforts improved scalability. To do so, we report the model size for a different number of keypoints at train time for *RF-Net* and *DeCo-Net*. The results can be seen in Figure 4.7. The model size grows semi-linear with more keypoints. Since *RF-Net* computes the feature for each keypoint separately, it starts at a lower memory usage but increases quickly with more keypoints. At roughly 256 keypoints *RF-Net's* and *DeCo-Net's* memory usage intersects and with more keypoints *RF-Net* requires more memory. *Deco-Net*, on the other hand, precomputes features for each pixel in the image. This means that it starts at a higher memory usage that increases only a little when increasing the number of keypoints. Therefore, our effort to improve scalability showed to be successful. When using 512 keypoints for training, *DeCo-Net* requires only half the amount of memory compared to *RF-Net*. Note that the experiment conducted in Section 4.4 indicates that using more keypoints than reasonable for a given image size can impair training.

4.10 Constraining RANSAC

When estimating homographies from two sets of keypoints it is crucial to find correct matches. Since there is no notion of continuity in keypoint matching, even minor mismatches can produce large errors. This problem shows in previous experiments where the `MAPE` often exceeds the `TMAPE` by a large margin. To reduce sensitivity to wrong matches we proposed a method to constrain *RANSAC*, *C-RANSAC*, in Section 3.2. We now evaluate if our method is useful. To do so, we use *C-RANSAC* to estimate homographies that align *MS COCO 2017* with $\rho = 32$ using keypoints extracted from *DeCo-Net* with *NN* matching. *MS COCO 2017* with $\rho = 32$ is a well-suited dataset because we can provide strong constraints. Our first constraint (*DIHE*) makes use of the robustness of *DIHE*. We use *DIHE* to estimate a brief pre-alignment $\boldsymbol{H}_{ref}$ and enforce the homography $\hat{\boldsymbol{H}}$ estimated by *DeCo-Net* + *C-RANSAC* to have an $\texttt{APE}(\hat{\boldsymbol{H}}, \boldsymbol{H}_{ref}) \leq 40$. We set $\tau_{ref} = 40$ because the maximum `APE` for *DIHE* on *MS COCO 2017* with $\rho = 32$ is less than 40 (`CorrH` $39.9 = 1.0$). Our second constraint (*ID*) makes use of prior knowledge about the dataset. Since we know that the maximum `APE` in the dataset is 32 we can set the reference homography $\boldsymbol{H}_{ref}$ to the identity and use the reference threshold $\tau_{ref} = 32$. Numbers for *DeCo-Net* with regular *RANSAC* and *C-RANSAC* with our two constraints are reported in Figure 4.8. *C-RANSAC* improves the `MAPE` and `CorrH` with $\tau = 39.9$ and $\tau = 5$. We assume `CorrH` with $\tau = 5$ only increases marginally because the cases that are affected by the constraint are hard, and thus, it is unlikely that a homography is estimated with an `APE` ≤ 5. The slightly worse `TMAPE` with $\tau = 39.9$ can be explained by the almost 25% increase in samples that are included in the `TMAPE`, which can be seen in the `CorrH` value with $\tau = 39.9$. Further, it shows that the strength of the constraint correlates with the improvement in performance. Thus, the stronger *ID* constraint improves performance more than the *DIHE* constraint. All in all, *C-RANSAC* proves to be extremely effective in finding correct matches, given an appropriate constraint. In particular, homographies with large `APE`s are eliminated, as can be seen in the `MAPE` and `CorrH` with $\tau = 39.9$.

