Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Comparison of Methods for Matching of Images with Weakly Textured Areas

And their usage in 3D image reconstruction

Ondřej Kafka

Supervisor: Ing. Michal Polic
Field of study: Cybernetics and robotics
May 2021

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Kafka Ondřej**	Personal ID number:	**483475**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Cybernetics and Robotics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Comparison of Methods for Matching of Images with Weakly Textured Areas

Bachelor’s thesis title in Czech:

Porovnání metod pro hledání korespondencí mezi obrazy s málo texturovanými oblastmi

Guidelines:

1. Get familiar with handcrafted techniques [1,2] about the matching of weakly textured areas (e.g., walls, floors, ceilings).
2. Get familiar with the learned method Sparse-NCNet [3] for obtaining tentative matches.
3. Compare the 3D scene accuracy created by handcrafted preprocessing and matching by standard algorithms (e.g., preprocessing by Wallis filter, detection by SIFT, matching by RANSAC) with learned matching by Sparse-NCNet. The reconstruction on top of the matches may be done by a standard reconstruction pipeline, e.g., COLMAP.
4. Retrain the Sparse-NCNet on the weakly textured areas and compare the proposed model with previous results.

[Bonus] Propose and develop a method for matching that overcome matching on weakly textured areas and has the same or better result on the standard well-textured images.

Bibliography / sources:

[1] Gaiani, Marco, et al. 'An advanced pre-processing pipeline to improve automated photogrammetric reconstructions of architectural scenes.' Remote sensing 8.3 (2016): 178.
[2] Ballabeni, Andrea, et al. 'Advances in Image Pre-processing to Improve Automated 3D Reconstruction.' International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences (2015).
[3] Rocco, Ignacio, Relja Arandjelović, and Josef Sivic. 'Efficient Neighbourhood Consensus Networks via Submanifold Sparse Convolutions.' ECCV (2020).

Name and workplace of bachelor’s thesis supervisor:

Ing. Michal Polic, Applied Algebra and Geometry, CIIRC

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **17.01.2021** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until: **30.09.2022**

Ing. Michal Polic	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt | Student’s signature

Acknowledgements

Above all who deserve it, I would like to express great appreciation to my supervisor Ing. Michal Polic, for support and help whenever I needed some. Of course, equally great thanks belong to my family, for they have created a perfect environment, where I did not have to worry about anything besides the thesis. And for their never-fading support in my studies. I also feel it appropriate to express gratitude to CTU, for without it being such a great university, this thesis would never have come into being.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, May 2021

Abstract

This thesis contains a closer look at comparison correspondence search methods used in 3D image reconstructions. Primarily at their ability to tackle reconstruction-problematic factors and areas that are decreasing the quality of reconstruction of given datasets.

Keywords: feature extraction, structure from motion, feature matching, 3D image reconstruction

Supervisor: Ing. Michal Polic

Abstrakt

Tato bakalářská práce se zabývá srovnáním metod pro hledání korespondencí mezi snímky, kteréžto metody jsou potom využity v 3D rekonstrukci. Především se práce zaměřuje na jejich schopnost vypořádat se s těžko rekonstruovatelnými fakotry a oblastmi, které snižují kvalitu rekonstrukce na daných datasetech.

Klíčová slova: feature extraction, structure from motion, feature matching, 3D rekonstrukce ze snímků

Překlad názvu: Porovnání metod pro hledání korespondencí mezi obrazy s málo texturovanými oblastmi — A jejich využití v 3D rekonstrukci z obrázků

Contents

Figures

Tables

Chapter 1

Introduction

The main goal of this work is a comparison of methods for image 3D reconstruction on specific scenes, which cause reconstruction problems even to state-of-the-art commercial software. The 3D image reconstruction algorithms are applied in various fields, *e.g.*, self-driving industry vehicles, google earth, the game and film industry, or augmented reality. These methods are also used by companies such as Google, Facebook, or Matterport. The reconstruction systems are also used in research projects such as SPRING, or ARTWIN.

1.1 Motivation and objectives

This bachelor thesis focuses on the comparison of different feature extracting and feature matching algorithms. This is motivated by the failures of the 3D reconstruction from images using the structure from motion (SfM) [10] and multi-view stereo (MVS) [11] on the specific datasets. The problematic features contained in these datasets are, *e.g.*, textureless areas, under/over exposed areas or illumination changes. One of the possible ways to improve 3D reconstruction results on challenging datasets is by choosing a more robust and accurate feature extractor and feature matcher. This approach is adopted in the thesis. Therefore we compare various methods of feature extraction (FE) and matching (FM) and their results in the 3D reconstruction pipeline. Another possible way to improve the reconstruction results would be by changing other parts of the reconstruction pipeline, such as triangulation, bundle adjustment, or whole incremental reconstruction. Nevertheless, these methods are used in the industry without any significant improvements over the recent years. On the other hand, feature extraction, description, and matching are being regularly improved [12]. Besides new hand-crafted feature extractors and matchers, does in the last years, these updates also include learnable neural networks (NN) [13] [14].

Popular pipelines providing 3D image reconstruction are COLMAP [10], MeshRoom (MR) [15], and Capturing Reality (CR). The cornerstones of these pipelines are SfM and MVS, where the latter builds on the results of SfM.

As mentioned before, failures of 3D reconstructions are caused by scene-specific factors, which are problematic even for state-of-the-art commercial software such as CR. The reconstruction issues arise with the presence of weakly textured, uniformly colored areas, scene transitions from dark to light (*e.g.*, the light goes on or off, or camera aperture changes in the dark corridor), or reflective and repetitive surfaces. Mentioned problematic scene properties often directly lead to low numbers of generated image matches. In addition, they cause poor match distribution in the image, as most of the challenging areas are almost matchless using classical methods, *e.g.*, a combination of SIFT [16] and geometric verification using RANSAC-based model estimator [17]. This lack of matches then negatively affects the accuracy of position estimates of registered cameras. The camera position inaccuracies then also decrease 3D point positions' precisions. The experiments in [18] shown the relationship between poor matching and higher scene uncertainty. Not to mention that poor matches reduce the number of registered cameras and often cause the model to break down into more sub-models, or, in the worst case, make the reconstruction completely fail, resulting only in one model with a small fraction of registered cameras or no registered images at all. Example of reconstruction failure on a dataset with problematic features s captured in Figure 1.1, which captures part of the reconstruction result, which broke into 3 models. This model used 5 out of 23 images, even though the images were obviously overlapping with the images from other models. The other models used 5 and 3 cameras The failure was probably caused by the illumination change combination with plain walls. Reconstruction was created with commercial software Capturing Reality.

The 3D image reconstruction algorithms are also used in industrial applications, *e.g.*, in research projects such as SPRING and ARTWIN. ARTWIN's main products include online factory digital 3D image and device localization. SPRING uses 3D reconstruction for device localization, too. These projects are mentioned because the industrial applications are especially sensitive to the reconstruction breakdowns and are full hollow spaces and long corridors. The corridors are further lined with weakly textured areas and illumination changes. These problematic interiors are a substantial part of the mentioned projects' application environment, so the reconstruction breakdowns are not rare. It often leads to loss of orientation, which is there a feature of utmost importance. Another reason to mention ARTWIN is that the thesis is created as a part of this project, and its results will find their use in the real industrial environment.

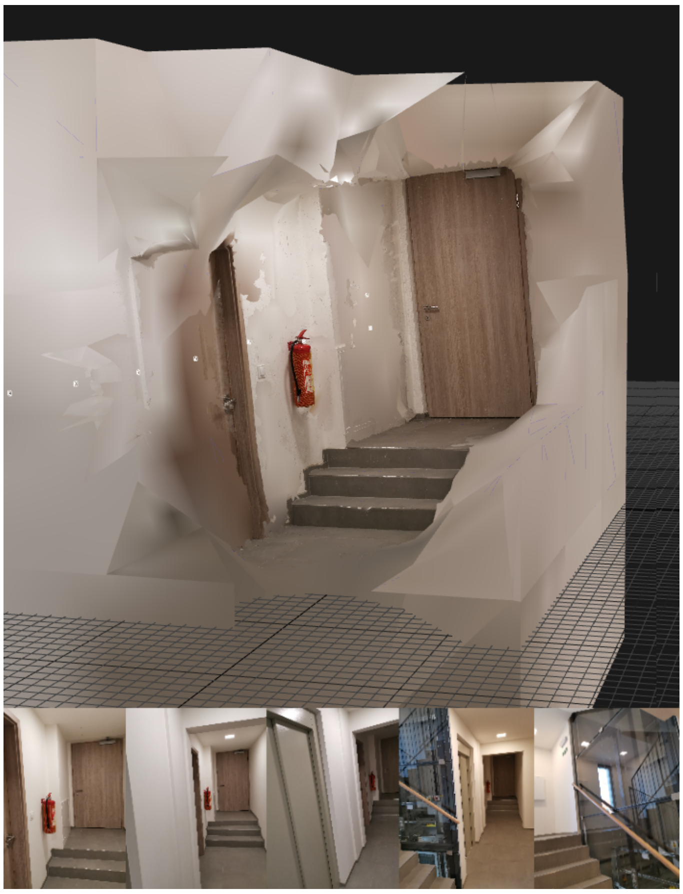


Figure 1.1: Reconstruction created with CR commercial software resulting in 5/23 registered images in this model.

Visualisation of the interior in which we are trying to improve 3D reconstruction is together with dense reconstruction captured in Figure 1.2. This dataset is from ETH3D benchmark [2] and is used for comparison in § 5.2.

1.2 Outline

Before evaluating and examining the individual FM and FE methods, the work begins with a definition of terms in the Chapter 2. Later in that chapter are introduced principles of SfM pipeline - correspondence search in § 2.1 and SfM itself in § 2.2. The § 2.1 includes details of previously used methods and algorithms, presenting both the hand-crafted state-of-the-art algorithms in § 2.1.1 and learnable algorithms introduced in § 2.1.2. These parts serve to unify terms and aim to pass on some basic understanding of principles of the FE, FM and SfM and methods parameters. These are then used and exmained in experiments, which are carried out in Chapter 4. Nonetheless,

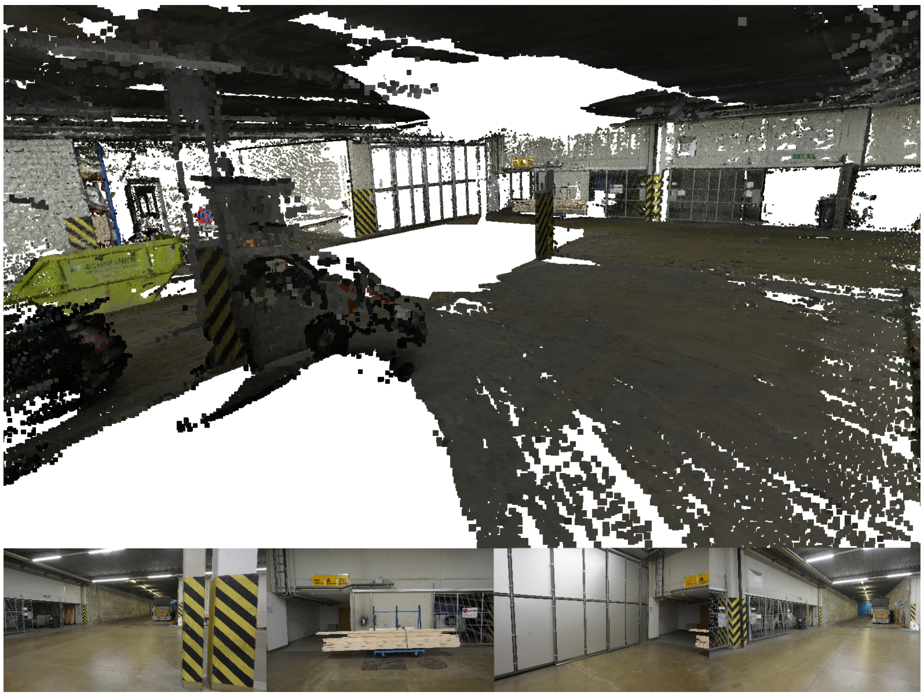


Figure 1.2: Example of environment, where are projects such as ARTWIN or SPRING applied. On the picture is the delivery area dataset from ETH3D datasets, which is then used in further comparisons. The dense reconstruction on the image was created with the baseline SIFT correspondence search combined with the COLMAP pipeline.

before the experiments is included Chapter 3, introducing shortcomings of one of the methods, the SparseNCNet. In the chapter are also proposed approaches to this issue in § 3.1.1 and we provide details about re-training of this network in § 3.1.2. Then, the Chapter 4 begins with collection of ground-truth data for FE and FM benchmarking, which is the described in § 4.2. Benchmarks are then used in experiments searching for best-performing parameters of SIFT in § 4.3, Wallis filter in § 4.5.1 and CLAHE in § 4.5.2, SparseNCNet in § 4.6, SuperGlue + SuperPoint in § 4.7 and comparison of MAGSAC++ and LO-RANSAC in § 4.4. The best-performing parameters and methods are eventually used in Chapter 5, which introduces the results of two benchmarks. One is evaluating purely FE and FM performance in § 5.1 and the second results of complete 3D reconstruction pipeline with use of ETH3D benchmark in § 5.2.

1.3 Codes

All codes used in this thesis are attached in file `codes.zip`. These are also available in Gitlab repository at [19].

Chapter 2

Previous work

At the beginning of this section are defined the words frequently used in this thesis. The definitions are included, as different authors assign the words slightly different meanings. The words are: features, feature detection, feature description (these two together are referred to as feature extraction), feature matching, Structure from Motion (SfM) and Multi View Stereo (MVS). The introduction of keywords is followed by a closer look at principles of the feature extraction and matching algorithms, including the state-of-the-art methods. Eventually, we describe the usage of these approaches in structure from motion and the SfM pipeline itself.

The feature is a distinguishable local area of interest in the image. Therefore it can be uniquely described and matched across several images. This area, often a local neighborhood of point, corner or blob (area with shared property among pixels, such as color, see Figure 2.1), can be described by a vector descriptor, which should be invariant to rotation, translation, scale and illumination changes in the ideal case. Even though images often capture the same object, they are taken from different angles and positions. Thus they are subject to rotation and translation. We need it to be invariant because otherwise, we would not be able to create correct correspondences among features in given images.

Feature extraction is a process that takes an image as an input and returns a set of features with descriptors. At first, it detects points of interest, *i.e.*, points that are unique and easily recognizable in the given image. This process is called *feature detection*. Next follows *feature description*, which creates a mathematical description (descriptors) of the detected points of interest. The result is a set of features. Feature extraction is captured together with feature matching in Figure 2.2.

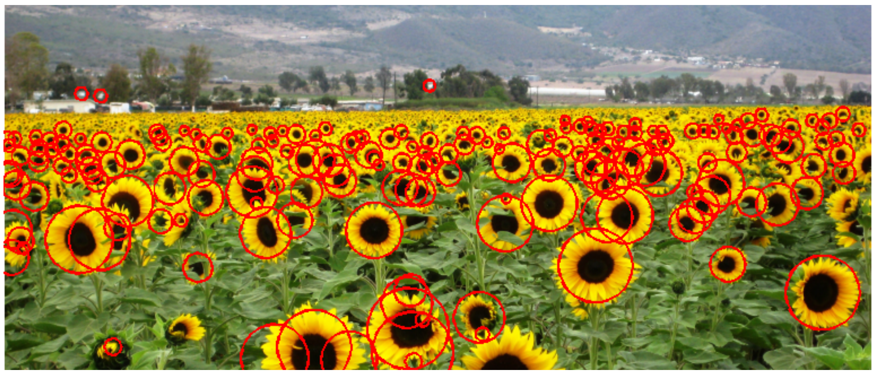


Figure 2.1: Detected blobs in sunflower image obtained by detecting Hessian maxima in scale-space. Image taken from [1].

Feature matching stage provides a comparison of descriptors across given images. As a result of this comparison, it identifies similar features. The output of this stage is a set of *tentative matches*. These are based solely on appearance and are not geometrically verified. Matched features are captured in Figure 2.2.

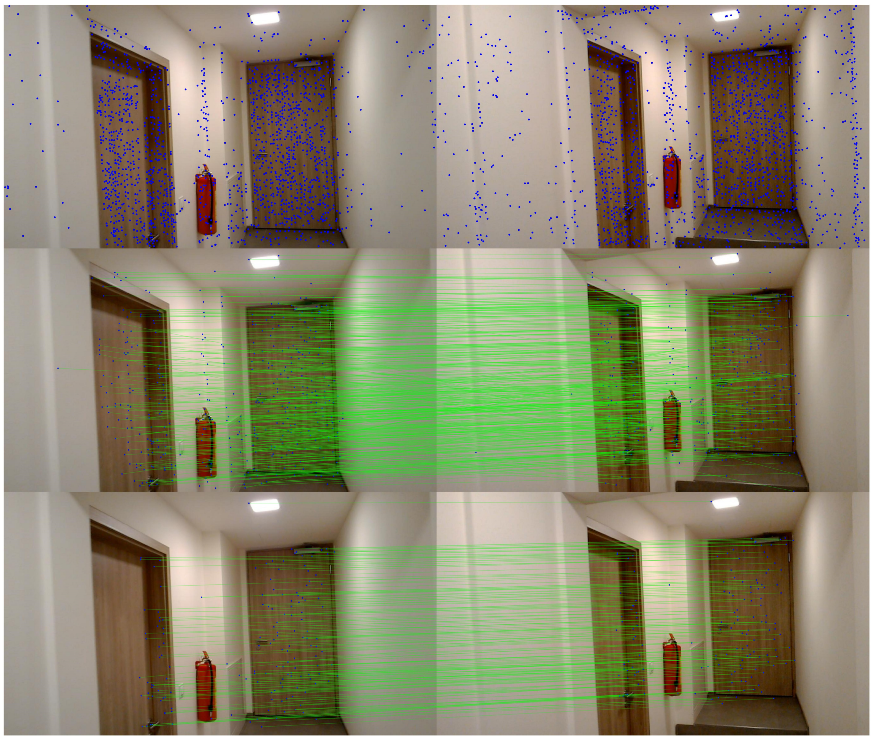


Figure 2.2: Visualisation of feature matching stages. From top to bottom, we visualised outputs from *feature extraction*, *feature matching* and *geometric verification*.

Geometric verification stage provides filtration of incorrect matches by computing transformation between two matched images. Matches corresponding to this transformation are considered valid.

Structure from Motion is an algorithm utilizing series of two-dimensional images and creating a three-dimensional structure of the object or scene captured in the images. The result of this process is a sparse 3D point cloud. SfM is contained in pipelines as COLMAP and Meshroom, which are not commercial, and as a representative of commercial pipelines, we name Capturing Reality.

Multi view stereo is a successive step to SfM providing a dense 3D point cloud similar to the one created by LIDAR (method of measuring ranges with the use of a pulse laser). However, LIDAR is able to measure the points even on textureless areas. Visualization of such point cloud is shown in Figure 2.3.

Feature detection, description, and matching are not used exclusively in 3D image reconstruction, which is the usage we are studying in this work. Other applications include panoramic image stitching [20], image retrieval and visual localization [21], *i.e.*, pose estimation in large scale environment [22], or object detection and instance segmentation [23].

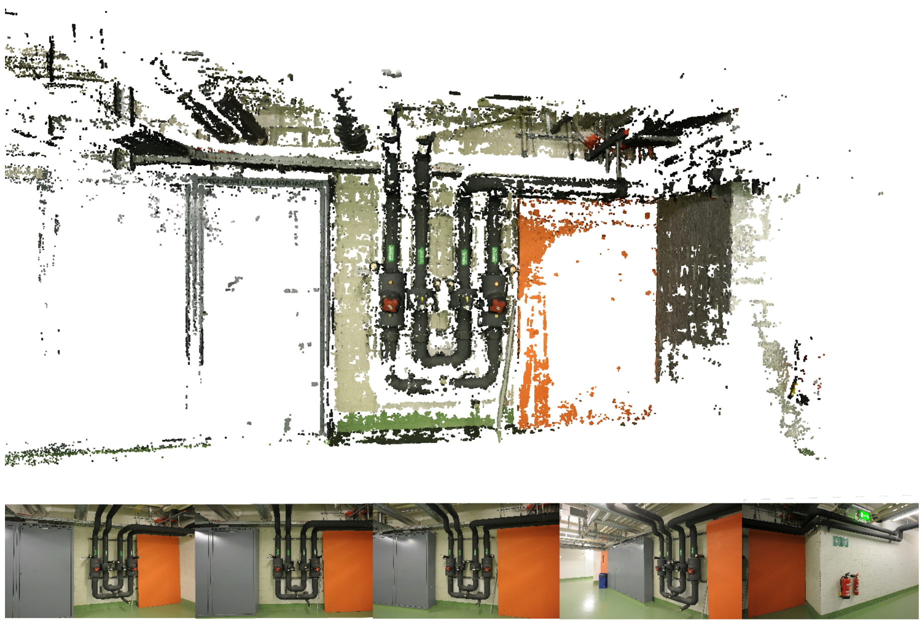


Figure 2.3: Visualisation of dense point cloud produced after MVS. Images used to create reconstruction are below point cloud. The images are part of ETH3D dataset [2]. This dataset is also used in benchmark in § 5.2.

2.1 Feature extraction and matching

Feature extraction and matching generally begins with a set of N_I images $\mathcal{I} = \{ I_j \mid I_j \in \mathbb{R}^{H \times W \times cc}, j = 1..N_I \}$, where $H \times W$ is the resolution of images, and cc is a number of color channels. We mostly start with RGB images, where the $cc = 3$. Then, there are two most common approaches to feature extraction and matching.

The first approach is employs hand-crafted mathematical algorithms. In this case, all of the steps are separable, and their results can be assigned unambiguous meaning. Nevertheless, all of the stages can be provided in a single algorithm.

The second, trainable, approach is based on neural networks (NNs). This approach has recently gained popularity in many research fields - because it can achieve excellent results in a wide range of tasks in lots of different fields besides computer vision. These are, *e.g.*, language translation, semantic segmentation, or automatic control. This approach does not necessarily need to be composed of more separable stages, but some of the algorithms are. However, we often can not assign an unambiguous mathematical interpretation to the outputs of the network layers or even individual stages. In the case of hand-crafted algorithms, the meaning of each parameter is connected to some mathematical property such as rotation. However, in the case of the NNs, we may not be able to say anything about the meaning of the numbers, *e.g.*, the vector descriptors.

2.1.1 Hand-crafted algorithms

Hand-crafted algorithms often begin with the feature extraction process. There, the algorithm selects for every image $I_j \in \mathcal{I}$ a set of N_{F_j} features $F_j = \{ (\mathbf{x_{ij}}, \mathbf{f_{ij}}) \mid \mathbf{x_{ij}} \in \mathbb{R}^2, i = 1...N_{F_j} \}$, where $\mathbf{x_{ij}}$ are coordinates in image I_j and $\mathbf{f_{ij}}$ is a related feature descriptor. Size of feature descriptor depends on the feature description algorithm, common dimensions are $\mathbf{f_{ij}} \in \mathbb{R}^{128}$, $\in \mathbb{R}^{256}$ or $\in \mathbb{R}^{512}$. The features are selected by finding the local extrema of some feature selection function. These extrema are, in short, points with sufficient color or pixel intensity gradients in the neighborhood of the points. Then follows a description of the points with vector descriptors. The descriptors can capture the color or intensity gradients, rotation or other description-algorithm-specific parameters to a vector. These two steps can be covered together in one algorithm. The output of such a merged approach is then directly a set F_j. One of the most cited feature detectors and descriptors is SIFT (Scale-invariant feature transform) descriptor [16], which is a standard in the industry for a long time and has been used in many applications, *e.g.*, mobile robot localization [24], or morphometry [25]. SIFT algorithm works with gray-scale images. As our datasets ar collections mostly of RGB images, a conversion of $I_j \in \mathbb{R}^{H\times W\times 3}$ to $I_j^{(G)} \in \mathbb{R}^{H\times W}$ is needed.

The basic component of feature selection function in this algorithm is the scale-space defined with the notation from [3] as a function $L(x, y, \sigma)$. Scale-space is produced by the convolution of an input image $I_j^{(G)}$ with a Gaussian kernel $G(x, y, \sigma)$ with a variable scale σ. The convolution $*$ is performed in 2D image spatial coordinates (x, y). The scale space of the image is therefore defined as

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

The feature selection function is Difference of Gaussians (DoG), which is approximation of Laplacian of Gaussian, defined as

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma). \tag{2.2}$$

It is a difference of two adjacent scales $L(x, y, \sigma)$ differing by a constant multiplicative factor k. The x and y are spatial coordinates in the image.

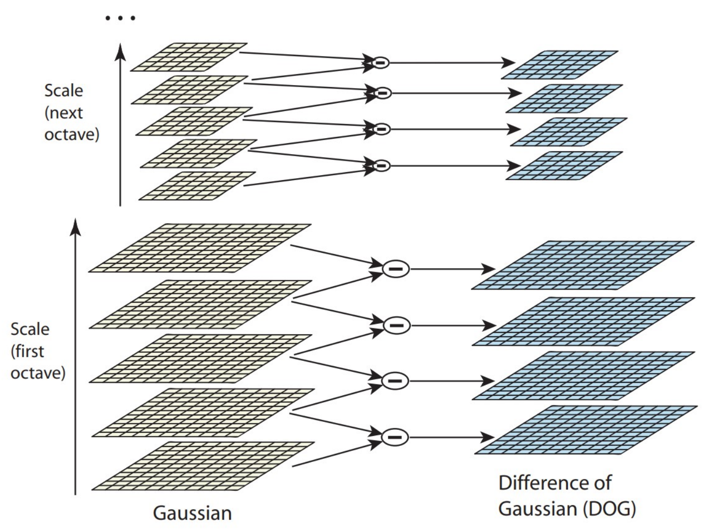


Figure 2.4: Initial image is in each octave convolved with Gaussians. Thus are produced the sets of scale space images (on the left). Then are adjacent convolved images subtracted and produce difference-of-Gaussian images (on the right). Image was taken from [3].

The scale space is sectioned into octaves, where in each octave is the image reduced in size by a factor of two. The Eq. 2.2 is repeated for each adjacent pair of scales in each octave, as is depicted in image Figure 2.4. Than the results $D(x, y, \sigma)$ of Eq. 2.1 are searched for extrema. Each scale-space point is compared to its eight neighbors in the current scale as well as to nine pixels in next scale and nine pixels in previous scale, as shown in Figure 2.5. Local extrema are potential keypoints. However, not all extrema are stable or sufficiently contrastive, so they have to be filtered to get more accurate results.

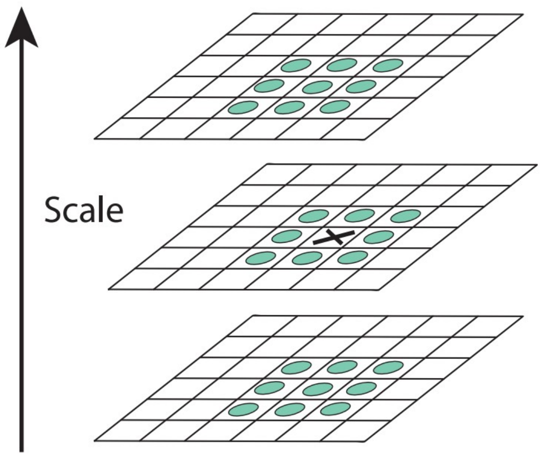


Figure 2.5: Extrema of the DoG images are detected by comparing a pixel (X) to ts eight neighbors in the current scale as well as to nine pixels in next scale and nine pixels in previous scale (circles). Image was taken from [3].

First, with the use of second-order Taylor series expansion of scale-space $D(x, y, \sigma)$ is found extrema location with subpixel precision. Then, if the intensity at this extrema is smaller than a defined parameter *peak threshold*, the point is rejected. The DoG function is responsive to edges, and the edge points are often not uniquely describable. Therefore, edge keypoints need to be filtered, too. For this is used an approximation of 2×2 Hessian matrix $\mathbf{H}$ of $D(x, y, \sigma)$, where the two dimensions used are x and y coordinates. The derivatives in Hessian are estimated by taking differences of adjacents scale-space points. The eigenvalues λ_1 and λ_2 of $\mathbf{H}$ are proportional to the principal curvatures of D. Therefore, eigenvalues can be used to determine whether the extrema is an edge. If the ratio of the eigenvalues is greater than a parameter *edge threshold*, that keypoint is probably on edge and is discarded. After the filtration, the remaining points are considered strong interest points.

The keypoint descriptors of SIFT algorithm determine a location, scale, and orientation. These are obtained using histograms, which are computed from orientations and size of intensity gradients in the neighborhood of the described keypoints. The SIFT results in set of features F_j corresponding to image I_j, with descriptors $\mathbf{f_{ij}} \in \mathbb{R}^{128}$. An example of SIFT described features is visualized in Figure 2.6.

There is a similar alternative to SIFT called SURF, which utilizes approximation of the determinant of Hessian blob detector. Note that SIFT is slow in comparison with younger methods [12], such as MSER [26], FAST [27] or ORB [28]. MSER extracts the regions of interest using the sequence of thresholded images. FAST compares pixels on a circle of 16 pixels (with radius 3) centered at the feature point. ORB is based on FAST and combines it with feature descriptor BRIEF [29]. Mentioned algorithms are compared by Miksik *et al.* in [12].

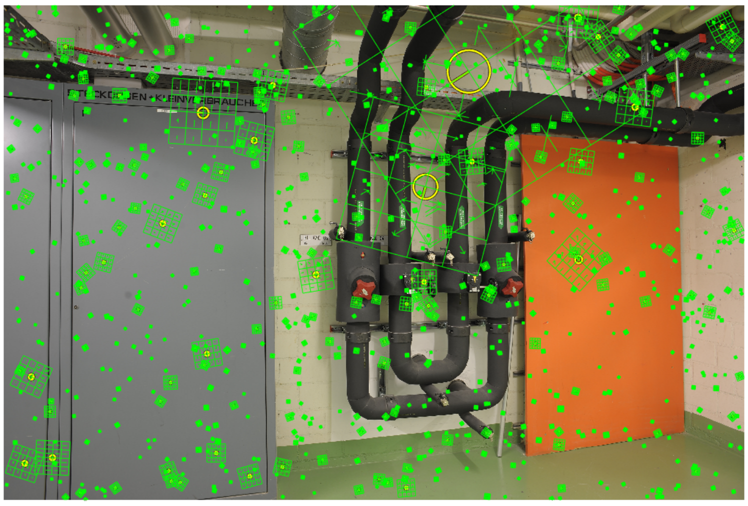


Figure 2.6: Visualisation of features detected and described by SIFT. picturing feature orientation, position and histogram grid, pictured using library VLFEAT [4].

As mentioned, hand-crafted algorithms are primarily based on the gradient approach, so they are susceptible to color or intensity gradient changes. On texture-less areas are gradients too small to create uniquely described feature points, and therefore the matching, the next step of reconstruction, is usually very challenging. The hand-crafted feature extractors have shown excellent results on images taken with significant viewpoint changes, with almost constant illumination. However, their robustness to illumination changes is not as good. This is shown, *e.g.*, in [30] or [31].

The step following feature extraction involves a comparison of the obtained described features for defined image pairs. The comparison step is utilizing the similarity metrics applied to the descriptors. These metrics can be L1, or L2 norm [32]. The naive approach compares every two features in each image pair. Aside from the naive algorithms, most common approaches are based on types of the nearest neighbor search, *e.g.*, FLANN [33]. After matching by descriptors, we obtain so-called tentative matches. These matches are based solely on appearance, and it is not guaranteed that all are correct correspondences of one 3D point projected to two 2D images. Therefore, we should geometrically verify them to find correct matches.

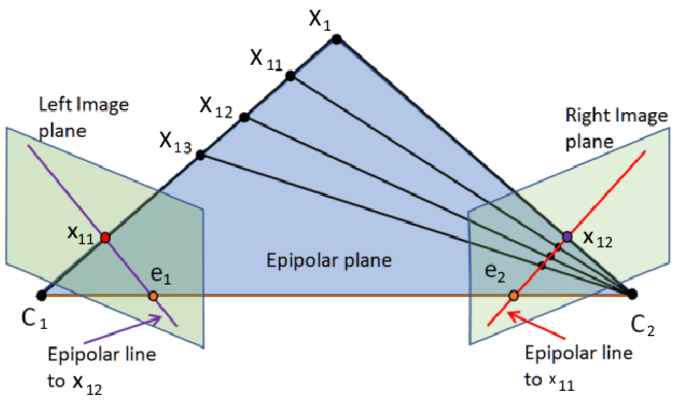


Figure 2.7: Diagram of epipolar geometry. Half line going through camera center C_1 and point x_{11} in left image plane corresponds to possible positions of 3D point X_1 corresponding to x_{11}. Projection of this line to right image plane creates epipolar line. Image taken from [5].

The verification of the matches is performed by estimating a relative pose. The first step is about assembling equations using epipolar geometry, which projects points in one image to the lines in the second image. The task of computation of the transformation parameters is called relative pose estimation. Epipolar geometry without radial distortion diagram is shown Figure 2.7.

Let's suppose we have two images, I_1 and I_2, where $\mathbf{x_{11}}$ and $\mathbf{x_{12}}$ are projections of 3D point $\mathbf{X_1}$ into the two images respectively. Their relation can be described by *fundamental matrix* $\mathbf{F} \in \mathbb{R}^{3\times3}$ as

$$\mathbf{x_{11}^T F x_{12}} = \mathbf{0}. \tag{2.3}$$

As we are working with epipolar geometry, the rank of the fundamental matrix $\mathbf{F}$ is two. Therefore, the part $\mathbf{Fx_{12}}$ represents a line in image I_1. The left side of Eq. 2.3 is then distance of $\mathbf{x_{11}}$ from this line. A fundamental matrix is used in the case of unknown internal parameters of the camera. In case we know the camera intrinsic parameters before reconstruction, we apply calibrated matrix $\mathbf{E} \in \mathbb{R}^{3\times3}$, called *essential matrix* and for which

$$\mathbf{E} = (\mathbf{K}')^T\mathbf{FK}, \tag{2.4}$$

where $\mathbf{K}, \mathbf{K}' \in \mathbb{R}^{3\times3}$ are the calibration matrices composed of the internal parameters of the cameras. When we use the same physical camera for both images, there is only one calibration matrix $\mathbf{K}$. The computation can be further extended by trifocal tensor $\mathbf{T} \in \mathbb{R}^{3\times3\times3}$, which incorporates all geometries among three views observing the same point.

The relative pose is estimated using RANSAC-based algorithms. In short, it iteratively selects random subsets of the set of points and uses solvers like f10e [34] to fit a model to assembled equations. The most feature-supported model is considered correct, and the corresponding matches are then declared valid. Feature-support is expressed with a number of inliers, which are points closer to the epipolar lines than the threshold set as a parameter. Matches not compatible with the transformation are dismissed. In later comparison in § 4.4, we use LO-RANSAC [35] and state-of the art algorithm MAGSAC++ [36] to cover robust model estimation.

2.1.2 Learned algorithms

The second approach to feature extraction and matching utilizes trainable algorithms, usually neural networks. However, even among NN-based feature extractors and matchers can be found a broad palette of solutions, including combinations of hand-crafted algorithms and convolutional neural networks (CNNs). CNNs are often the choice number one in image processing. One of the most common approaches amidst image processing pipelines is encoder-decoder architecture. Networks with this structure use encoders as the first step. The encoders gradually reduce the image's dimensionality and encode information about the image's features into channels instead of spatial dimensions. Such architecture can be seen in the left part of Figure 2.8. The encoders consist of convolutional layers, spatial downsampling performed by max-pooling, and activation functions. Commonly used encoders are, *e.g.*, VGG [37] or Resnet [38]. The output of the encoder can have various purposes of use. This purpose is assigned by a decoder, which then processes the final output or output from some of the previous layers of the encoder. *E.g.*, the decoder in SegNet [39] can label encoded features by categories they belong to. This is called image segmentation. State-of-the-art encoder-decoder pipeline for image-segmentation is YOLO [40]. Another decoder-performed action is description of encoded features, which is done within a network called Superpoint [41], D2-Net [42] or R2D2 [14]. Note that these architectures contain their own encoders.

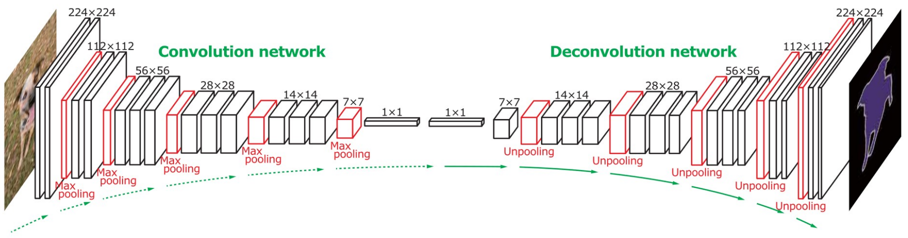
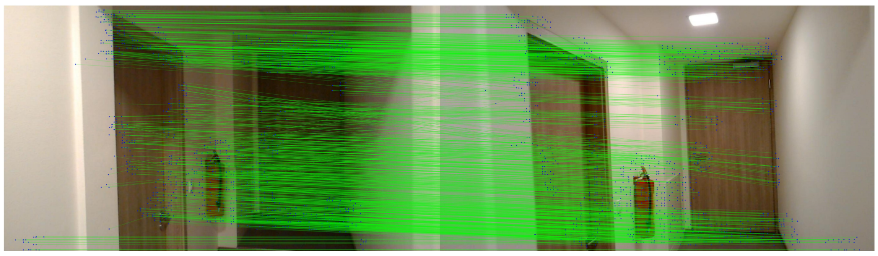


Figure 2.8: Encoder-decoder architecture used in image segmentation. Image taken from [6]

This introduction suggests that one possibility of the NN use in the correspondence search is to perform separate feature extraction. The features can then be matched by hand-crafted algorithms or by networks trained purely for feature matching, as shown by Ranftl *et al.* [43], Yi *et al.* [44] or in Superglue [13]. We have chosen the latter one in combination with SuperPoint to our comparison as the representative of this category. Superglue also takes into account the context in the images and not purely the descriptors. Another possibility for the usage of NNs trained for feature matching could be to match features extracted by some hand-crafted feature extractor, SIFT or ORB. Results of these methods in comparison with NN extraction combined with NN matching are shown in [13].

One of the other types of NNs combines all of the three steps, feature detection, description, and matching, into one end-to-end-trainable correspondence searching network. This approach is practiced by R2-Net [45], NCNet [46] or SparseNCNet [7]. The latter achieves the best results [7]. Thus we have decided to use it as the representative of end-to-end trainable approaches. Visualization of tentative matches produced by SparseNCNet can be seen in Figure 2.9a. Some networks claim to produce already verified matches - including SparseNCNet and SuperGlue. However, the features and matches produced by NNs could carry significant errors. Thus, in order to use them in 3D reconstruction, we apply geometric verification in this case, too. Matches produced by SparseNCNet and verified by MAGSAC++ are shown in Figure 2.9b.


(a) : Matches generated by SparseNCNet without geometric verification.

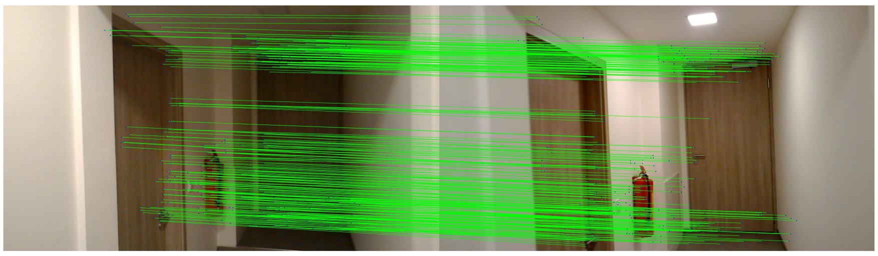
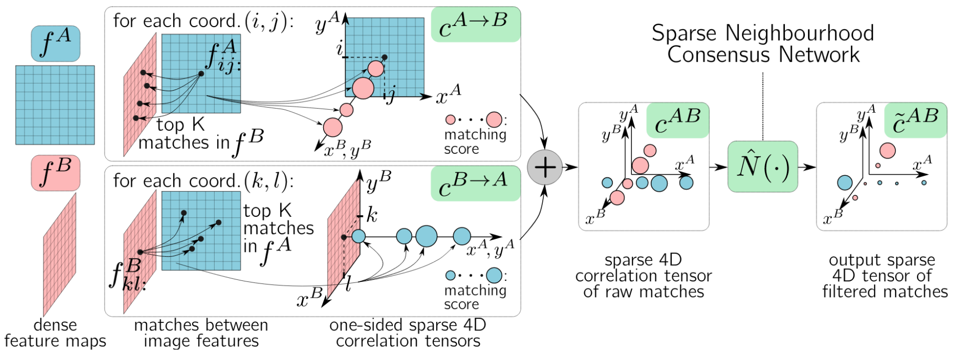

(b) : Matches generated by SparseNCNet verified by MAGSAC++.