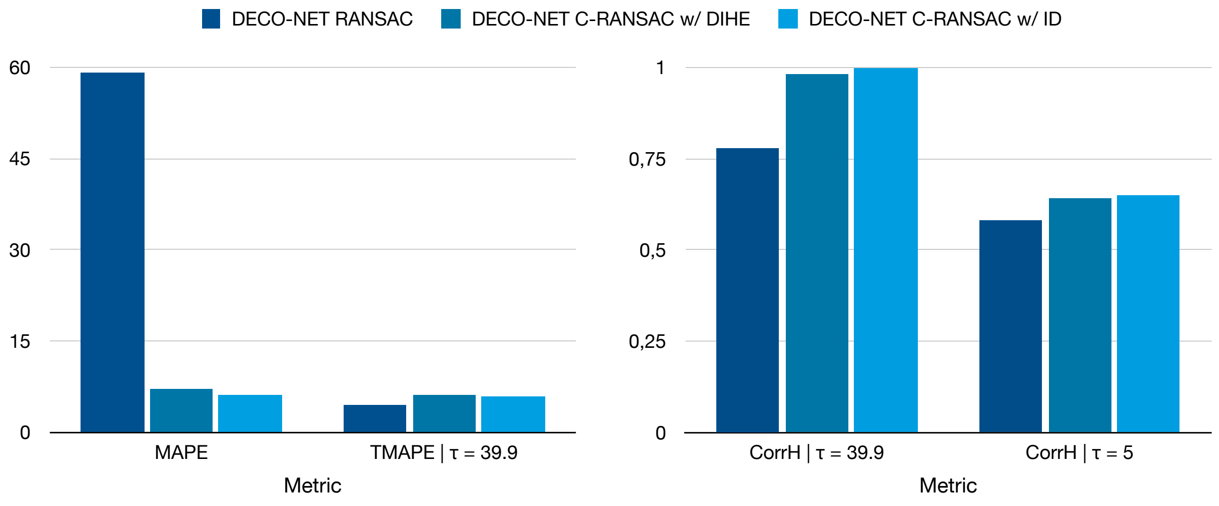


Figure 4.8: Comparison of RANSAC and our novel C-RANSAC method on MS COCO 2017 with $\rho = 32$ using keypoints extracted from DeCo-Net with NN matching. Two different constraints, ID and DIHE explained in Section 4.10, were used.

5 Discussion and Future Work

The leading question of this thesis was whether deep learning induced a paradigm shift in image registration and if this shift was justified. After evaluating state-of-the-art, traditional (*SIFT* [46]) and learning-based (*DIHE [14], RF-Net [68], DeCo-Net*) approaches for homography estimation and surveying recent literature, we can answer both questions with a clear *'yes'*. Our conclusion is supported by the development of a variety of new approaches that utilize deep learning for image registration, and by a quantitative evaluation of traditional and learning-based approaches conducted in Chapter 4, indicating the superiority of learning-based approaches. However, this does not imply that traditional methods are outdated or should be neglected. *SIFT* can often keep up with the learned methods, in particular on datasets that only show geometric variations, and outperforms *DIHE* and *DeCo-Net* on several benchmarks. Most importantly, *SIFT* has the advantage of no need for training, little memory usage, fast inference time, and easy usability due to implementations in open source libraries like *OpenCV* [49]. Therefore, we recommend using traditional methods as an initial approach to solve the image registration problem at hand and switch to more complex learning-based approaches if performance requirements are not met.
Overall, *RF-Net* achieved the best performance in our experiments. It proves to be extremely effective in compensating geometric variations, even on the hard *MS COCO 2017* dataset with $\rho = 64$ where other learned approaches failed. While working with *RF-Net* we found two potential downsides that could limit practical use. Firstly, the model size increases significantly when increasing the number of keypoints. This issue might be problematic when working on large images and having small GPUs. Secondly, the detector is highly dependent on the descriptor, resulting in a heuristic training scheme where the descriptor is trained twice in each training iteration. This dependence could be even more problematic when fine-tuning only the detector on a new, smaller dataset. With that said, future studies should consider these potential practical problems.
The issues just mentioned are addressed by our novel method *DeCo-Net*. Our experiments show that both scalability and independence have been improved compared to *RF-Net*. This comes at the cost of decreasing the ability to compensate geometric variations, as can be seen in the performance on the synthetic *MS COCO 2017* dataset. However, on the natural images of *HPatches*, *DeCo-Net* outperformed all competing methods, showing that it is a powerful keypoint extraction method for real-world datasets with limited geometric variation.
Both *RF-Net* and *DeCo-Net* have a multitude of hyperparameters of which some can have a significant impact on the performance. Additionally, we observed that the matching strategy of the keypoint-based approaches strongly affects the quality of the matches. Since different matching strategies are easy to implement and cause almost no computational overhead, we recommend considering several strategies simultaneously in practical applications. Further, we noticed that minor mismatches can