Figure 2.9: Comparison of geometrically verified matches produced by SparseNC-Net with its matches without geometric verification.

As we have chosen to test SparseNCnet as main NN representative in this work, we present a summary of this network's principles using notation from [7].

SparseNCNet architecture diagram is captured in Figure 2.10. This network is faster and less memory demanding enhancement of its predecessor, NCNet. Nevertheless, they share many features.



.

Figure 2.10: Visualisation of SparseNCnet architecture, image taken from [7]

This algorithm consists of several internal stages. First of which is feature extraction. Input to this stage is a pair of RGB images $I_A, I_B \in \mathcal{I}$. This step results in L2-normalized dense feature maps $f_A, f_B \in \mathbb{R}^{h\times w\times c}$, which are extracted via a fully convolutional network $F(\cdot)$, where

$$f_A = F(I_A), \quad f_B = F(I_B). \tag{2.5}$$

The c means channels of features and $h \times w$ is a spatial resolution of the feature maps. This resolution is obtained after downsampling the input images by the factor 1/8 or 1/16. The fully convolutional network $F(\cdot)$ used as feature encoder in the implementation published with the SparseNCNet paper is Resnet101 [38]. The dense features f_A and f_B are then compared and stacked into *sparse correlation tensor* $c^{AB} \in \mathbb{R}^{h\times w\times h\times w}$. The *sparse correlation tensor* is defined as a sum of two *one-sided sparse correlation tensors*,

$$c^{AB} = c^{A\to B} + c^{B\to A}. \tag{2.6}$$

The *one-sided sparse correlation tensor* $c^{A\to B}$ holds for every feature from one image only the top K nearest neigbors from the second image (this is one of the differences to the NCNet). Specificaly, each feature descriptor f^A_{ij} on coordinates (i,j) in feature map f^A of image I_A is matched with only K nearest descriptors f^B_{kl} lying on coordinates (k,l) of f^B from image I_B. This algorithm is not storing all the possible matches to achieve fastest and less memory demanding performance. The coefficients of tensor $c^{A\to B}$ are defined as

$$c^{A\to B}_{ijkl} = \begin{cases} \langle f^A_{ij}, f^B_{kl}\rangle & \text{if } f^B_{kl} \text{ is in } K \text{ top NNb of } f^A_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

and the similar applies for $c^{B\to A}$. In the following step is applied convolutional neural network $\hat{N}(\cdot)$ on the correlation tensor. $\hat{N}(\cdot)$ consists of applying twice a sparse 4D convolutional network $N(\cdot)$ in order to get matches invariant to the order of the input images. The final correlation tensor is then

$$\tilde{c}^{AB} = \hat{N}(c^{AB}) = N(c^{AB}) + (N((c^{AB})^T))^T, \tag{2.8}$$

where by transposition is meant exchanging the first two dimensions with the last two dimensions. The purpose of network $N(\cdot)$ is to perform the neighbourhood consensus filtering. Then we compute matches as

$$m = ((i,j),(k,l)) \text{ is a match if } \begin{cases} (i,j) = \operatorname{argmax}_{(a,b)}(\tilde{c}^{AB}_{abkl}) \text{ or} \\ (k,l) = \operatorname{argmax}_{(c,d)}(\tilde{c}^{AB}_{ijcd}) \end{cases}, \tag{2.9}$$

where (i,j) are coordinates running in all the size of feature map f_A, and (k,l) in the size of f_B. Note, that this argmax operation is applied on the rightmost picture in diagram Figure 2.10. As the individual features are localized on the feature maps, which have the spatial resolution of a fraction of the input images, the error of the localization is so significant, that in this stage, the results would be inapplacble in the 3D reconstruction and in most of the applications, too. Therefore, there is included relocalization as a following step, to enhance accuracy of localization of the matches on the high resolution image. The principle of relocalization is in upsampling of the input images as the very first step. After applying convolutional network $F(\cdot)$, we obtain feature maps $\hat{f}_A$, $\hat{f}_B \in \mathbb{R}^{2h\times 2w\times c}$. Then after applying a 2×2 max-pooling operation with a stride of 2 we obtain f_A and f_B. With subsequent application of the aforementioned procedure we receive set of matches $m = ((i,j),(k,l))$. Now we can perform *hard relocalization*, which finds best matches in 2×2 sized patches in $\hat{f}_A$ with corresponding 2×2 from $\hat{f}_B$. The corresponding 2×2 patches are the ones, in which were located values on indices $((i,j),(k,l))$ matched in m before downsampling. Then we obtain relocalized matches

$$m_h = 2m + ((\delta i, \delta j), (\delta k, \delta l)), \tag{2.10}$$

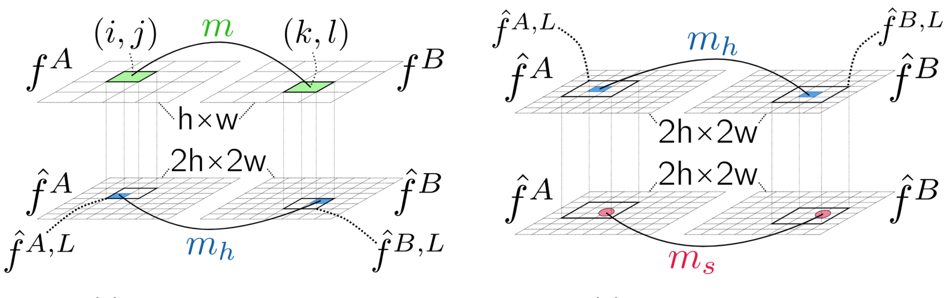
where $((\delta i, \delta j), (\delta k, \delta l))$ symbolize indices in the 2×2 patches from $\hat{f}_A$ and $\hat{f}_B$ respectively. The last step is *soft relocalization*, where samples from $\hat{f}^{A,L}_{ij}$, $\hat{f}^{B,L}_{kl}$ of size 3×3 around both feature indices $((i_h, j_h),(k_h, l_h))$ for each match from m_h are taken (in m_h are indices $((i_h, j_h),(k_h, l_h))$ already in $2w\times 2h$ range). Eventually, there is applied a softargmax function on these patches which results in sub-pixel precision, the final matches (with floating point indices) are computed as

$$m_s = m_h + \Delta m_s, \tag{2.11}$$

where

$$\Delta m_s = ((\delta i_h, \delta j_h), (\delta k_h, \delta l_h)) \begin{cases} (\delta i_h, \delta j_h) = \operatorname{softargmax}_{(a,b)}(\langle f^{A,L}_{ab}, f^{B,L}_{k_h l_h}\rangle) \\ (\delta k_h, \delta l_h)) = \operatorname{softargmax}_{(c,d)}(\langle f^{A}_{i_h j_h}, f^{B}_{cd}\rangle) \end{cases}. \tag{2.12}$$

The coordinates (a,b) runs over 3×3 patch of $f^A_{i_h j_h}$ and (c,d) over patch of $f^A_{ik_h l_h}$. Whole relocalization process is fore better understanding visualized in Figure 2.11.



(a) : Hard relocalization. **(b) :** Soft relocalization

Figure 2.11: SparseNCNet relocalization stages - image from [7].

2.1.3 Pre-processing

Image pre-processing algorithms are applied to enhance the suitability of the images for subsequent utilization. Image pre-processing is used in wide range of applications, *e.g.*, medicine [47] or photogrammetry [48]. Gaiani *et al.* [8] address the issue of the lack of correspondences on texture-less and issue caused by significant illumination changes by pre-processing. Both of the issues cause 3D reconstruction failures. The proposed approach provides greater detail in over-exposed and under-exposed areas simultaneously. Specifically, they have shown significant improvements in 3D reconstruction with the use of pre-processing pipeline shown in Figure 2.12. Generally, pre-processing methods take the set of input images $\mathcal{I}$ and apply mathematical operations, *e.g.*, convolution with Gaussian kernel or RGB to grayscale conversion on each image $I_j \in \mathcal{I}$. The pre-processing is applied before the feature extraction begins or before the MVS stereo step in order to find better correspondence between images. The main goal of the algorithm in [8] is to improve dense image matching results. This strategy employs three steps to achieve the improvement. First, to increase the number of correct image correspondences in textureless areas. Then maximize the track length to increase the reliability of the computed 3D coordinates and correctly orient the largest number of cameras [8].

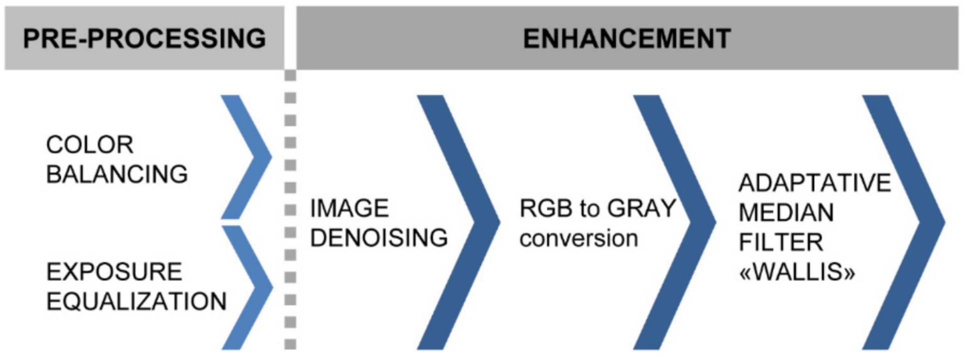


Figure 2.12: Image preprocessing pipeline used in [8].

The pipeline begins with global color enhancements, ensuring consistency of subsequent color-to-gray conversion. However, this stage aims at improvements mainly during dataset acquisition. The datasets are collected with GretagMacbeth ColorChecker [49] present in images at least after each change of environment or lighting conditions. However, our datasets have been collected without ColorChecker.

Color enhancements are followed by denoising. In this paper were compared several hand-crafted denoising algorithms $D(\cdot)$. The image denoising is applied to input image I_j, as the color enhancement stage is skipped. The output is then

$$I_j^{(D)} = D(I_j). \tag{2.13}$$

The proposed approach applied in the denoising stage is called the Color Block Matching 3D filter (CBM3D), which is an extension of the Block Matching 3D filter (BM3D). BM3D is a sliding-window-based algorithm and extends the Discrete Cosine Transform [50] and NL-means algorithms [51]. Note, that hand-crafted denoising approaches such as BM3D have been recently outeprformed by denoising NNs [52], *e.g.*, FfdNet [53] or CycleISP [54].

After denoising, images are converted from RGB space to grayscale, as

$$I_j^{(G)} = r2g(I_j^{(D)}). \tag{2.14}$$

Gaiani *et al.* proposed a conversion method called Bruteforce Isoluminants Decrease (BID) based on *Matlab rgb2gray* formula.

As the last step is employed image content enrichment. In Gianini *et al.* pipeline is this provided by Wallis filter [55]. Wallis filter is a local adaptive median filter, *i.e.*, it adjusts pixel brightness values in local areas. In contrast, a global filters use the same contrast values throughout an entire image, and therefore can not enhance details in both high- and low-level-of-brightness areas simultaneously. The inputs of Wallis filter are besides the denoised gray-scale input image $I_j^{(G)}$ also desired standard deviation M and desired mean S. Moreover, Wallis filter has weight parameters A and B which control how much are the computed values propagated to the output image.

The Wallis filter formula proposed in [8] is

$$I_j^{(w)}(x,y) = \frac{S I_j^{(G)}(x,y)}{s+A} + BM + (1-B)m, \tag{2.15}$$

where $I_j^{(w)}$ is pre-processed output image, s is the local mean, and m is the local standard deviation. Wallis filter can be seen as a sliding window of the user-defined size W. The amount of detail in the final image is inversely proportional to the size of the window. Nonetheless, the values m and s are computed for the window centred in (x, y).

A similar locally adaptive filter to Wallis is the Contrast Limited Adaptive Histogram equalization (CLAHE) filter, which we also use in the later comparison. It is based on adaptive histogram equalization (AHE). AHE transforms each pixel using the histogram of a surrounding local square patch of the pixel. This transformation is derived from histogram equalization. Histogram equalization is, in short, mapping the patch histogram onto a different scale in order to create an equally distributed histogram. The desired uniform distribution could be replaced by exponential or Rayleigh distribution. CLAHE adds a limit contrast level, limiting the noise amplification by clipping the histogram at a predefined level. The level is set as parameter *clip limit*. Another parameter is *tile number*, which defines the number of tiles the input image is divided to.

Both of the methods used for local contrast and detail enhancements are shown as in the Figure 2.13.

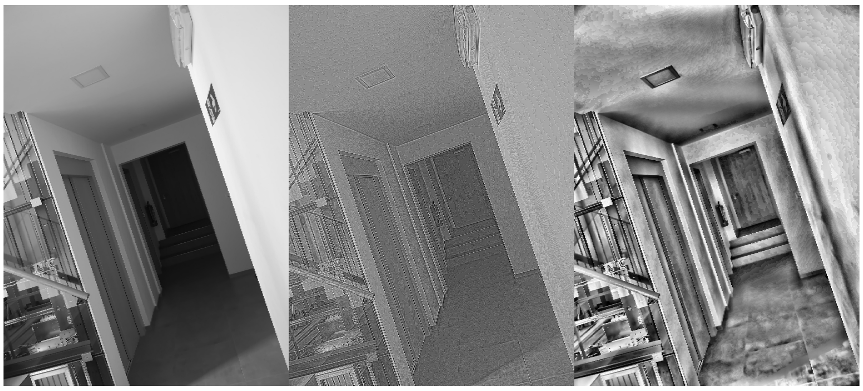


Figure 2.13: From left: Original image, image after application Wallis Filter and image after application after CLAHE filter.

2.2 Structure-From-Motion

The Structure from motion is a method used to create a sparse 3D model from series of 2D images. These images can be seen as projections of the original 3D objects into 2D from different viewpoints. The correspondences among images are called *tracks*. As the images capture the same scene from different angles, some 3D points are expected to be present in more images. The output of this pipeline is a 3D structure in the form of colored point cloud and camera poses, similar to the point cloud shown in Figure 2.14. In other words, outputs are set of N_X points $\mathcal{X} = \{ \mathbf{X_i} \mid \mathbf{X_i} \in \mathbb{R}^3, i = 1..N_X \}$, each of them with assigned color, and camera poses $\mathcal{P} = \{ \mathbf{P_j} \in \mathbb{R}^{4\times4} \mid j = 1..N_I \}$, where $\mathbf{P_j}$ is a homogeneous transformation matrix, *i.e.*, $\mathbf{P_j} \in \mathbf{SE(3)}$, which includes rotation matrix of camera $\mathbf{R_j} \in \mathbb{R}^{3\times3}$ and coordinates of camera center $\mathbf{C_j} \in \mathbb{R}^3$. This is a mathematical model which can, but doesn't have to consider radial distortion. In our model is radial distortion not considered. With $\hat{\mathbf{x}}_{\mathbf{ij}} \in \mathbb{R}^2$ being projection of i-th 3D point $\mathbf{X_i}$ to the j-th image I_j applies

$$\lambda \begin{bmatrix} \hat{\mathbf{x}}_{\mathbf{ij}} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{P_j} \begin{bmatrix} \mathbf{X_i} \\ 1 \end{bmatrix}, \tag{2.16}$$

where λ coefficient secures the up-to-scale configuration on the left side, *i.e.*, to secure that the $\hat{\mathbf{x}}_{\mathbf{ij}}$ is in the image plane, where the third value of projected vecotr equals one.

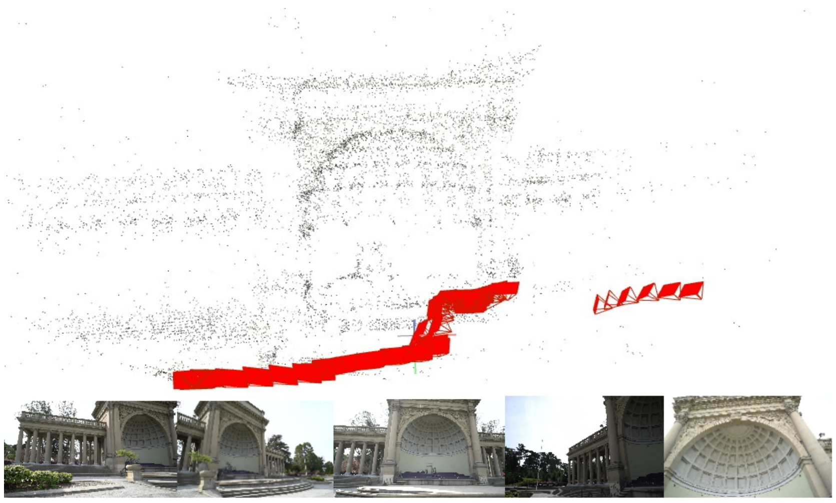


Figure 2.14: Sparse 3D reconstruction output in the form of colored point-cloud produced by SfM. The images from which the reconstruction was created are under the reconstruction. There can also be seen reconstructed camera poses in red. Images are from Tanks and Temples benchmarking dataset [9].

The reconstruction process consists of several sub-stages. It begins with the *correspondence search*, which includes together feature extraction, matching, and geometric verification. These are principles discussed in § 2.1. The subsequent stage is the reconstruction itself. The reconstruction approach covered here is composed of relative pose estimation, which is estimated already during geometric verification, then absolute pose estimation, point triangulation, and bundle adjustment [10], all applied gradually for each camera.

The reconstruction principles are discussed as the main aim of this work is to compare approaches to correspondence search in order to maximize the quality of 3D reconstruction. Thus, it should be clarified how the reconstruction methods utilize the correspondences. The reconstruction pipeline used in experiments is COLMAP, so the principles below are consistent with the methods used in this pipeline. SfM pipeline used by COLMAP is captured

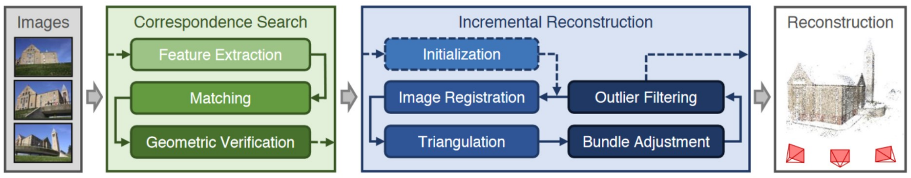


Figure 2.15: Architecture of Structure-from-Motion pipline, used in COLMAP. Image taken from [10].

in Figure 2.15. Nonetheless, this pipeline is common for most of the SfM softwares.

An inseparable component of computation of the 3D model is camera calibration, which is the determination of intrinsic parameters. The reconstruction may be highly imprecise, even with slightly wrong intrinsic parameters. Generally, we can come across two main approaches to camera calibration. In first, the parameters are determined before the reconstruction. Therefore, calibration matrix $\mathbf{K}$ is known, and as the estimated transformation remains essential matrix $\mathbf{E}$. In the other approach are the parameters computed before the extrinsic parameters and 3D points from the assembled equations. The approach used in this case is, *i.e.*, f7 [10] and the estimated transformation is a fundamental matrix $\mathbf{F}$.