result in large errors caused by the discontinuity of keypoint matching. To address this problem we proposed *C-RANSAC* as a method to constrain *RANSAC* that shows to be extremely helpful. We managed to reduce the *mean average projection error* of *DeCo-Net* on *MS COCO 2017* with $\rho = 32$ by 50 points when incorporating prior knowledge of the dataset or a brief estimate from *DIHE*.
We have encountered several problems when testing *DIHE*. *DIHE* lacks accuracy and could only be trained on *MS COCO 2017* with $\rho = 32$. The small *HPatches* dataset had an insufficient number of samples and the distortions in *MS COCO 2017* with $\rho = 64$ have been too strong for the network to learn. If trained successfully, the method is extremely robust in the sense that no strong outliers were produced. Nowruzi et al. showed that performance can be increased by stacking multiple instances of the *DIHE* network that align residuals [61]. Another advantage of *DIHE* is the simplicity of the end-to-end approach. However, due to the lack of accuracy and problems while training, we do not see *DIHE* to be competitive to the keypoint-based approaches. Nonetheless, the great robustness of *DIHE* can be used to constrain *C-RANSAC* and increase the performance of keypoint-based approaches.
We assume that *RF-Net* and *DeCo-Net* can be trained on much smaller datasets than *DIHE* because each patch around a keypoint can be interpreted as an individual sample. So when training on the 230 samples in *HPatches* with 512 keypoints, we have an effective dataset size of $230 \times 512 = 117760$. The ability to train on small datasets might be the biggest practical advantage of the keypoint-based approaches over other learning-based approaches that require much larger datasets. Another option to circumvent the problem of small datasets is domain adaption. All methods were able to transfer some knowledge obtained from self-supervised training on the synthetic *MS COCO 2017* dataset to the real *HPatches* dataset. *RF-Net* could even achieve comparable performance to *SIFT* without the need for any human supervision.
In this thesis, we limited our attention to the class of viewpoint registration where mainly geometric variations occur. The traditional method, *SIFT*, is designed to compensate those geometric variations, and thus, proves to be competitive in all our benchmarks. Since learning-based approaches can produce more task-specific similarity measures, we believe that learning-based approaches are even more useful for other registration classes, e.g. multimodal registration, where the notion of similarity exceeds sheer appearance.
Essentially, in viewpoint registration, the design of a strong keypoint estimation approach comes down to the concept of equivariance. A function f is said to be equivariant if applying a symmetry transformation T and then computing f produces the same result as computing f and then applying T. In our context of learned keypoint estimation and homography estimation, equivariance means that it does not make a difference if we first apply a transformation, i.e. a homography $\boldsymbol{H}$, to an image I and then compute the keypoints or if we first compute the keypoints and then apply $\boldsymbol{H}$, as illustrated in Figure 5.1. The reason why equivariance is essential for keypoint estimation can be seen in *DeCo-Net* for the simple case of a translation $\boldsymbol{H}'$. Convolutional neural networks are equivariant to translation by design [11, 10]. Thus, when computing the keypoints and features for an image I and its translated version $\boldsymbol{H}' \circ I$, the resulting feature maps are also just a translated version of each other. As a consequence, the keypoints are located at the exact corresponding image locations while the features for each corresponding keypoint are the same as well. This means we have two sets of keypoints with a perfect *matching score* of 1.0, given that no two features of the same set are equal. Since this behavior is caused by design, the network does not even need to be trained. This idea of equivariance can be generalized to projective transformations to implement

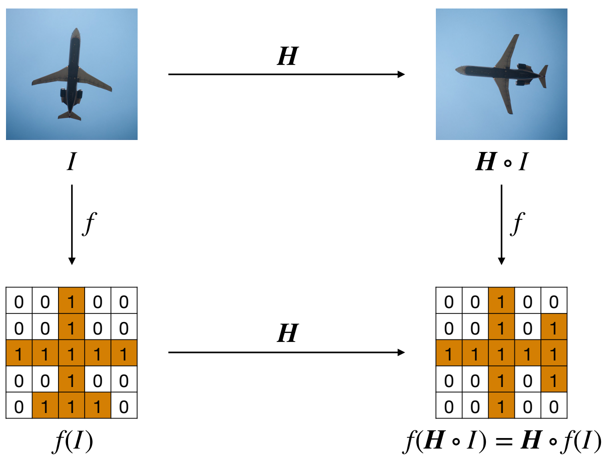


Figure 5.1: Illustration of the concept of equivariance in the context of learned keypoint estimation. It does not matter if we first apply a homography $\boldsymbol{H}$ to an image I and then a function f that computes the keypoints (denoted in orange), or if we first apply f to I and then $\boldsymbol{H}$.

an even stronger keypoint estimation model. Unfortunately, convolutional neural networks are only equivariant to translation, and hence, equivariance to projective transformations must be realized in another way. We assume this inability to be equivariant to projective transformations is the reason why *DeCo-Net* performs worse on the synthetic datasets than on *HPatches*. On the synthetic dataset, performance reflects the model's capacity to be equivariant to viewpoint changes. Here, *RF-Net* has an advantage since the input itself is normalized to a canonical pose before the feature is computed. Assuming a perfect local geometry estimation, the feature is computed from the same input (ignoring artifacts introduced by the warping), and thus, equal for two corresponding keypoints. In *DeCo-Net*, on the other hand, the feature map itself must be equivariant to projective transformations which is only possible to a limited degree (except in the trivial solution where all features are the same), so the features cannot be as equal as in *RF-Net* for two corresponding keypoints. Therefore, a reasonable future step would be to increase *DeCo-Net's* equivariance to projective transformations. More equivariance to scaling could, for example, be achieved by incorporating feature pyramids [13] or a feature pyramid network [44]. This would introduce feature representations at different scales to *DeCo-Net* that can be sampled according to the scale of a keypoint. This idea can be generalized to work for other transformations, e.g. rotation, by introducing additional feature maps for a pre-defined set of rotation classes. To do so, the input image is rotated by the angles of the rotation class (e.g. $[0°, 90°, 180°, 270°]$) and feature maps are computed from all rotated images. The feature sampling now needs to be adjusted to sample from the correct rotation-space. The downside to this method is, considering the large size of one dense feature map, the enormous memory requirement. A more elegant approach to achieve more equivariance is to make use of recent approaches that aim to increase the equivariance of convolutional neural networks by design [11, 10, 75]. Examples of such are *equivariant transformer networks (ET)* [75] that achieve equivariance to specific transformations, e.g. rotation, by exploiting the translational equivariance of CNNs in specialized canonical coordinate systems. A regular image can be represented in a canonical coordinate system in which the transformation group G of interest is equivariant under translation, resulting in equivariance to G [75]. Another important future step is to evaluate the methods on more diverse and larger datasets. To gain deeper knowledge about an algorithm's performance, other image registration classes but viewpoint registration must be examined as well. Furthermore, one could automatically adjust the triplet loss balancing factor β. We observed that too high and low values for β can cause the descriptor to get stuck in local optima and to produce features that are too similar or too different. Those two cases could be detected by assuring that the statistical deviation of the features $\boldsymbol{D}$ does not fall below or exceed predefined thresholds τ_{low} and τ_{high}

$$\tau_{low} \leq \frac{1}{K}\sum_{k}^{K} |\boldsymbol{D}^k - m(\boldsymbol{D})| \leq \tau_{high},$$

with K denoting the number of keypoints and m denoting a measure of central tendency, e.g. the mean. If such a local optimum is detected, β can be adjusted accordingly to escape from the local optimum.
To conclude, we evaluated different traditional and learning-based approaches for homography estimation and showed that deep learning caused a paradigm shift in image registration. While doing so, we established new benchmarks and presented *DeCo-Net*, a novel, learned method for local feature detection and description. Furthermore, we proposed an extension of *RANSAC* and the *average projection error* metrics to estimate and evaluate homographies. Lastly, we showed that learned keypoint extraction methods can be trained in a self-supervised setting and only require small datasets compared to other deep learning techniques. This independence of data and the associated low costs for data acquisition might be key to enable a broad usage of learned keypoint extraction methods in real-world applications.