The reconstruction itself begins with a selection of an initial pair. There are several heuristics to find a suitable initial pair [32]. The correct initial pair selection is crucial, as the reconstruction is based on extending the initial pair with other views. Therefore, we want the initial pair to be densely matched with other images to secure robustness and reconstruction accuracy. The wrong initial pair could cause the reconstruction to break down even though the reconstruction course would be smooth with a different pair.

With selected initial pair starts the incremental part of the reconstruction algorithm. Every iteration begins with the registration of a new camera $\mathbf{I_j}$. The condition for registering another image is that it has to observe some points of the already created 3D model. In the registration process is the estimated position of camera $\mathbf{P_j}$ using a RANSAC-based algorithm and an absolute pose solver. In case of COLMAP is used approach of P3P in combination with PnP [56] [10]. In the case of uncalibrated camera is estimated position together with intrinsic parameters using, *e.g.*, minimal solvers [57]. To find the absolute pose of the registered camera, we have to determine the central coordinate system, which is often defined as the coordinate system of the first camera. The first camera position $\mathbf{C_1}$ is, therefore, $\mathbf{0}$ and the rotation matrix is the unit matrix. Then, if a new image is registered, we have obtained its pose, and at least some features in the image observe the already existing 3D points. If any of the remaining features are matched with features in other images, with the use of triangulation is added 3D point corresponding to this match to the overall reconstruction - the track length of the feature needs to be at least 2. Methods for multi-view triangulation are, *e.g.*, [10] [58]. As mentioned, to extend the number of already reconstructed 3D points is used triangulation, *i.e.*, the process of creating 3D point $\mathbf{X_i}$ using more cameras observing the same 3D point to estimate its position. In triangulation, it is, in short, searched for the intersection of lines going through camera centers and mutually corresponding features. However, as we work in the real world, with floating numbers with limited precision and sensors with errors, the lines never come to the precise intersections, and it is searched for the point closest to all lines.

Every iteration contains errors caused by imprecise point localization, camera pose estimation, noisy matches, and imprecise calibration. Errors from each iteration accumulate with the increasing number of registered images - this effect is known as *drift*. The drift is reduced with a nonlinear optimization method called bundle adjustment (BA). BA aims at jointly refining a set of camera poses and 3D structure point estimates by minimization of reprojection error. Let's assume that $\hat{\mathbf{x}}_{\mathbf{ij}}$ is the projection of 3D point X_i to image plane of camera P_j projected using Eq. 2.16. The reprojection error is then a distance of $\hat{\mathbf{x}}_{\mathbf{ij}}$ and original feature position in the image $\mathbf{x_{ij}}$. Therefore, the error is generally defined as

$$E = \sum_j \|\hat{\mathbf{x}}_{\mathbf{ij}} - \mathbf{x_{ij}}\|_2^2, \tag{2.17}$$

where $\|\cdot\|$ is L2-norm. The reprojection error in COLMAP is then a variation of previous equation defined as

$$E = \sum_j \rho_j \left(\|\pi(\mathbf{X_i}, \mathbf{P_j}) - \mathbf{x_{ij}}\|_2^2 \right), \tag{2.18}$$

where ρ_j is weight function for possible outlier down weighting and π is projection function to image plane.

Note, that the reprojection error is invariant to scaling and rotation. In COLMAP is BA also used to find radial distortion parameters θ. This is then used to undistort the features in input images, which can be and are then used in P3P algorithm. As the bundle adjustment function is nonlinear, the algorithms to solve this issue are nonlinear least-squares methods. For solving problems of this type is often used library Ceres [59], which is also used in COLMAP.

Chapter 3

Proposed enhancements

In this chapter are discussed the drawbacks of SparseNCNet network and proposed the solution to this issues in § 3.1.1. In § 3.1.2 is then discussed re-training of the network.

3.1 SparseNCNet

3.1.1 Preecision and scores

As described in § 2.1.2 the SparseNCNet network takes as input two images and performs feature extraction and matching in a single feed-forward run. This results in a new set of keypoints determined with a sub-pixel precision for each image I_i with every image matching. This is a suitable property if we aim at the feature matching task itself, for it returns the precise matches. However, in the case of the feature matches further used in, *i.e.*, SfM, this brings difficulties. We name two reasons for this. The first reason is that in order to obtain a reconstruction, we need the track length higher than 2 for some keypoints, which is in this case often not achieved. Moreover, the higher the track length, the higher accuracy in 3D point positions obtained during the triangulation we gain. The second reason is that the results of such an approach are not compatible with classical SfM approaches, which assign arrays of keypoints to each image and then add matches, created as lists pointing to these keypoints arrays and into the database. What is more, the matches from SparseNCNet are sorted by the score, and the further used matches are often the top-scored k_m matches. This approach could possibly dismiss the slightly worse-scoring matches, thank the top k_m filter, nonetheless, these matches could have higher track length. In figure Figure 3.1 are pictured some of the matches generated by SparseNCNet and colored by

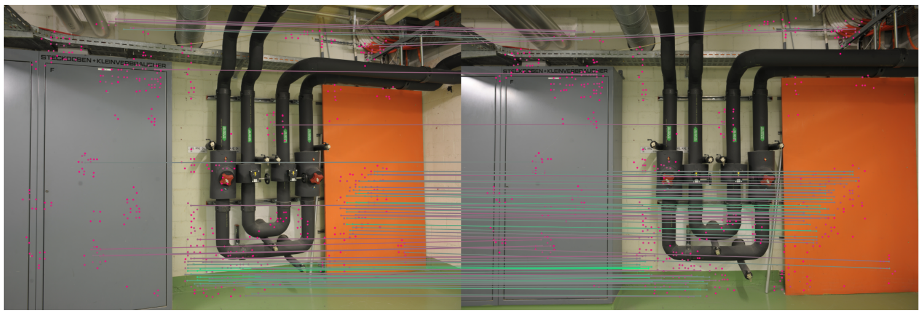


Figure 3.1: Visualisation of the top 100 matches between two images generated by SparseNCNet. Matches are colored by their score, with green standing for the highest score and red for the lowest.

assigned score. Note that the score for each match is computed from the sparse correlation defined in § 2.1.2 tensor using the $softargmax$ function.

To address all of the mentioned issues, we created a hashtable $H_j \in \mathbb{N}_0^{\lfloor \eta H \times \eta W \rceil}$ of keypoints for each image I_j. $H \times W$ stands for the resolution of input image I_j and $\eta \in \mathbb{R}^+$ is a precision factor. To each H_j corresponds a list of keypoints $\mathcal{K}_{H_j} = \{ (\mathbf{x_{kj}}, r_{kj}, s_{kj}) \mid \mathbf{x_{kj}} \in \mathbb{R}^2, r_{kj} \in \mathbb{N}_0, s_{kj} \in \mathbb{R}^+, k = 1...N_K \}$, where r_{kj} is number of match-references to the k-th keypoint in i-th hashtable, s_{kj} sum of scores of these matches, and N_K is a number of keypoints in the list. The list is ordered by the index k. This index is assigned to inserted keypoint in the time of adding it to the list and is computed from the corresponding hashtable. The adding process is explained below.

These hash tables are initialized with zeros on all positions. Then are the values of H_j updated with every matching of corresponding image I_j. After feed-forward run of image pair I_A, I_B we obtain N_{AB} keypoints $\mathbf{x_{iA}}$ corresponding to I_A and N_{AB} keypoints $\mathbf{x_{iB}}$ for I_B, where the i-th keypoints create match. SparseNCNet also generated corresponding match-scores $s_i^{(AB)}$. Let's demonstrate further process only for $\mathbf{x_{iA}}$. However, it is performed for both keypoint sets. At first is computed

$$\mathbf{x_{iH_A}} = \lfloor \mathbf{x_{iA}} \eta + 0.5 \rfloor, \tag{3.1}$$

what are coordinates in hashtable H_A. Then is taken

$$idx = H_A(\mathbf{x_{iH_A}}), \tag{3.2}$$

based on which we find the index of the keypoint as

$$k = \begin{cases} idx \text{ if } idx > 0 \\ max(H_A) + 1 \text{ otherwise} \end{cases} . \tag{3.3}$$

If the k is assigned as in the second case, the point is added to the $\mathcal{K}_{H_A}$ as $\mathbf{x_{kA}}$ with unit r_{kA} and with score $s_{kA} = s_i^{(AB)}$. In this case we also assign value $k = max(H_A) + 1$ to $H_A(\mathbf{x_{iH_A}})$, *i.e.*,

$$H_A(\mathbf{x_{iH_A}}) <= k = max(H_A) + 1. \tag{3.4}$$

If the $k = idx$, then for the k-th member of $\mathcal{K}_{H_A}$, $\mathbf{x_{kA}}$, is s_{kA} changed to $s_{kA} + s_i^{(AB)}$ and r_{kA} incremeted by one. The index k is then in both cases from Eq. 3.3 used in the matches list to point to the keypoint in the set $\mathcal{K}_{H_A}$ corresponding to image I_A.

This approach brings several advantages. Use of a hashtable is faster than would be a similar approach of storing all previously detected keypoints in an array and then searching this array, whether the keypoint is already registered. Another positive change is that we monitor the track length of each keypoint, and we also note the score, which has been the point evaluated throughout all the matchings. The last enhancement, which the hash-table approach brings, is the dynamic possibility of joint tuning of keypoint accuracy and the chance of higher track lengths. With smaller values of μ decreases the resolution of hashtables and with it also decreases the maximal accuracy of keypoints. However, as this approach merges more possible keypoint positions into one hashtable cell, it increases the chance of higher track length and thus the chance of successful 3D reconstruction.

3.1.2 Re-training

In order to use SparseNCNet in larger-scale 3D reconstruction applications in industrial environments, we need to train it on the datasets suitable for these purposes. In the [7] was this network trained mostly for day-night correspondences as shown in image Figure 3.2 and a significant part of the training data was from the exterior. Even though the illumination-change robustness is, in our case, also very usable, the main focus is still on weakly-textured-areas correspondence search. However, these data were not used in SparseNNCenet training epochs.

The SparseNCNet training dataset contains overlapping image pairs, which are divided into sets, based on the environment of capturing. The loss function is always computed for the whole batch. The batch contains N_b image pairs I_A, I_B and the ids of the training sets they belong to. After the matching each image pair I_{A_i} and I_{B_i} from the batch is computed the score as

$$s_{pos} = \frac{1}{N_b} \sum_{i=1}^{N_b} \frac{1}{2} (s^{(A_j B_j)} + s^{(B_j A_j)}), \tag{3.5}$$

where $s^{(A_i B_i)}$ is score of I_{A_i} and I_{B_i} matching and $s^{(B_i A_i)}$ score of matching with reversed order.

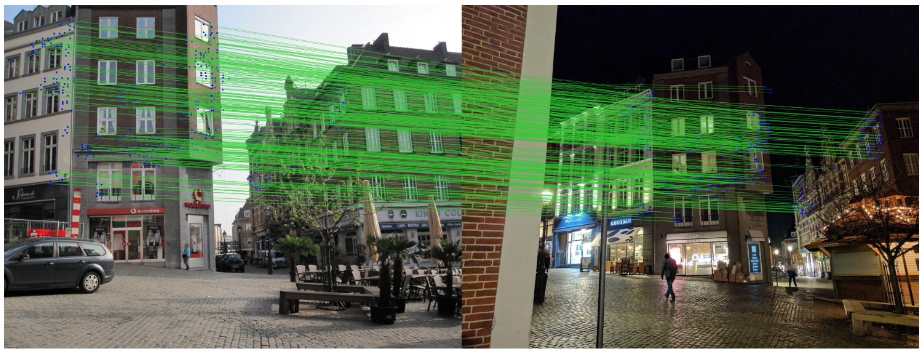


Figure 3.2: Day-night matches produced by SparseNCNet. Images are taken from Aachen Day-Night dataset.

Then are the second images I_{B_i} in all image pairs rolled around in the batch by one, creating image pairs which are not overlapping. Then is repeated the same process resulting this time in s_{neg}. Loss function is then defined as

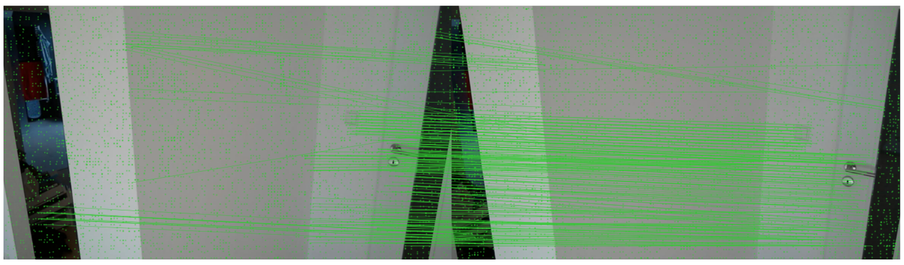
$$loss = s_{neg} - s_{pos}. \tag{3.6}$$

The loss indicates, that by minimizing it, we aim at as much high-socring positive matches as possible and at as least low-scoring negative matches as possible.

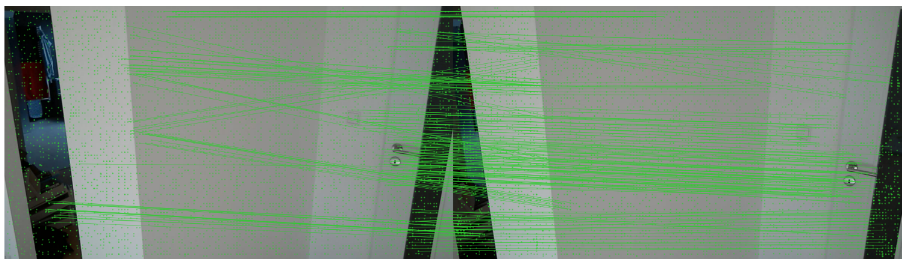
As mentioned in § 2.1.2, the SparseNCNet utilizes feature encoder Resnet101 as feature encoder. One of the possible changes in re-training the SparseNCNet would be to finetune the utilized network layers. Another re-trainable part of the network is the sparse, neighborhood consensus filtering providing network $N(\cdot)$, also introduced in § 2.1.2.

In our re-training attempt, we finetuned the last layer of the Resnet101 with whole $N(\cdot)$. Another parameters used for the re-training were batch size $b = 8$, learning rate $lr = 5 \cdot 10^{-4}$ and as optimizer we used Adam Optimizer [60]. Number of training image pairs was $N_t = 1180$ and testing image pairs $N_v = 415$. However, results did not bring any significant improvements, as can be seen in the Figure 3.3, because the re-training was conducted on mostly the same data as it was in the paper. However, there could be spotted slightly more emphasis on the wall correspondences in the re-trained case. Nonetheless, as soon as we collect enough training data of challenging interior spaces, the network is ready to continue re-training.

As we can see, the training is based only on the positiveness or negativeness of image overlap. Thus, another enhancement besides the training dataset could also be to improve the loss function to take into account the match distribution or directly the match positions. However, this is out of the scope of this work.



(a) : Matches produced by SparseNCNet with original weights.



(b) : Matches produced by re-trained SparseNCNet.

Figure 3.3: Comparison of top-scoring 200 matches generated by SparseNCNet with and without re-training.

Chapter 4

Experiments

This chapter begins with section § 4.1, which covers the reasoning behind ground-truth (GT) acquisition and describes the process of data collecting. This is followed by the description of the bench-marking system used to compare correspondence search approaches in § 4.2. Then is each of the approaches gradually searched for the best-performing parameter combination to be used on the challenging datasets. The first examined method is SIFT extractor in § 4.3, followed by relative pose estimators in § 4.4, then by preprocessing in § 4.5. The preprocessing experiments build on the results of both previous sections. In the following sections are examined correspondence search results of SparseNCNet in § 4.6 and SuperPoint + SuperGlue in § 4.7 both for various parameter combinations.

4.1 GT acquisition

There are several reasons why ground-truth data is needed. The primary goal of this work is to select the best-performing approach to correspondence search in the 3D reconstruction pipeline. Without a 3D ground truth, it is challenging to determine the accuracy of reconstruction. However, the device we use does not produce exactly a high-quality 3D dense point cloud, and it is out of the scope of this work to align all the points into a high-quality 3D dense point cloud. Therefore, to evaluate the reconstruction, we compare correspondence search results alone. The number and distribution of matches directly affect the uncertainty in camera poses and 3D point positions, as was shown in experiments by Polic *et al.* [18]. To compare various methods of feature operations, we needed to create an objective benchmarking system.

Results of each approach are verified with the ground-truth data matches and then compared with other methods. The ground-truth acquisition is

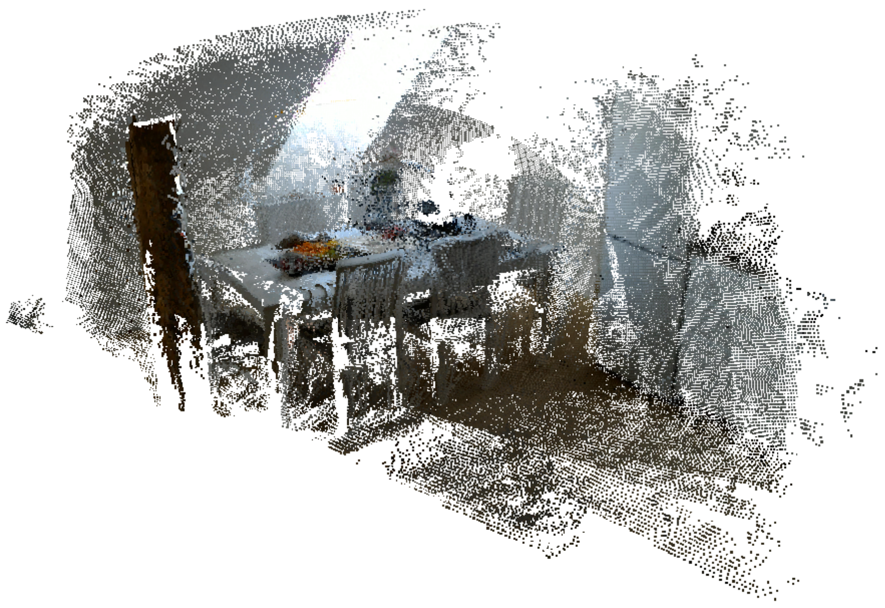


Figure 4.1: Visualisation of colored point cloud produced by Hololens.

also motivated by the use of trainable NNs. With suitable training data, the neural networks can be further trained in order to obtain better results on the challenging datasets mentioned in the chapters before, namely the ones containing plain walls and significant illumination changes. Specifically, the SparseNCNet re-training loop is ready to be used after collecting a sufficient amount of the training data to improve the results of 3D reconstruction on challenging datasets.

A device used to gather ground-truth data is Microsoft Hololens first-generation. Hololens is equipped with six cameras, of which four are tracking cameras, and one is the primary photometric camera, and the sixth mounted camera is a time of flight (TOF) depth camera. Therefore, the device produces both RGB images and depth maps. We can also receive corresponding time-stamp and position for each of produced images and depth maps. Note that images from the primary camera are taken with a different frequency (which is $\approx$ 15fps) than the depth maps (with $\approx$ 3fps). These two outputs we then process to obtain a colored 3D point cloud, an example of which can be seen in Figure 4.1. Another information provided by the device is calibration matrix K. Our goal is to align all the depth images into one common coordinate system ρ. In order to obtain 3D point in absolute coordinate system ρ, we apply

$$\lambda \begin{bmatrix} \mathbf{X_i^\rho} \\ 1 \end{bmatrix} = \mathbf{P_j}^{-1} \begin{bmatrix} -\mathbf{X_i^\delta}/1000 \\ 1 \end{bmatrix}, \tag{4.1}$$

where also

$$\mathbf{P_j}^{-1} = FrameToOrigin^T \, (CameraViewTransform^T)^{-1} \tag{4.2}$$

and where X_i^δ is a 3D point in the depth camera coordinate system δ, and the constant $1/-1000$ is used in order to obtain results in meters. As the Eq. 4.1 is defined as up-to-scale, λ secures the fourth coordinate to equal one. Our goal is a colored point cloud. Thus we have to assign to points a color. For this, we project the 3D points into an image coordinate system μ. Image to project the 3D point into is chosen as the one with the closest timestamp to the timestamp of the original depthmap of the point. The points are projected as

$$\lambda' \begin{bmatrix} \mathbf{x_i^\mu} \\ 1 \end{bmatrix} = K \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{P_j} \begin{bmatrix} \mathbf{X_i^\rho} \\ 1 \end{bmatrix}. \tag{4.3}$$

The color of the resulting pixel x_i^μ in the image is then assigned to 3D point X_i^p. The matrices $FrameToOrigin$ and $CameraViewTransform$ are provided by Hololens API. The relations above are derived to follow standard projective geometry equations defined in [32]. However, the Hololens API does not explicitly provide such a matrices, therefore we had to experimentally deduce the matrices, using the data provided by API.