Bibliography

[1] P. E. Anuta. “Spatial Registration of Multispectral and Multitemporal Digital Imagery Using Fast Fourier Transform Techniques”. In: *IEEE Transactions on Geoscience Electronics* 8 (1970).

[2] Shmuel Avidan. “Object classification using image segmentation”. In: *United States Patent* (2006).

[3] M. Bach Cuadra, V. Duay, and J.-Ph. Thiran. “Atlas-based Segmentation”. In: *Handbook of Biomedical Imaging: Methodologies and Clinical Research* (2015).

[4] Simon Baker, Ankur Datta, and Takeo Kanade. “Parameterizing Homographies”. In: *Tchnical Report: Carnegie Mellon University* CMU-RI-TR-06-11 (2006).

[5] Vassileios Balntas et al. “HPatches: A benchmark and evaluation of handcrafted and learned local descriptors”. In: *Conference on Computer Vision and Pattern Recognition* (2017).

[6] Vivek Bannore. “Image Registration for Super-Resolution”. In: *Iterative-Interpolation Super-Resolution Image Reconstruction* (2009).

[7] Jane Bromley et al. “Signature Verification using a "Siamese" Time Delay Neural Network”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 7 (1993).

[8] Lisa Gottesfeld Brown. “A Survey of Image Registration Techniques”. In: *ACM Computing Surveys* 24 (1992).

[9] Xi Cheng, Li Zhang, and Yefeng Zheng. “Deep similarity learning for multimodal medical images”. In: *CMBBE: Imaging and Visualization* 6 (2016).

[10] Taco S. Cohen and Max Welling. “Group Equivariant Convolutional Networks”. In: *International Conference on Machine Learning* (2016).

[11] Taco Cohen et al. “Gauge Equivariant Convolutional Networks and the Icosahedral CNN”. In: *Proceedings of the 36th International Conference on Machine Learning* 97 (2019).

[12] Robert Cupec et al. “Geometrically Constrained RANSAC for Stereo Image Registration in Presence of High Ambiguity in Feature Correspondence”. In: *4th European Conference on Mobile Robots* (2009).

[13] N. Dalal and B. Triggs. “Histograms of Oriented Gradients for Human Detection”. In: *Conference on Computer Vision and Pattern Recognition* (2005).

[14] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “Deep Image Homography Estimation”. In: *RSS Workshop on Limits and Potentials of Deep Learning in Robotics* (2016).

[15] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “SuperPoint: Self-Supervised Interest Point Detection and Description”. In: *Conference on Computer Vision and Pattern Recognition Workshops* (2018).

[16] Carl Doersch and Andrew Zisserman. “Multi-task Self-Supervised Visual Learning”. In: *International Conference on Computer Vision* (2017).

[17] Elan Dubrofsky. “Homography Estimation”. In: *Master’s Assay - Chapter 2.1* (2007).

[18] Mihai Dusmanu et al. “D2-Net: A Trainable CNN for Joint Detection and Description of Local Features”. In: *Conference on Computer Vision and Pattern Recognition* (2019).

[19] Dumitru Erhan et al. “Why does unsupervised pre-training help deep learning?” In: *Journal of Machine Learning Research* 11 (2010).

[20] Jingfan Fan et al. “Adversarial Similarity Network for Evaluating Image Alignment in Deep Learning Based Registration”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention* 11070 (2018).

[21] Martin A. Fischler and Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Communications of the ACM* 24 (1981).

[22] Alshimaa Y. Abo Gharbia et al. “Registration-based change detection for SAR images”. In: *NRIAG Journal of Astronomy and Geophysics* 9 (2020).

[23] Ian J. Goodfellow et al. “Generative Adversarial Networks”. In: *Conference on Neural Information Processing Systems* (2014).

[24] Arthur Ardeshir Goshtasby. “2-D and 3-D Image Registration: for Medical, Remote Sensing, and Industrial Applications”. In: *Wiley-Interscience* (2005).