Accuracy of Hololens camera positions and depth points has been examined in [61]. The results are value 1.6±0.1cm drift in camera position and 2.2±0.3° in camera angle, both per one second of tracking. These errors are then carried into the images, too. Here, the absolute pixel error projected into the images depends on the distance of the 3D point from the image plane. Thus, we could compute the maximal error of projection coming from the point 0.8m right in front of the camera sensor, which is the closest point observed in GT dataset. Another inaccuracy comes from the depth sensor, which is, on the other hand, most inaccurate with distant pixels. The noise carried by points as far from the sensor as 3.45m is 2cm, for the closest ranges at 0.8m it is 0.18cm. 3.45m is also the longest distance, which the depth sensor can measure. Using the distance restrictions stated before, mean maximal projection error between two neighboring ground-truth images brought by 3D point reconstruction inaccuracy is $e_m = 2.9$px. This was computed as

$$\lambda'' \begin{bmatrix} \mathbf{x_{ij}^\mu} \\ 1 \end{bmatrix} = K' \, h2a \left(\begin{bmatrix} \mathbf{R}'(\theta) & \mathbf{t}' \\ \mathbf{0} & 1 \end{bmatrix} \left(\begin{bmatrix} 0 \\ 0 \\ l \\ 1 \end{bmatrix} + \begin{bmatrix} \mathbf{s} \\ 1 \end{bmatrix} \right) \right), \tag{4.4}$$

with $l \in \{0.8, 3.45\}$ denoting furthest and closest points to depth sensor, $\theta \in \mathbb{R}^3$ being a vector of the angle error, $\mathbf{t}' \in \mathbb{R}^3$ error in camera position, and $\mathbf{s} \in \mathbb{R}^3$ the error in distance measured by the depth sensor. K' is then used calibration matrix, in our case diagonal matrix, having focal

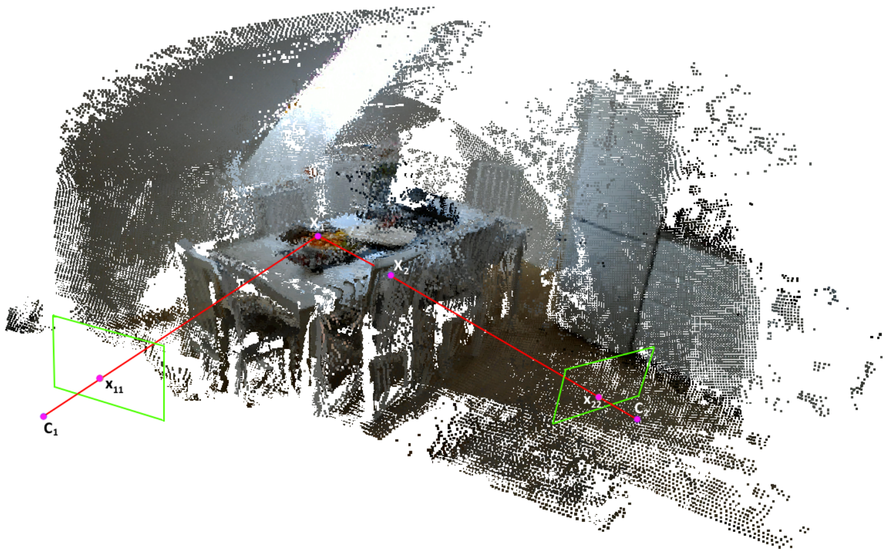


Figure 4.2: Error-causing camera positions and 3D point cloud combination.

lengths f_1, f_2 and 1 on the diagonal. $h2a$ is function transforming from homogeneous coordinates to ordinary 3D coordinates. The mean maximal error was obtained experimentally, where we gradually used elements from the error normal distributions, parameters of which was obtained from [61] and considered the mean value as the mean maximal error e_m. Approx. twice the value of the e_m is then used as parameter $\epsilon_{GT} = 6$, defined in , to minimize the number of incorrectly dismissed matches, because of GT errors.

Ground truth matches are obtained using projection of point cloud as defined in Eq. 4.3. If the same 3D point is projected into different images, the corresponding 2D projections are considered matches. As the 3D point cloud created by Hololens is dense, the projected keypoints are dense, too. After projection, ground truth 2D points are covering almost every pixel. Therefore wherever a match is found, we can find a near GT point and confirm or reject this match.

One potential flaw of this approach comes with more 3D points lying on the same projection line, as pictured in Figure 4.2. This could cause completely wrong matches to appear among ground truth. This case also can be seen in Figure 4.2, where the $\mathbf{X_1}$ lies on the structure which is occluded by structure with point $\mathbf{X_2}$ from the perspective of $\mathbf{C_2}$. Therefore, incorrect correspondence of $\mathbf{x_{11}}$ and $\mathbf{x_{22}}$ is created.

This issue can be addressed in two ways. Exact solution would be to create mesh from the point cloud, using, *e.g.*, Meshroom [15] and then to use a software such as Pyrender [62] to resolve the occlusions and project only the visible faces from the selected viewpoint to the images. However, creation of mesh from such data is challenging and out of the scope of this work. Nonetheless, as we are aware of this potential glitch, the datasets are collected with emphasis on avoiding such situations and image pairs are manually inspected before being incorporated into benchmarking data.

4.2 Benchmarking

As the quality of final reconstruction is generally affected by several factors, the benchmark system (BS) evaluates more aspects. The benchmarking data are collected as mentioned in § 4.1. The overall strategy of the BS is to verify feature matching results of individual algorithms using ground truth matches from § 4.1.

The correct match is defined as a match, where both corresponding features are closer then parameter ϵ_{GT} to GT features which correspond to each other. *I.e.*, set of correct matches $\mathcal{S}_{AB}$ between images I_A and I_B

$$\mathcal{S}_{AB} = \{\, (\mathbf{x_{jA}}, \mathbf{x_{jB}}) \mid \|\mathbf{x_{kA}^{(GT)}} - \mathbf{x_{jA}}\| \leq \epsilon_{GT}\,,\ \|\mathbf{x_{kB}^{(GT)}} - \mathbf{x_{jB}}\| \leq \epsilon_{GT}\,\}, \quad (4.5)$$

where $(\mathbf{x_{kA}^{(GT)}}, \mathbf{x_{kB}^{(GT)}})$ is a ground truth match between images I_A and I_B. This criterion is visualized in Figure 4.3.

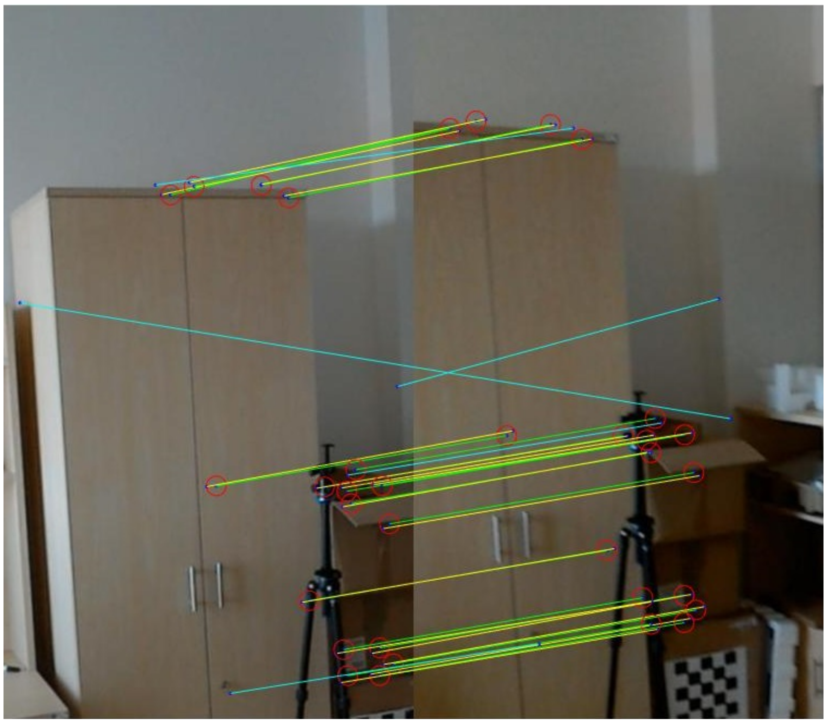


Figure 4.3: Visualisation of correct matches (yellow), threshold epsilon, here $\epsilon_{GT} = 9$ (red), ground truth matches (green) and incorrect matches (cyan).

The correct matches are then evaluated in three aspects, which are

- *absolute number of correct correspondences*,
- *relative number of correct correspondences*, which is computed as a ratio of number of correct matches to number of all matches returned by evaluated algorithm. This ratio is also called *Precision*, what is generally number of true positives to all positives (sum of true and false positives), and
- *correct match distribution*, which is computed by dividing the image into grid of size $N \times M$ and counting only k matches from each tile to the final image match count.

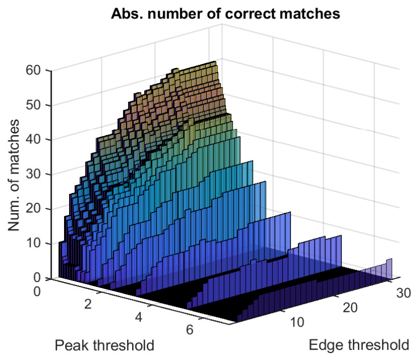
These aspects are connected purely to the evaluation of correspondence search. The 3D reconstruction results are then compared only using the best approaches from the previous stage. These are then evaluated using ETH3D benchmarking system [2], using publicly available datasets pipes and delivery area with ground truth. These datasets contain challenging factors as weakly textured areas.

4.3 SIFT parameter determination

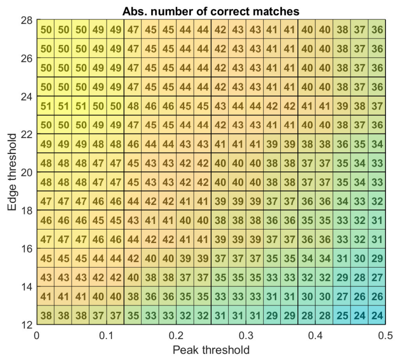
As mentioned in § 4.1, the overall quality of final 3D reconstruction is generally determined by the number and distribution of matches among input images. Matches and their distribution over the image, obtained using SIFT descriptor are directly affected by SIFT parameter values. The parameters were introduced in § 2.1.1. Therefore, we carried out an experiment to find the most suitable parameter combination for the challenging datasets.

The following experiments were performed as the baseline for further comparison. For feature detection and extraction was used SIFT detector + descriptor implemented in library VLFeat [4]. The matching was performed with the algorithm proposed by D. Lowe in [3] implemented in the same library. The purpose of the first experiment is to show the increasing sensitivity of the feature detector with the right direction of parameter changes. The tested parameters are peak threshold and edge threshold. The meaning of both parameters is explained in § 2.1.1. As shown in Figure 4.4a, the number of correct matches $n_S(p_T, e_T)$ increases for decreasing peak threshold p_T and increasing edge threshold eT. The Figure 4.4b focuses on the best-performing area.

However, with the increasing number of correct matches also comes a greater number of wrong matches. This can be best seen in Figure 4.5a,



(a) : Absolute number of correct matches dependency on SIFT parameters.

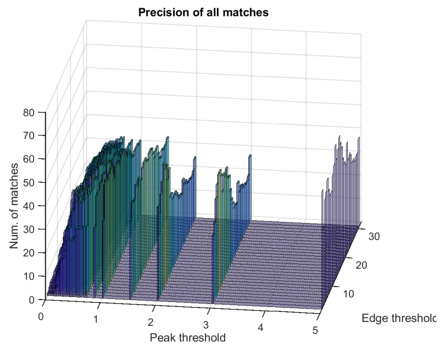


50	50	50	49	49	47	45	45	44	44	42	43	43	41	41	40	40	38	37	36
50	50	50	49	49	47	45	45	44	44	42	43	43	41	41	40	40	38	37	36
50	50	50	49	49	47	45	45	44	44	42	43	43	41	41	40	40	38	37	36
50	50	50	49	49	47	45	45	44	44	42	43	43	41	41	40	40	38	37	36
51	51	51	50	50	48	46	46	45	45	43	44	44	42	42	41	41	39	38	37
50	50	50	49	49	47	45	45	44	44	42	43	43	41	41	40	40	38	37	36
49	49	49	48	48	46	44	44	43	43	41	41	41	39	39	38	38	36	35	34
48	48	48	47	47	45	43	43	42	42	40	40	40	38	38	37	37	35	34	33
48	48	48	47	47	45	43	43	42	42	40	40	40	38	38	37	37	35	34	33
47	47	47	46	46	44	42	42	41	41	39	39	39	37	37	36	36	34	33	32
46	46	46	45	45	43	41	41	40	40	38	38	38	36	36	35	35	33	32	31
47	47	47	46	46	44	42	42	41	41	39	39	39	37	37	36	36	33	32	31
45	45	45	44	44	42	40	40	39	39	37	37	37	35	35	34	34	31	30	29
43	43	43	42	42	40	38	38	37	37	35	35	35	33	33	32	32	29	28	27
41	41	41	40	40	38	36	36	35	35	33	33	33	31	31	30	30	27	26	26
38	38	38	37	37	35	33	33	32	32	31	31	31	29	29	28	28	25	24	24

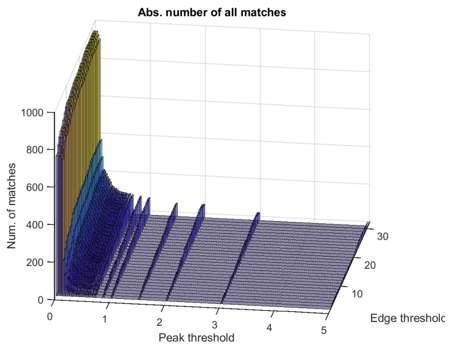
(b) : Absolute number of correct matches dependency on SIFT parameters, best-performing region focused.

Figure 4.4: Visualisation of number of correct matches dependency on SIFT parameters.

where is pictured a ratio of correct matches to all matches, *i.e.*, the precision $Pr_S(p_T, e_T)$. Figure 4.5b pictures an absolute number of all matches found by feature extractor and matcher, $n_{S_{all}}(p_T, e_T)$. By comparison of Figure 4.5b and Figure 4.4a, we can see the reason of low precision for high-sensitivity values. Also can be seen, that as the resulting numbers of matches of higher peak thresholds (lower sensitivity) are in order of units or lower tens, there suffice quite a low number of verified matches to gain high precision. See, *e.g.*, $(p_T, e_T) = (5, 10)$ in Figure 4.5a and Figure 4.5b. However, if the precision is 60% with 3 verified matches out of 5 found by the matcher, this is not sufficient to create the valid reconstruction. Note that these experiments are visualised only for one image pair to illustratively show the dependency of the number of matches on the parameters. The experiments using average over all datasets are then conducted in following section, § 4.4. However, the results are very similar for each image pair as the results visualised here.



(a) : Precision of correct matches dependency on SIFT parameters.



(b) : Absolute number of all matches dependency on SIFT parameters.

Figure 4.5: Visualisation of precision and abs number of all matches for SIFT parameter combinations.

4.4 Comparison of geometric verification methods

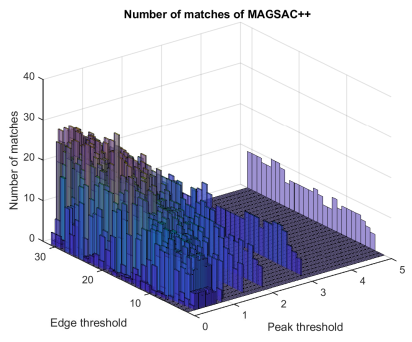
As shown before in § 4.3, the absolute number of matches increases with rising sensitivity. However, it is for the price of the simultaneously growing number of incorrect matches, often even with a higher rate, as shown in Figure 4.5a. And as the reconstruction algorithms can access no ground-truth data, the relative pose estimators' task to identify correct transformation becomes more challenging with higher numbers of incorrect matches and almost impossible with a high incorrect-correct matches ratio. In other words, if the precision of SIFT detector gets sufficiently low, the matches are so outlier-contaminated that the relative pose is almost impossible to compute. This is captured for one example image pair in Figure 4.6a, where can be seen a low number of correct matches $n_M(p_T, e_T)$ after geometric verification despite the high-sensitivity parameter settings and the high number of correct matches before geometric verification. The robust model estimator used in this figure is a state-of-the-art algorithm MAGSAC++ [36]. This algorithm does not explicitly state which relative pose solver does it utilize. Nonetheless, the minimal number of points to estimate the Fundamental matrix is 7. We also added another robust model estimator, LO-RANSAC [35], as a baseline, with its numbers of matches $n_L(p_T, e_T)$ shown in Figure 4.6c. Even though the f10e relative pose solver [34] is available for LO-RANSAC, we used the f7 [32] estimator, with the minimal number of points 7, to compare similar algorithms. The Figure 4.6b then shows the performance of MAGSAC++, focused on the best-performing region. The same then applies for Figure 4.6d and LO-RANSAC.

Another measure for geometric verification is precision. The precision, in short, shows a measure of the quality of verification output. The Figure 4.7a pictures precision of MAGSAC++ model estimator and the Figure 4.7b precision of LO-RANSAC. In the figures, we can see sizeable improvement in precision as opposed to high-sensitivity parameter combinations without any geometric verification, which can be seen in Figure 4.5a. Therefore we can see that we need geometric verification in order to use the SIFT matches in SfM pipeline. The improvement is the highest in the mid-sensitivity area, which we then use in the further application.

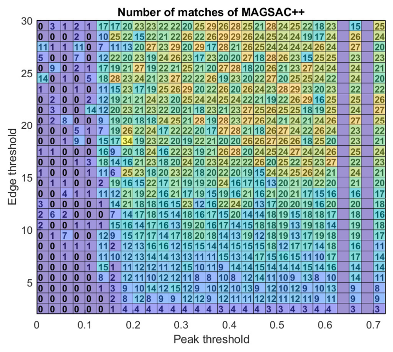
The average numbers over N_D GT image pairs for each bin on parameter coodrinates (p_T, e_T) are defined as

$$n_{avg}(p_T, e_T) = \frac{1}{N_D} \sum_{d=1}^{N_D} n_d(p_T, e_T), \tag{4.6}$$

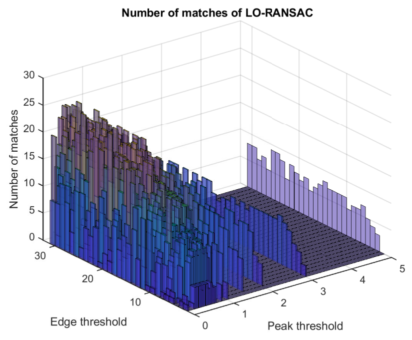
where $n_d(p_T, e_T))$ is number of correct matches for parameters $(p_T, e_T))$ in d-th image pair. The Figure 4.8a then shows extension of previous equation.



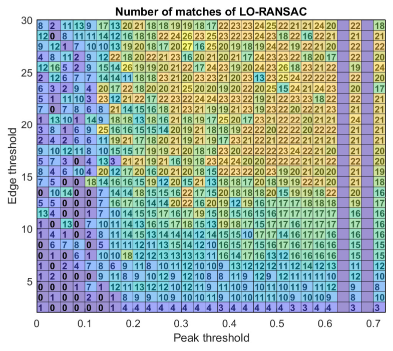
(a) : Numbers of correct SIFT feature matches verified by MAGSAC++ for different SIFT parameter combinations.



(b) : Numbers of correct SIFT feature matches verified by MAGSAC++ for different SIFT parameter combinations, focused at best-performing region.



(c) : Numbers of correct SIFT feature matches verified by LO-RANSAC for different SIFT parameter combinations.



(d) : Numbers of correct SIFT feature matches verified by LO-RANSAC for different SIFT parameter combinations, focused at best-performing region.

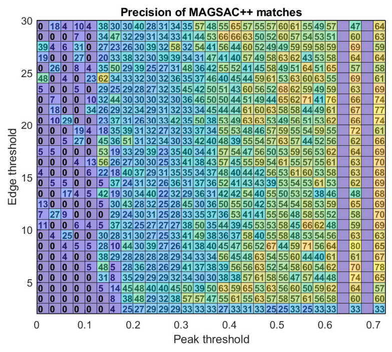
Figure 4.6: Comparison of number of correct SIFT features verified by MAGSAC++ and LO-RANSAC on the same image pair.

This states the minus sign if difference of numbers of matches

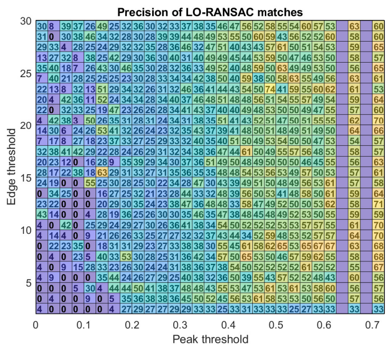
$$d_{GV}(p_T, e_T) = n_{M_{AVG}}(p_T, e_T) - n_{L_{AVG}}(p_T, e_T) \tag{4.7}$$

is negative, zero in case of equal performance and plus signs suggest better results of MAGSAC++. Note that the equality of performances is not defined exactly as $d_{GV}(p_T, e_T) = 0$, but we used threshold t_{GV}, therefore, the zero is written on coordinates (p_T, e_T) if $|d_{GV}(p_T, e_T)| < t_{GV}$. The Figure 4.8b then pictures difference of average precisions defined the same way as the difference of numbers of matches.

As can be seen in the comparison of two robust model estimators in Figure 4.8, MAGSAC++ achieves overall higher numbers of correct matches (captured in Figure 4.8a) with the same average precision as LO-RANSAC. The similar performance of these two estimators in terms of precision can be seen in Figure 4.8b. Therefore the MAGSAC++ is the estimator of choice

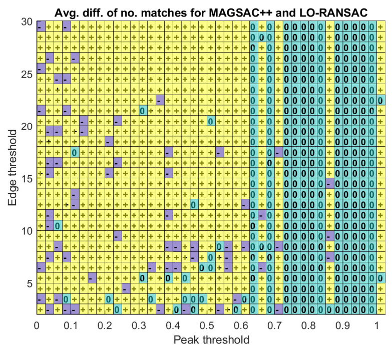


(a) : Precision of MAGSAC++ matches for different SIFT parameter combinations.

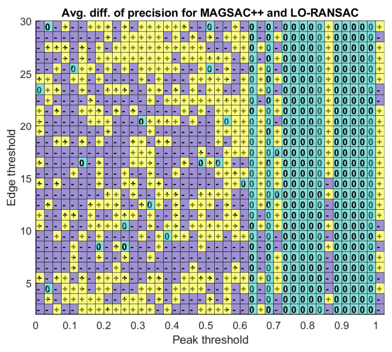


(b) : Precision of LO-RANSAC matches for different SIFT parameter combinations.

Figure 4.7: Comparison of precision of matches verified by MAGSAC++ and LO-RANSAC.



(a) : Sign of difference $d_{GV}(p_T, e_T)$ for average number of matches, defined in Eq. 4.7. Used threshold is $t_{GV} = 0.1$.

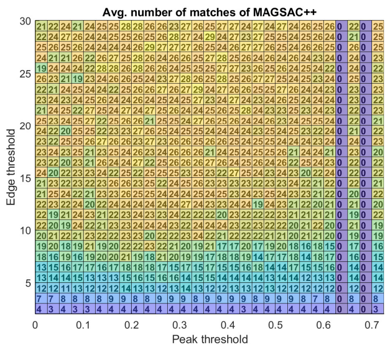


(b) : Sign of difference $d_{GV}(p_T, e_T)$ for average precisions, defined in Eq. 4.7. Used threshold is $t_{GV} = 0.1$.

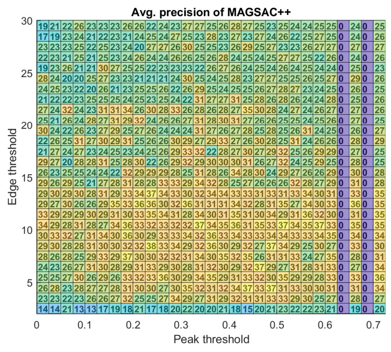
Figure 4.8: Comparison of MAGSAC++ and LO-RANSAC.

for further experiments.

As the testing of parameters of SIFT together with parameters of other methods, such as the methods used for preprocessing, would be highly time-expensive, we choose the most suitable parameters of SIFT for the subsequent experiments here. In the Figure 4.9 are shown best performing regions for MAGSAC++ average precision and number of matches. We can see that parameters $(p_T, e_T) = (0.3, 12)$ bring a reasonable balance between good precision and the number of matches.



(a) : Average number of correct matches over all image paris of MAGSAC++.



(b) : Average precision of correct matches over all image paris of MAGSAC++.

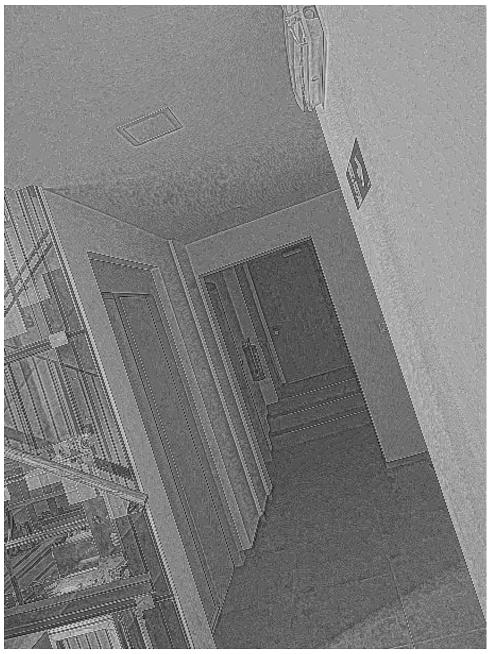
Figure 4.9: Average results of MAGSAC++.

4.5 Preprocessing results comparison

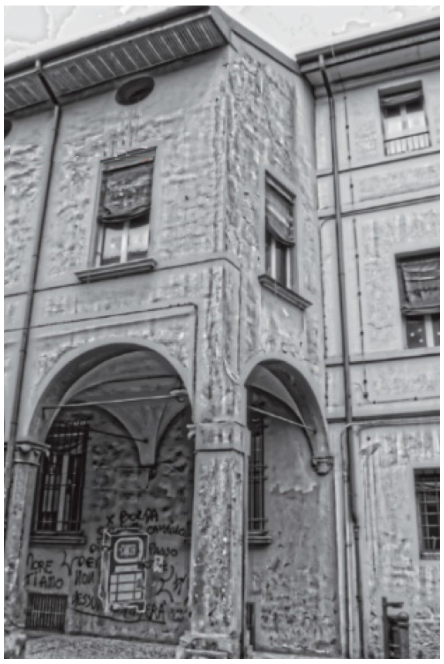
It was mentioned in § 2.1.3 that the application of preprocessing pipeline could significantly improve the results of 3D reconstruction. Our first goal in this experiment is to replicate the pipeline proposed in [8], in order to employ it in our comparison. The codes and dataset were not published in [8]. Therefore, we do to have datasets collected with Color checker and the color enhancement stage is skipped. Next follows the denoising stage, where is employed CBM3D implemented in [63]. The RGB to gray-scale conversion algorithm used is *MATLAB rgb2gray*, due to unavailable code or implementation details of their BID color converter. However, the BID is an extension of *MATLAB rgb2gray* anyway. As the last stage is employed content enrichment. Approach proposed in [8] utilizes a Wallis filter, which is tested in § 4.5.1. However, as the CLAHE filter offers a similar performance, we decided to add it to the comparison as the content enrichment stage as well. Results of which are captured in § 4.5.2.

4.5.1 Wallis filter results

The Wallis filter according to Gaiani *et al.* [8] is defined as Eq. 2.15. The proposed best performing parameter-combination is $A = 0.8$, B is linearly dependent on the mean pixel intensity of the input image, W is proportional to image size. In our case $W = 14$, $S = 127$ and $M = 60$. The result after application of accordingly set-up filter looks as it is captured in Figure 4.10a. The result is not quite similar to the result obtained in the paper, shown in Figure 4.10b.



(a) : Results obtained after application of Wallis filter by us with the same parameters as in Figure 4.10b.



(b) : Results obtained after application of Wallis filter shown in paper [8].

Figure 4.10: Comparison of our and paper Wallis filter application results.

Note, that Wallis filter is by Jazayeri *et al.* [64] defined as

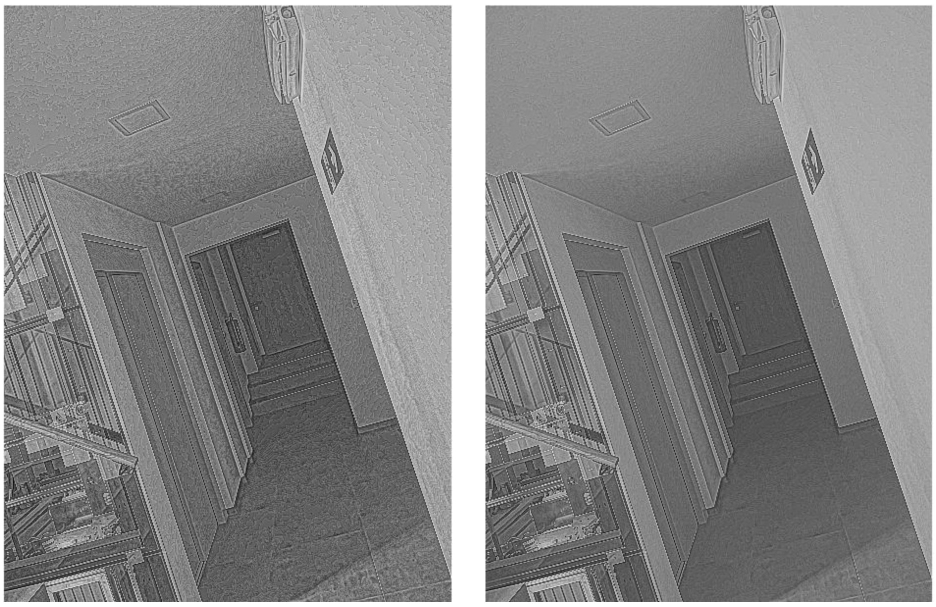
$$I_j^{(w)}(x,y) = \frac{SAI_j(x,y)}{sA + \frac{S}{A}} + BM + (1-B)m, \tag{4.8}$$

what is different formula than the one used by Gaiani *et al.* in [8]. Image after application of Wallis filter defined according to Eq. 4.8 is pictured in Figure 4.11a. Another published definition of Wallis filter is

$$I_j^{(w)}(x,y) = \frac{SI_j(x,y)}{s + \frac{S}{A}} + BM + (1-B). \tag{4.9}$$

The results of application result Eq. 4.9 of which can be seen in Figure 4.11b. As the original paper published by Wallis [55] is not accessible online, we can not compare the definitions to the original one and we stick to the version defined in Eq. 2.15 as their pipeline is to be replicated. Nonetheless, all of the implementations bring similar results, only with different parameter A.

Note that the experiments are performed using SIFT parameters obtaining the highest overall score in Figure 2.5 and § 4.4, which is peak threshold $p_T = 0.3$ and edge threshold $e_T = 12$ in combination with MAGSAC++. In Figure 4.12a are pictured results of comparison of number of matches found among images preprocessed wit Wallis filter in the pipeline. The results are pictured for various parameters A and B, *i.e.*, $n_W(A,B)$, with number of



(a) : Input image after application of Wallis filter defined as in Eq. 4.8.

(b) : Input image after application of Wallis filter defined as in Eq. 4.9.

Figure 4.11: Application of Wallis filter implemented as defined in other sources.

matches found with no preprocessing, *i.e.*, n_N. The figure states the minus sign if absolute difference of numbers of matches

$$d_W(A,B) = n_W(A,B) - n_N \tag{4.10}$$

is negative, zero in case of equal performance and plus signs suggest better results of preprocessed images. In some pictures is plus sign replaced by percentage of improvement over image with no preprocessing $p_W(A,B)$, computed as

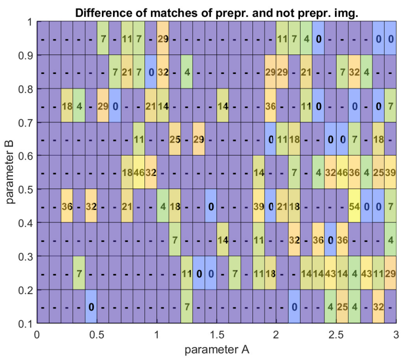
$$p_W(A,B) = \left(\frac{n_W(A,B)}{n_N} - 1\right) \cdot 100. \tag{4.11}$$

The Figure 4.12b, Figure 4.12c and the Figure 4.12d capture results of the comparison for another image pairs.

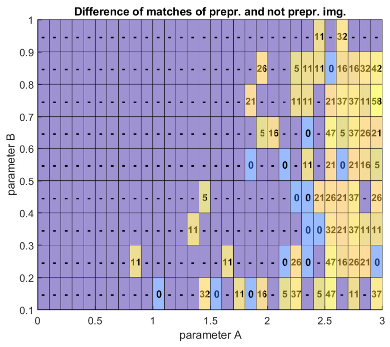
In this example can be seen that some parameter combinations of Wallis filter yield improvement, but are not necessarily the same for all iamge pairs.

On one hand, as shown in Figure 4.13, the resluts of preprocessing for some parameter combinations bring higher number of verified matches. But on the other hand the matches are not found on textureless areas. These were supposed to be found according to [8]. However, the additional matches are found mostly in shadowed regions. The Figure 4.13 visualises results for best-performing parameter combination on the image pair, on which the experimets were realized.

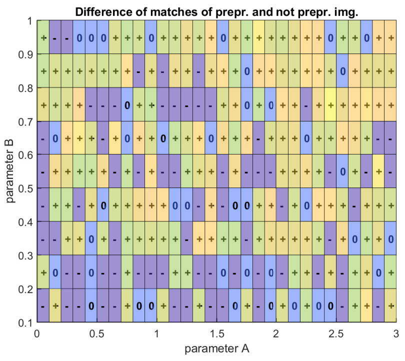
The Figure 4.14a utilizes the equation Eq. 4.6 in combination with Eq. 4.11



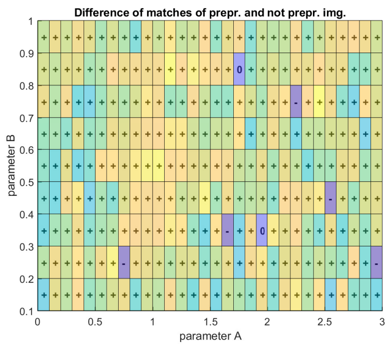
(a) : Difference of absolute matches between images with preprocessing and without preprocessing. Minus sign means higher scores of images with no preprocessing, positive number shows improvement in percents with use of preprocessing.



(b) : Difference of absolute matches between images with preprocessing and without preprocessing. Minus sign means higher scores of images with no preprocessing, positive number shows improvement in percents with use of preprocessing.



(c) : Difference of absolute matches between images with preprocessing and without preprocessing. Plus sign means higher scores of images with preprocessing.

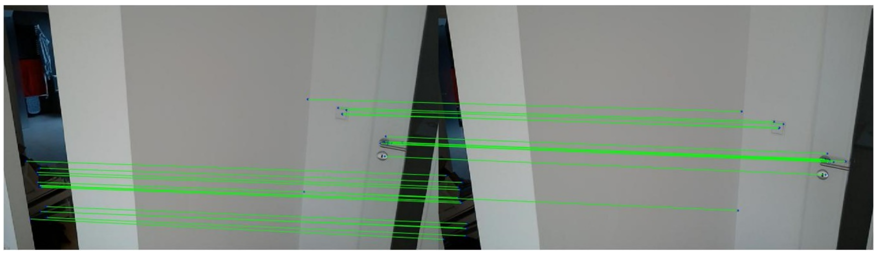


(d) : Sign of difference of absolute matches between images with preprocessing and without preprocessing. Plus sign, therefore means higher scores of images with preprocessing.

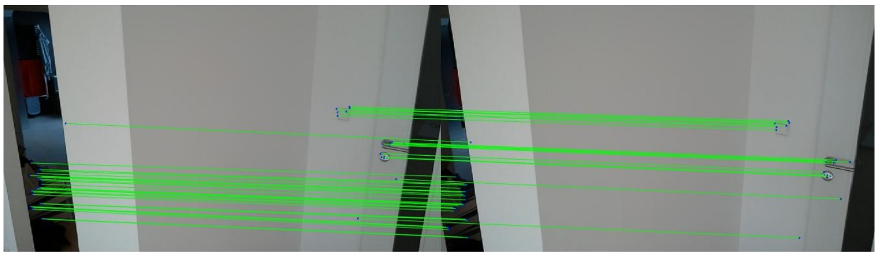
Figure 4.12: Comparison of results achieved with Wallis filter with results without preprocessing on four different image pairs for several paremeter combinations (p_A, p_B).

using coordinate space of parameters of Wallis filter, which is (A, B). The Figure 4.14a thus shows percentage of improvement over $n_{N_{AVG}}$, what is averaged number of vanilla matches throughout all N_D image pairs. The Figure 4.14b then shows rounded percentage of pairs, for which the parameter combination (A, B) really improved the results.

As can be said from Figure 4.14b, Figure 4.12 and Figure 4.14a, there is no universal combination of parameters, which improves the results for all of the situations. Each dataset is unique and fits best with different parameters. However, as the Wallis mostly improves the results, we could choose the

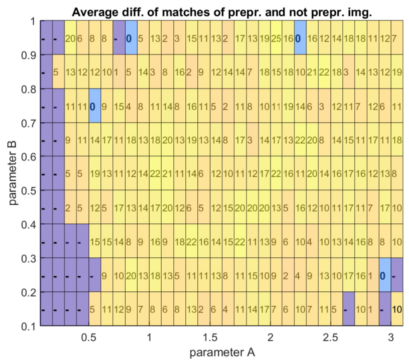


(a) : Correct geometrically verified matches found with no preprocessing.

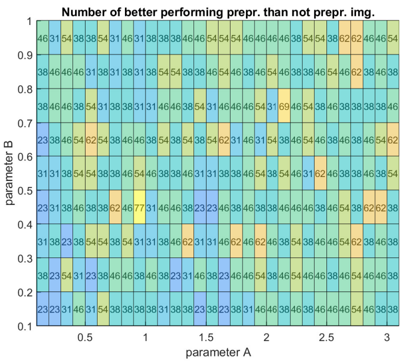


(b) : Correct geometrically verified matches found with application of Wallis filter.

Figure 4.13: Comparison of correct verified matches found with preprocessing and without.



(a) : Average percentage of improvement after preprocessing over average vanilla matching, improvement defined as in Eq. 4.11.



(b) : Percentage of image pairs, for which the parameter combination (p_A, p_B) really improved the results over the vanilla images.

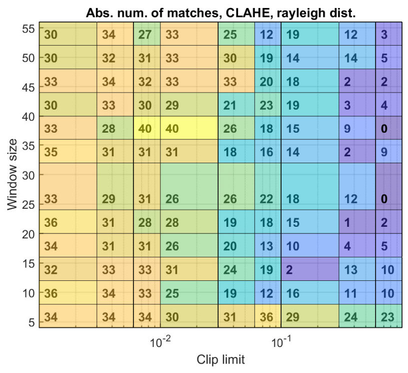
Figure 4.14: Average Wallis filter results over image pairs in all 5 datasets.

parameter combination yielding improvements of reasonable percentage in most cases. This combination is according to the figures $(A, B) = (2.1, 0.7)$.