[25] R. I. Hartley and A. Zisserman. “Multiple View Geometry in Computer Vision”. In: *Cambridge University Press* (2004).

[26] Grant Haskins, Uwe Kruger, and Pingkun Yan. “Deep learning in medical image registration: a survey”. In: *Machine Vision and Applications* 31 (2019).

[27] Grant Haskins et al. “Learning deep similarity metric for 3D MR-TRUS image registration”. In: *International Journal of Computer Assisted Radiology and Surgery* 14 (2018).

[28] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Conference on Computer Vision and Pattern Recognition* (2016).

[29] Kaiming He et al. “Mask R-CNN”. In: *International Conference on Computer Vision* (2017).

[30] Mattias Heinrich et al. “MIND: Modality Independent Neighbourhood Descriptor for Multi-Modal Deformable Registration”. In: *Medical image analysis* 16 (2012).

[31] R. Hesse. “Development and Evaluation of 3D Autoencoders for Feature Extraction”. In: *Bachelor’s Thesis* (2017).

[32] Yipeng Hu et al. “Weakly-supervised convolutional neural networks for multimodal image registration”. In: *Medical Image Analysis* 49 (2018).

[33] Ilija Ilievski et al. “Efficient Hyperparameter Optimization of Deep Learning Algorithms Using Deterministic RBF Surrogates”. In: *Computing Research Repository* arXiv 1607.08316 (2016).

[34] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning* 37 (2015).

[35] Max Jaderberg et al. “Spatial Transformer Networks”. In: *Conference on Neural Information Processing Systems* (2015).

[36] Longlong Jing and Yingli Tian. “Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey”. In: *Computing Research Repository* arXiv 1902.06162 (2019).

[37] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition.” In: *Computing Research Repository* arXiv 1409.1556 (2014).

[38] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* Poster (2014).

[39] Wouter M. Kouw and Marco Loog. “An introduction to domain adaptation and transfer learning”. In: *Computing Research Repository* arXiv 1812.11806 (2018).

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Conference on Neural Information Processing Systems* (2012).

[41] Van-Hung Le et al. “Acquiring qualified samples for RANSAC using geometrical constraints”. In: *Pattern Recognition Letters* 102 (2017).

[42] Hongming Li and Yong Fan. “Non-rigid image registration using self-supervised fully convolutional networks without training data”. In: *15th International Symposium on Biomedical Imaging* (2018).

[43] Rui Liao et al. “An Artificial Agent for Robust Image Registration”. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[44] Tsung-Yi Lin et al. “Feature Pyramid Networks for Object Detection”. In: *Conference on Computer Vision and Pattern Recognition* (2017).

[45] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *European Conference on Computer Vision* (2014).

[46] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60 (2004).

[47] Kai Ma et al. “Multimodal Image Registration with Deep Context Reinforcement Learning”. In: *Medical Image Computing and Computer Assisted Intervention* (2017).

[48] Andrew L. Maas. “Rectifier Nonlinearities Improve Neural Network Acoustic Models”. In: *International Conference on Machine Learning* (2013).

[49] Naveenkumar Mahamkali and A Vadivel. *OpenCV for Computer Vision Applications*. (2015).

[50] Niall O’ Mahony et al. “Deep Learning vs. Traditional Computer Vision”. In: *Computing Research Repository* arXiv 1910.13796 (2019).

[51] Stephen Ka Fai Mann. “Compositing multiple pictures of the same scene: generalized large displacement 8-parameter motion”. In: *The Society of Imaging Science and Technology Proceedings of the 46th Annual IS&T Conference* (1993).

[52] Alfonso Medela and Artzai Picon. “Constellation Loss: Improving the efficiency of deep metric learning loss functions for optimal embedding”. In: *Computing Research Repository* arXiv 1905.10675 (2019).

[53] Shun Miao et al. “Dilated FCN for Multi-Agent 2D/3D Medical Image Registration”. In: *Association for the Advancement of Artificial Intelligence* (2018).

[54] Krystian Mikolajczyk and Cordelia Schmid. “A Performance Evaluation of Local Descriptors”. In: *IEEE transactions on pattern analysis and machine intelligence* 27 (2005).

[55] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518 (2015).

[56] David Monnin et al. “An Effective Rigidity Constraint for Improving RANSAC in Homography Estimation”. In: *Advanced Concepts for Intelligent Vision Systems* (2010).

[57] Hans P. Moravec. “Rover Visual Obstacle Avoidance”. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence* 2 (1981).

[58] Sayan Nag. “Image Registration Techniques: A Survey”. In: *Computing Research Repository* arXiv 1712.07540 (2017).

[59] Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning* (2010).

[60] Ty Nguyen et al. “Unsupervised Deep Homography: A Fast and Robust Homography Estimation Model”. In: *IEEE Robotics and Automation Letters* 3 (2018).

[61] F. E. Nowruzi, R. Laganiere, and N. Japkowicz. “Homography Estimation from Image Pairs with Hierarchical Convolutional Networks”. In: *International Conference on Computer Vision Workshops* (2017).

[62] NVIDIA. *GeForce RTX 2080 Ti*. Codename: TU102-300-K1-A1. (2018).

[63] Yuki Ono et al. “LF-Net: Learning Local Features from Images”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (2018).

[64] Nicolas Papernot et al. “Semi-supervised knowledge transfer for deep learning from private training data”. In: *International Conference on Learning Representations* (2016).

[65] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32* (2019).

[66] Edward Rosten, Reid Porter, and Tom Drummond. “FASTER and better: A Machine Learning Approach to Corner Detection”. In: *IEEE transactions on pattern analysis and machine intelligence* 32 (2010).

[67] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115 (2015).

[68] Xuelun Shen et al. “RF-Net: An End-To-End Image Matching Network Based on Receptive Field”. In: *Conference on Computer Vision and Pattern Recognition* (2019).

[69] Nadezhda Shusharinaa and Gregory Sharp. “Image registration using radial basis functions with adaptive radius”. In: *Med Phys.* (2012).

[70] Martin Simonovsky et al. “A Deep Metric for Multimodal Registration”. In: *Medical Image Computing and Computer-Assisted Intervention* (2016).

[71] Kihyuk Sohn. “Improved Deep Metric Learning with Multi-class N-pair Loss Objective”. In: *Advances in Neural Information Processing Systems* 29 (2016).

[72] Parvez Rahi Sombir Singh Bisht Bhumika Gupta. “Image Registration Concept and Techniques: A Review”. In: *Int. Journal of Engineering Research and Applications* 4 (2014).

[73] G. W. Stewart. “On the Early History of the Singular Value Decomposition”. In: *Society for Industrial and Applied Mathematics* (1993).

[74] I. E. Sutherland. “Three-dimensional data input by tablet”. In: *Proceedings of the IEEE* 62 (1974).

[75] Kai Sheng Tai, Peter Bailis, and Gregory Valiant. “Equivariant Transformer Networks”. In: *International Conference on Machine Learning* (2019).

[76] L. A. Teverovskiy et al. “Feature-Based vs. Intensity-Based Brain Image Registration: Comprehensive Comparison Using Mutual Information”. In: *International Symposium on Biomedical Imaging: From Nano to Macro* (2007).

[77] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Instance Normalization: The Missing Ingredient for Fast Stylization”. In: *Computing Research Repository* arXiv 1607.08022 (2016).

[78] Paul Viola and William M. Wells III. “Alignment by Maximization of Mutual Information”. In: *International Journal of Computer Vision* 24 (1997).

[79] Bob D. de Vos et al. “A deep learning framework for unsupervised affine and deformable image registration”. In: *Medical Image Analysis* 52 (2019).

[80] Ziyu Wang et al. “Dueling Network Architectures for Deep Reinforcement Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning* 48 (2016).

[81] Guorong Wu et al. “Unsupervised deep feature learning for deformable registration of MR brain images”. In: *Medical image computing and computer-assisted intervention* 16 (2013).

[82] Arber Zela et al. “Towards Automated Deep Learning: Efficient Joint Neural Architecture and Hyperparameter Search”. In: *Computing Research Repository* arXiv 1807.06906 (2018).