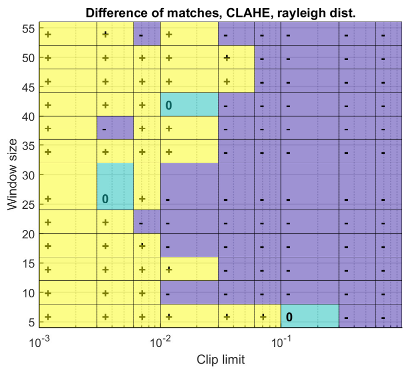
4.5.2 CLAHE results

As the implementation of the CLAHE filter, we used *MATLAB adaphisteq*. The experiments are again visualized in filter-parameter space, with parameters *clip limit* CL, where $CL \in [0, 1]$ and *tiles number* $TS \in \mathbb{N}^{N \times M}$.

The implementation supports multiple distributions besides the *uniform* distribution in histogram equalization process. This was explained in § 2.1.3. The other distributions are *Rayleigh* and *exponential*, both included in experiments. The Figure 4.19 pictures results of one example image pair obtained after application of CLAHE as content enrichment method in the preprocessing pipeline from § 2.1.3 for different (TS, CL) combinations. The Figure 4.16f pictures absolute number of correct matches $n_C(TS, CL)$ found in preprocessed image pair. In Figure 4.15b is then shown a sign of difference $d_C(TS, CL) = n_C(TS, CL) - n_N$, where n_N is, as defined before, number of matches on the image pair without preprocessing.



(a) : Absolute number of correct matches found in preprocessed image pair with parameters (TS, CL), using rayleigh distribution.

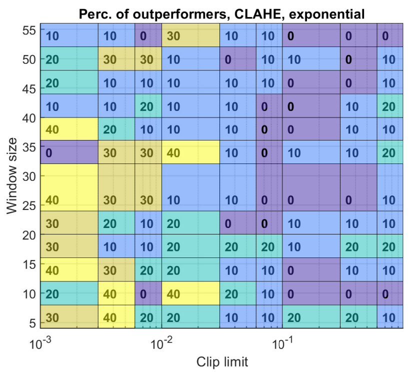


(b) : Sign of difference between number of matches of image pair with preprocessing (TS, CL) minus matches of vanilla images, using ray. dist.

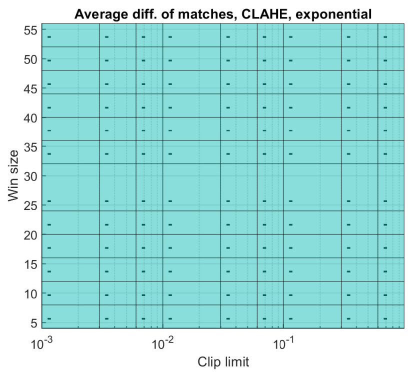
Figure 4.15: Results for different CLAHE parameters on one example image pair.

Average results among all ground-truth image-pairs are pictured in Figure 4.16. This is visualized for each distribution separately. Left column pictures percentages of preprocessed image pairs, which outperformed the original image pairs for CLAHE parameter combination (TS, CL). Right column in Figure 4.16 then shows the sign of difference in absolute number of matches $d_C(TS, CL)$.

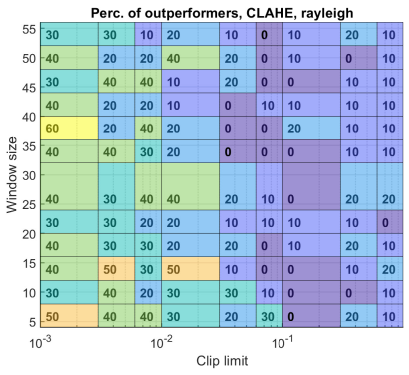
The best average performance appears for all distributions parameters in bottom and left part of the parameter space, where are the parameter combinations with the lowest preprocessing effect. $TS = 1$ and $CL = 0$ means no preprocessing at all. The trend in difference figures shows, that even though some parameter combinations in some datasets show improvements, the overall average performace does not improve with the application of CLAHE filter. Therefore, we choose the Wallis filter as content enrichement stage of preprocessing pipeline for further experiments.



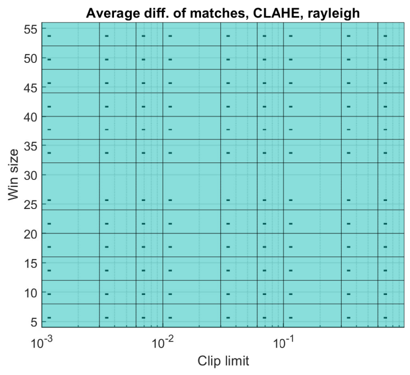
(a) : Percentage of image pairs preprocessed with parameters (TS, CL) performing better than vanilla image pair, parameter exp. dist.



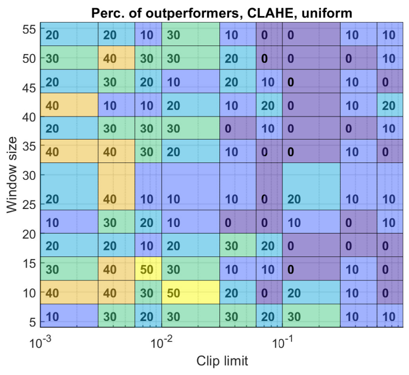
(b) : Sign of difference between average matches of image pairs with preprocessing (TS, CL) minus average matches of vanilla images, using exponential dist.



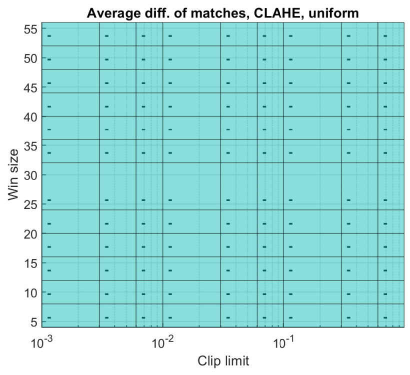
(c) : Percentage of image pairs preprocessed with parameters (TS, CL) performing better than vanilla image pair, parameter rayleigh distribution.



(d) : Sign of difference between average matches of image pairs with preprocessing (TS, CL) minus average matches of vanilla images, using rayleigh dist.



(e) : Percentage of image pairs preprocessed with parameters (TS, CL) performing better than vanilla image pair, parameter uniform distribution.



(f) : Sign of difference between average matches of image pairs with preprocessing (TS, CL) minus average matches of vanilla images, using uniform dist.

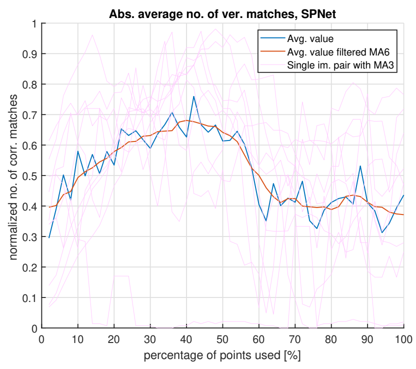
Figure 4.16: Comparison of average benchmark results for all distributions using CLAHE filter.

4.6 SparseNCNet

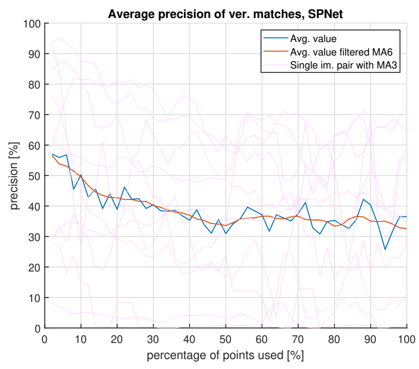
The only easily adjustable visibly-results-affecting parameter of the SparseNCNet from the user side without re-training, is the number of the used highest-scoring matches, k_m. Beware, that this is not the constant K defined in § 2.1.2. Therefore, k_m could be computed after the NN's feed-forward. This fact we use to compute the best percentage of score-sorted matches p_m to keep in order to obtain the best results. As the number of matches n_m produced by SparseNCNet may vary for different image pairs from tens to thousands, the absolute number would then be unusable for some image pairs. Thus the experiments aim at computing the p_m, and then compute k_m as

$$k_m = round(\frac{p_m}{100} \cdot n_m). \tag{4.12}$$

The average normalized number of matches is in Figure 4.17a and average precision in Figure 4.17b, both for various values of p_m. The figures show several lines, where blue is the value of the of precision or number of matches, red is the same value filtered with moving average of window size 6 (MA6) and the pink, low-opacity lines picture values for each image pair, where all of them are filtered with MA3. Note, that by normalization is meant dividing all values corresponding to the same image pair by maximum of them. All of the correct matches were found after geometric verification using MAGSAC++.



(a) : Average normalized number of correct matches dependency on percentage of kept matches parameter p_m.



(b) : Average precision of correct matches dependency on percentage of kept matches parameter p_m.

Figure 4.17: Results of SparseNCNet experiments dependency on percentage of used matches p_m.

The percentage $p_m = 36\%$ brings a suitable balance between the number of correct, verified matches and the precision. Therefore is this value used in further comparisons. In the Figure 4.22 are eventually show correct matches found with SparseNCNet with found parameters. The image pair is the same as in the Figure 4.13.

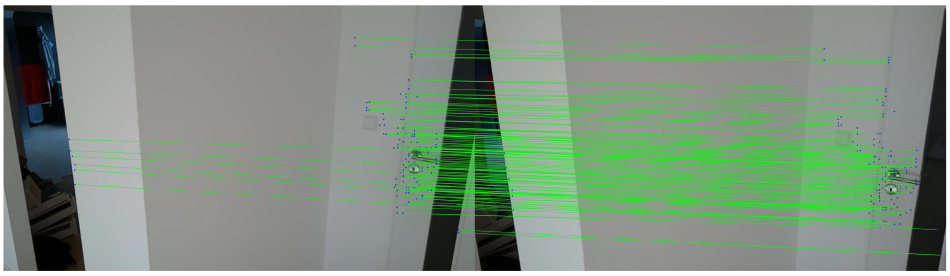


Figure 4.18: SparseNCNet results on the same image pair as in Figure 4.13.

4.7 SuperGlue + SuperPoint

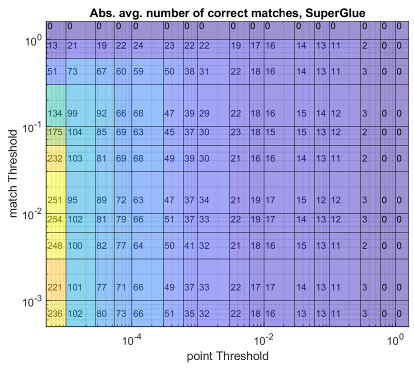
The last approach employed in comparison is the combination of learnable feature detector SuperPoint and learnable feature matcher SuperGlue. The first of the parameters examined in experiments is the parameter of SuperPoint, *point threshold* pT, which works similarly to a combination of *edge threshold* and *peak threshold* in the case of SIFT. This means that the value of *point threshold* directly affects the sensitivity of the feature detector and, therefore a number of detected keypoints. Another examined parameter is *match threshold* mT, which is defined in SuperGlue, and it has an effect on a number of matches, which are declared valid.

The results were tested in several aspects. The first criterion is captured in Figure 4.19a, which captures average number $n_{SG}(pT, mT)$ of correct matches returned by SuperGlue and verified by MAGSAC++ , computed as in Eq. 4.6, just for each parameter space (pT, mT). Next aspect is shown in Figure 4.19b. There can be seen average precision of the same matches $Pr_{SG}(pT, mT)$. Note that in both figures, the bottom-last row uses $mT = 0$ and the most left column employs $pT = 0$. It is visualized this way because of logarithmic scale.

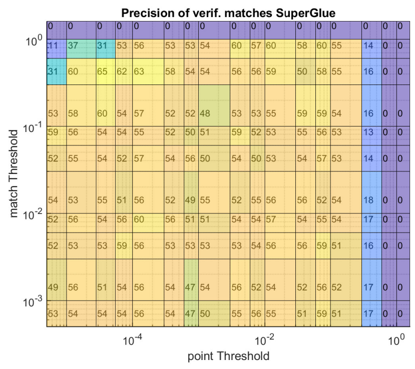
SuperGlue is also network producing supposedly verified matches. However, unlike SparceNCNet, the matches really strongly resemble the matches verified with some hand-crafted model estimator. As can be seen in Figure 4.20, the geometric verification process with MAGSAC++ removed also correct matches produced by SuperGlue and shown in Figure 4.20a. The verified version is then in Figure 4.20b.

Therefore, we decided also to show the performance of SuperGlue without any geometric verification. This can be seen in Figure 4.21, where Figure 4.21a shows absolute average number of matches and in Figure 4.21b is shown precision of this approach without MAGSAC++.

From Figure 4.21 and Figure 4.19 we can say that the precision is worse in the case of non-verified matches for most of the parameter space. However,

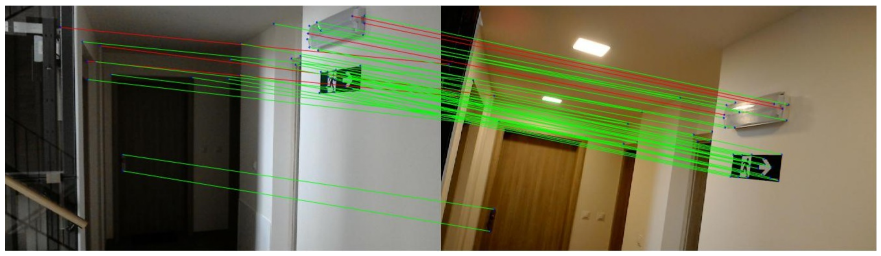


(a) : Average number of correct matches for each parameter combination (pT, mT).

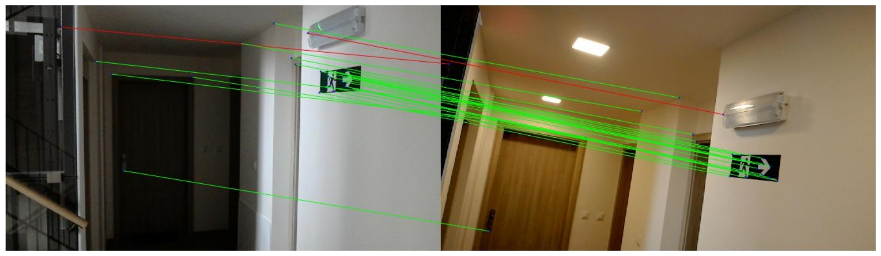


(b) : Average precision of matches returned by SuperGlue for each parameter combination (pT, mT).

Figure 4.19: Average results for different SuperGLue and SuperPoint parameters, using MAGSAC++.



(a) : Matches found using Superglue and no additional geometric verification.

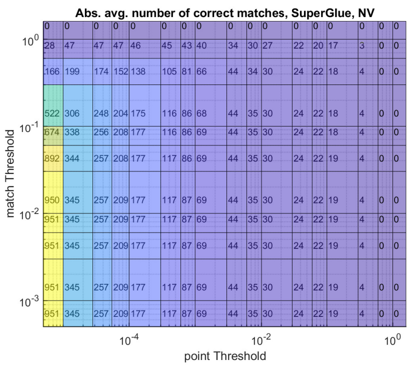


(b) : Matches found using Superglue and additional geometric verification MAGSAC++.

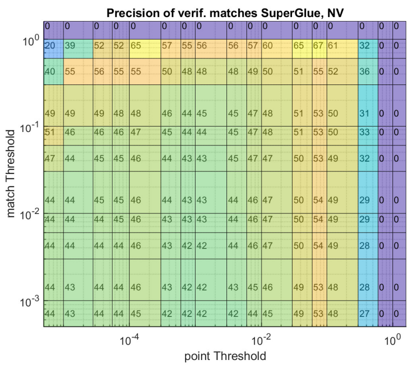
Figure 4.20: Comparison of matches found by SuperGlue with MAGSAC++ and without.

the absolute number of matches is on the other hand several times higher.

In the Figure 4.22 are captured results on the same image pair as in Figure 4.13 and and Figure 4.22 this time using SuperPoint + SuperGlue. The Figure 4.22a are shown correct matches obtained purely with SuperPoint + SuperGlue, the Figure 4.22b then shows correct matches after verification with MAGSAC++. We can see obvious improvement in number of matches on the weakly textured area in both of the cases using SuperPoint + SuperGlue as opposed to the results of previous methods. This is probably because of



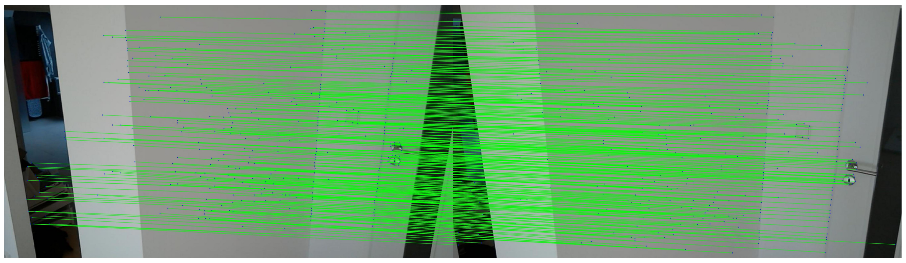
(a) : Aaverage number of correct matches for different parameter combinations (pT, mT), no MAGSAC++.



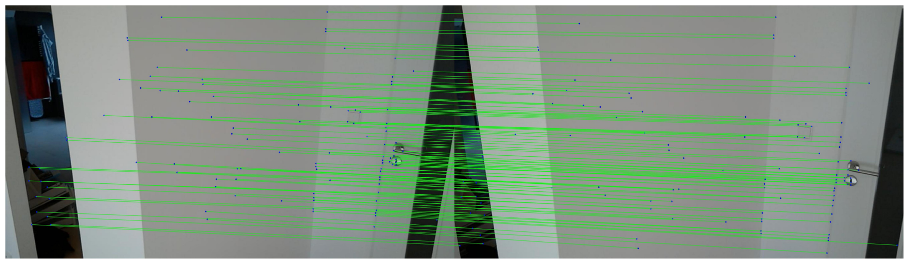
(b) : Average precision of matches returned by SuperGlue for parameter combinations (pT, mT), no MAGSAC++.

Figure 4.21: Average results for different SuperGlue and SuperPoint parameters, not using MAGSAC++.

precise context information incorporation in the SuperGlue network.



(a) : SuperGlue correct matches on the same image pair as previous methods without MAGSAC++.



(b) : SuperGlue correct matches on the same image pair as previous methods with MAGSAC++ verification.

Figure 4.22: SuperGlue correct matches shown on the same image pair as in Figure 4.13 and Figure 4.22 for comparison.

Chapter 5

Comparison of methods

In the previous chapter were examined various methods of correspondence search for the best performing parameters, which can be used in challenging datasets, such as factories. This chapter brings benchmark results, which compared the previously examined methods using the obtained parameters. The first part is a comparison of the same type as in the Chapter 4, *i.e.*, using GT collected with Microsoft Hololens to verify matches between 2D image pairs. This experiments are described in § 5.1. Another benchmark is performed using benchmarking system ETH3D [2] in § 5.2. This is a comparison of final 3D reconstruction after SfM and MVS with ground truth collected with laser-scan.

5.1 Correspondence search benchmark

As was mentioned several times before, the quality of the final reconstruction is determined by number and distribution of correct matches. Therefore, following benchmark compares all of the approaches in terms of average number of matches, average precision and average distribution, with all of the evaluated aspects being defined in Chapter 4. The parameters used for distribution evaluation, which were introduced in § 4.2 is a grid size $N \times M = 6 \times 8$ and counting only $k = 10$ matches per tile to final number. The evaluated methods are

- **SIFT + MAG**, what is SIFT extractor and descriptor used in combination with MAGSAC++ and with best parameters defined in § 4.4, as $(p_T, e_T) = (0.3, 12)$,
- **Wal+SIFT+MAG**, being preprocessing pipeline from [8], reconstructed

	Dataset no.								
	1			2			3		
	abs.	pre.	dis.	abs.	pre.	dis.	abs.	pre.	dis.
SIFT+MAG	12.0	27.1	11.5	6.5	21.2	6.5	32.0	52.7	22.3
Wal+SIFT+MAG	9.0	25.0	7.8	9.3	30.2	9.3	20.7	39.5	10.8
SupGlue	**61.5**	**42.2**	52.5	**144.3**	69.6	**112.9**	**107.0**	62.6	**92.3**
SupGlue+MAG	60.5	39.4	**56.0**	65.5	**72.5**	59.9	54.0	**73.5**	50.5
SpNCNet+MAG	55.0	13.7	24.8	91.3	41.1	59.8	71.3	45.5	46.8

Table 5.1: Average results for each aproach in three example datasets.

	Average		
	abs.	pre.	dis.
SIFT+MAG	18.3	35.2	15.4
Wallis+SIFT+MAG	18.6	33.3	14.5
SuperGlue	**126.8**	57.6	**100.0**
SuperGlue + MAG	71.6	**63.1**	64.9
SparseNCNet	116.8	36.9	71.9

Table 5.2: Average results of correspondence search over all datasets.

in § 4.5.1 with suitable parameters obtained in the same section. THe parameters are $(A, B) = (21, 0.7)$ in combination with MAGSAC++,

- **SupGlue**, standing for SuperPoint and SuperGlue combination without geometric verification and with parameters $(pT, mT) = (3 \cdot 10^{-5}, 0.3)$,
- **SupGlue + MAG**, what is Superpoit, SuperGlue and MAGSAC++ with the parameters $(pT, mT) = (6 \cdot 10^{-5}, 0.03)$ in combination with MAGSAC++,
- **SpNCNet+ MAG**, standing for SparseNCNet using parameter $p_m = 36\%$ in combination with MAGSAC++.

Average results for each aproach in five datasets are shown in Table 5.1. The overall average results for all tested methods are then captured in Table 5.2.

5.2 3D reconstruction benchmark

3D reconstruction benchmarking is performed using benchmarking system ETH3D [2]. This system is used because 3D models of our ground-truth data are not sufficiently accurate as opposed to GT from ETH3D, which is collected using laser scan. Another reason is that the implementation of such a 3D benchmarking system would be out of the scope of this work. Thus, we created an automated pipeline for dense 3D reconstruction, subsequent to the correspondence-search methods. This pipeline is utilizing COLMAP, namely, its SfM and MVS parts. The process begins with correspondence search, where are employed the same methods as in § 5.1, *i.e.*, SIFT + MAGSAC++, Wallis + SIFT + MAGSAC++, Superpoint + SuperGlue, Superpoint +SuperGlue + MAGSAC++, and SparseNCNet + MAGSAC++ plus SPNCNet, Wallis + SIFT, SIFT. The parameters used for the methods are the same as in § 5.1, with just a note to SparseNCNet where we have experimentally determined the best-performing precision parameter $\eta = 0.13$. This parameter was defined in § 3.1.1. And note to the SIFT is that the images feeded into SIFT were resized by scale 1/4. This is motivated by the smaller resolution of the images on which we searched for the best parameters.

Results of this part are image keypoints, matches and in the case of methods which including geometric verification, MAGSAC++, also two-view geometries with fundamental matrices F. The results are then added to the COLMAP database together with images and cameras with intrinsic parameters given in GT. In the case of unverified matches, then follows COLMAP stage *exhaustive matcher*, which performs geometric verification on the calculated tentative matches. Afterwards are gradually executed COLMAP stages. At first, the *mapper*, which results in sparse reconstruction, followed by *image undistorer*, *patch match stereo* and *stereo fusion*. The last three stages then provide the final dense pointcloud $\mathcal{X}^{(D)} = \{\, \mathbf{X_i^{(D)}} \mid \mathbf{X_i^{(D)}} \in \mathbb{R}^3, i = 1..N_{X_D} \,\}$. Note that in order obtain comparison results of COLMAP reconstruction and GT in the ETH3D benchmark, we need to align our reconstruction with the GT. This is accomplished with the use of the given GT camera positions. We find transformation mapping between corresponding camera centers obtained in SfM part $\mathbf{C_j}$ to GT camera centers $\mathbf{C_j^\sigma}$. It is defined as

$$\mathbf{C_j^\sigma} = s^\sigma \mathbf{R}^\sigma \mathbf{C_j} + \mathbf{t}^\sigma, \tag{5.1}$$

where s^σ denotes scale, $\mathbf{R}^\sigma$ rotation matrix and $\mathbf{t}^\sigma$ translation vector. This could also be written using a homogeneous matrix. Nevertheless, for simplicity, it is written separately. This transformation is then applied on each point of the obtained dense pointcloud, and we obtain aligned dense pointcloud, as

$$\mathbf{X_i^{(A)}} = s^\sigma \mathbf{R}^\sigma \mathbf{X}_i^{(D)} + \mathbf{t}^\sigma. \tag{5.2}$$

This is the pointcloud, which we feed into the ETH3D benchmarking system. Another parameter is array of tolerances in meters. The system then for each of given tolerances returns three evaluated metrics which are

- **accuracy**[%], the ratio saying how much of the reconstruction is closer to the ground truth than the tolerance,
- **completeness**[%], the ratio saying how large part of ground truth is closer to the reconstruction than the tolerance, and

ETH3D Accuracy results [%]								
Dataset no.	1				2			
tolerances[m]	0.02	0.05	0.10	0.20	0.02	0.05	0.10	0.20
SIFT	15.8	55.9	87.2	93.2	**83.5**	**95.5**	**98.0**	**98.8**
SIFT+MAG	37.0	65.8	84.5	90.5	0.2	2.5	51.2	85.4
Wal+SIFT	64.5	**89.8**	**95.7**	**97.9**	65.6	89.7	95.9	98.1
Wal+SIFT+MAG	7.8	17.8	33.6	59.5	58.4	86.3	95.4	98.1
SupGlue	**75.1**	89.2	92.9	95.8	8.8	19.5	32.4	47.8
SupGlue+Mag	65.3	82.6	88.1	93.8	37.7	59.2	69.7	83.7
SpNet	65.8	81.0	86.7	90.5	14.8	48.1	65.3	82.6
SpNet+MAG	0.0	0.0	0.0	0.0	4.4	11.6	21.8	31.5

Table 5.3: Accuracy evaluation of COLMAP pipeline with various correspondence search methods using ETH3D benchmarking system and two ETH3D datasets, electro and pipes.

- **F1 score**[%], being the harmonic mean of accuracy and completeness [2].

In all of the metrics is the higher number the better. Results for all methods in two ETH3D datasets pipes and delivery area are divided into three tables, each evaulating one of the metrics. Accuracy metrics are evaluated in Table 5.3, the Completeness ETH3D results are in Table 5.4 and the F1-scores for all correspondence search methods are captured in Table 5.5.

ETH3D Completeness results [%]								
Dataset no.	1				2			
tolerances[m]	0.02	0.05	0.10	0.20	0.02	0.05	0.10	0.20
SIFT	3.0	13.1	23.9	**35.3**	**52.1**	**73.9**	**84.1**	**91.8**
SIFT+MAG	2.5	7.5	12.8	16.1	0.7	41.3	68.7	82.4
Wal+SIFT	**7.0**	**15.6**	**23.8**	32.8	43.4	71.1	83.3	91.2
Wal+SIFT+MAG	0.4	1.7	4.2	8.2	42.9	67.9	82.3	91.0
SupGlue	4.4	8.4	11.2	13.8	0.8	3.2	7.8	18.6
SupGlue+Mag	2.2	5.4	8.6	12.0	2.5	9.6	18.0	26.1
SpNet	3.5	8.1	11.9	16.8	6.8	24.9	45.7	64.1
SpNet+MAG	0.0	0.0	0.0	0.0	0.7	3.2	8.2	18.7

Table 5.4: Completenes evaluation of COLMAP pipeline with various correspondence search methods using ETH3D benchmarking system and two ETH3D datasets, electro and pipes.

ETH3D F1-score results [%]								
Dataset no.	1				2			
tolerances[m]	0.02	0.05	0.10	0.20	0.02	0.05	0.10	0.20
SIFT	5.1	21.3	37.6	**51.2**	**64.2**	**83.3**	**90.5**	**95.1**
SIFT+MAG	4.6	13.5	22.2	27.4	0.4	4.6	58.7	83.9
Wal+SIFT	**12.7**	**26.6**	**38.1**	49.1	52.2	79.3	89.1	94.5
Wal+SIFT+MAG	0.7	3.1	7.5	14.4	49.5	76.0	88.4	94.4
SupGlue	8.4	15.3	20.0	24.1	1.5	5.6	12.6	26.7
SupGlue+Mag	4.2	10.2	15.6	21.3	4.7	16.6	28.6	39.8
SpNet	6.7	14.8	20.9	28.4	9.4	32.8	53.7	72.2
SpNet+MAG	0.0	0.0	0.0	0.0	1.1	5.1	11.9	23.5

Table 5.5: F1-score evaluation of COLMAP pipeline with various correspondence search methods using ETH3D benchmarking system and two ETH3D datasets, electro and pipes.

Chapter 6

Conclusions

The goal of this thesis was to compare the performance of several correspondence search methods in the 3D reconstruction pipeline used on specific datasets, containing challenging factors such as weakly textured areas as walls and illumination changes between images.

To conduct a comparison, we first introduced the methods of FE and FM in a theoretical part. The approaches included in the introductory part were both hand-crafted algorithms and NNs. We introduced the most common feature extractor and descriptor, SIFT descriptor, and robust model estimators LO-RANSAC and MAGSAC++. Further, we included the whole preprocessing pipeline preceding the FE. This pipeline was created by denoising, RGB to gray-scale conversion, and content enrichment stage, where we examined the Wallis filter and CLAHE filter. From NNs, we introduced principles applied in SparseNCNet. In the next part, we also proposed enhancements for SparseNCNet. These enhancements were motivated by the low track length of keypoints and thus failures of 3D reconstruction. We also provided re-training information and results.

After the introductory part, we have shown our method of collecting GT image matches of the challenging scenes using Microsoft Hololens. We then used the GT matches to create an FM benchmarking system. This system was then employed to search the methods for the best-performing parameter combinations. Besides the parameters, we also chose between the robust model estimators LO-RANSAC and MAGSAC++. We used the latter for further benchmarking because the MAGSAC++ gave higher numbers of correct matches with the same precision. Similarly, among content enrichment methods utilized in preprocessing pipeline, we chose the Wallis filter over CLAHE because of the higher benchmark scores.

With the obtained methods and parameters, we then proceeded to the main comparison. This comparison included two parts. The first part evaluated the performance of all the methods on purely FE and FM results using our collected GT matches. The evaluated metrics were the absolute average number of correct matches, the average precision, and the average distribution evaluation. The hand-crafted approaches were highly outperformed by the NNs in all the categories in this part. The best results were brought by Superpoint and Superglue combination, even finding correspondences on the completely textureless areas.

The second part employed ETH3D benchmarking system using datasets pipes and delivery area. First, we incorporated each of the matching methods' results into the COLMAP pipeline, and the dense point clouds generated by the pipeline were then aligned with the GT data. Such aligned data were then fed into the ETH3D benchmark to evaluate the COLMAP-reconstructed dense 3D point clouds using the ETH3D laser-scan-collected GT. Worth mentioning is that all of the methods carried better results on the ETH3D datasets without MAGSAC++ verification. These results mean either that COLMAP provides a better robust model estimation than MAGSAC++ or that the COLMAP model estimator is just better integrated into the 3D reconstruction pipeline. The best overall scores in the second part were achieved using the Wallis filter on the pipes dataset and pure SIFT on the delivery area dataset, leaving the NN-based methods far behind.

Undoubtedly, this topic offers a lot of space for further improvement and further work, mainly using NNs in the correspondence search. As we found out in the thesis, these networks can create matches where the hand-crafted approaches can not, and what to the number and distribution of correspondences outperform the hand-crafted algorithms by far. However, the overall quality of the 3D reconstruction is generally worse for NNS. The worse reconstruction quality is probably caused mainly by the resizing of images, which negatively affects the accuracy of keypoint locations. The resizing is native in the case of SuperGlue, and the keypoints are not localized with subpixel precision but only with the resolution of the resized image. Although the SparseNCNet generates keypoints with subpixel precision, they need to be rounded to a small grid in order to increase the track length and create the reconstruction. When we used the grid with higher resolution, we did not obtain any reconstruction at all due to the small track lengths of keypoints. Note that we used 26 images in 5 datasets in the first part of the benchmarking. However, more GT data would help to confirm or refute our conclusions, for which are all the benchmark implementations ready.

Further work on this topic may also focus on redesigning the loss-function of the SparseNCNet training network to create matches more suitable for 3D reconstruction.

Appendix A

Bibliography

[1] J. Pritts, "Sunflowers scale space hessian." https://cw.fel.cvut.cz/old/courses/ucuws17/labs/03_1_cspond/start, 2010.

[2] T. Schops, J. L. Schonberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3260–3269, 2017.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[4] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms." http://www.vlfeat.org/, 2008.

[5] D. Chotrov, Z. Uzunova, Y. Yordanov, and S. Maleshkov, "Mixed-reality spatial configuration with a zed mini stereoscopic camera," 11 2018.

[6] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, pp. 1520–1528, 2015.

[7] I. Rocco, R. Arandjelović, and J. Sivic, "Efficient neighbourhood consensus networks via submanifold sparse convolutions," in *European Conference on Computer Vision*, 2020.

[8] M. Gaiani, F. Remondino, F. I. Apollonio, and A. Ballabeni, "An advanced pre-processing pipeline to improve automated photogrammetric reconstructions of architectural scenes," *Remote sensing*, vol. 8, no. 3, p. 178, 2016.

[9] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[10] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, “Pixelwise view selection for unstructured multi-view stereo,” in *European Conference on Computer Vision (ECCV)*, 2016.

[12] O. Miksik and K. Mikolajczyk, “Evaluation of local detectors and descriptors for fast feature matching,” in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 2681–2684, IEEE, 2012.

[13] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superglue: Learning feature matching with graph neural networks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4938–4947, 2020.

[14] J. Revaud, P. Weinzaepfel, C. De Souza, N. Pion, G. Csurka, Y. Cabon, and M. Humenberger, “R2d2: repeatable and reliable detector and descriptor,” *arXiv preprint arXiv:1906.06195*, 2019.

[15] AliceVision, “Meshroom: A 3D reconstruction software.,” 2018.

[16] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[17] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[18] M. Polic, S. Steidl, C. Albl, Z. Kukelova, and T. Pajdla, “Uncertainty based camera model selection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5991–6000, 2020.

[19] K. Ondrej, “Comparison of correspondence search methods.” `https://gitlab.fel.cvut.cz/kafkaon1/comparison-of-correspondence-search-methods`, 2021.

[20] M. Brown and D. G. Lowe, “Automatic panoramic image stitching using invariant features,” *International journal of computer vision*, vol. 74, no. 1, pp. 59–73, 2007.

[21] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, “Netvlad: Cnn architecture for weakly supervised place recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5297–5307, 2016.

[22] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii, “Inloc: Indoor visual localization with dense matching and view synthesis,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7199–7209, 2018.

[23] C. Choy, J. Gwak, and S. Savarese, “4d spatio-temporal convnets: Minkowski convolutional neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3075–3084, 2019.

[24] S. Se, D. Lowe, and J. Little, “Vision-based mobile robot localization and mapping using scale-invariant features,” in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 2, pp. 2051–2058, IEEE, 2001.

[25] M. Toews, W. Wells III, D. L. Collins, and T. Arbel, “Feature-based morphometry: Discovering group-related anatomical patterns,” *NeuroImage*, vol. 49, no. 3, pp. 2318–2327, 2010.

[26] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions,” *Image and vision computing*, vol. 22, no. 10, pp. 761–767, 2004.

[27] M. Ghahremani, Y. Liu, and B. Tiddeman, “Ffd: Fast feature detector,” *IEEE Transactions on Image Processing*, vol. 30, pp. 1153–1168, 2020.

[28] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *2011 International conference on computer vision*, pp. 2564–2571, Ieee, 2011.

[29] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, “Brief: Computing a local binary descriptor very fast,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1281–1298, 2011.

[30] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, “A comparison of affine region detectors,” *International journal of computer vision*, vol. 65, no. 1, pp. 43–72, 2005.

[31] H. Zhou, T. Sattler, and D. W. Jacobs, “Evaluating local features for day-night matching,” in *European Conference on Computer Vision*, pp. 724–736, Springer, 2016.

[32] A. M. Andrew, “Multiple view geometry in computer vision,” *Kybernetes*, 2001.

[33] M. Muja and D. Lowe, “Flann-fast library for approximate nearest neighbors user manual,” *Computer Science Department, University of British Columbia, Vancouver, BC, Canada*, vol. 5, 2009.

[34] Z. Kukelova, J. Heller, M. Bujnak, A. Fitzgibbon, and T. Pajdla, "Efficient solution to the epipolar geometry for radially distorted cameras," in *Proceedings of the IEEE international conference on computer vision*, pp. 2309–2317, 2015.

[35] O. Chum, J. Matas, and J. Kittler, "Locally optimized ransac," in *Joint Pattern Recognition Symposium*, pp. 236–243, Springer, 2003.

[36] D. Barath, J. Matas, and J. Noskova, "MAGSAC: marginalizing sample consensus," in *Conference on Computer Vision and Pattern Recognition*, 2019.

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

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[39] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[40] R. Huang, J. Pedoeem, and C. Chen, "Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 2503–2510, IEEE, 2018.

[41] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 224–236, 2018.

[42] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-net: A trainable cnn for joint description and detection of local features," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8092–8101, 2019.

[43] R. Ranftl and V. Koltun, "Deep fundamental matrix estimation," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 284–299, 2018.

[44] K. M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua, "Learning to find good correspondences," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2666–2674, 2018.

[45] X. Shen, C. Wang, X. Li, Z. Yu, J. Li, C. Wen, M. Cheng, and Z. He, "Rf-net: An end-to-end image matching network based on receptive field,"

in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8132–8140, 2019.

[46] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic, "Neighbourhood consensus networks," *arXiv preprint arXiv:1810.10510*, 2018.

[47] Q. K. Al-Shayea and M. S. Al-Ani, "An efficient approach to 3d image reconstruction," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 8, p. 35, 2016.

[48] G. Guidi, S. Gonizzi, and L. Micoli, "Image pre-processing for optimizing automated photogrammetry performances," in *ISPRS Technical Commission V Symposium*, vol. 2, pp. 145–152, ISPRS, 2014.

[49] C. S. McCamy, H. Marcus, J. G. Davidson, *et al.*, "A color-rendition chart," *J. App. Photog. Eng*, vol. 2, no. 3, pp. 95–99, 1976.

[50] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[51] I. Ram, M. Elad, and I. Cohen, "Image denoising using nl-means via smooth patch ordering," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1350–1354, IEEE, 2013.

[52] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, "Denoising prior driven deep neural network for image restoration," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 10, pp. 2305–2318, 2018.

[53] K. Zhang, W. Zuo, and L. Zhang, "Ffdnet: Toward a fast and flexible solution for cnn-based image denoising," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4608–4622, 2018.

[54] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, "Cycleisp: Real image restoration via improved data synthesis," in *CVPR*, 2020.

[55] R. H. Wallis, "An approach for the space variant restoration and enhancement of images," in *Proc. Symposium on Current Mathematical Problems in Image Science, Nov., 1976*, 1976.

[56] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[57] M. Bujnak, Z. Kukelova, and T. Pajdla, "A general solution to the p4p problem for camera with unknown focal length," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[58] C. Aholt, S. Agarwal, and R. Thomas, "A qcqp approach to triangulation," in *European Conference on Computer Vision*, pp. 654–667, Springer, 2012.

[59] S. Agarwal, K. Mierle, and Others, "Ceres solver." `http://ceres-solver.org`.

[60] Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pp. 1–2, IEEE, 2018.

[61] P. Hübner, K. Clintworth, Q. Liu, M. Weinmann, and S. Wursthorn, "Evaluation of hololens tracking and depth sensing for indoor mapping applications," *Sensors*, vol. 20, no. 4, p. 1021, 2020.

[62] M. Matl, "Pyrender." `https://github.com/mmatl/pyrender`, 2019.

[63] Y. Mäkinen, L. Azzari, and A. Foi, "Collaborative filtering of correlated noise: Exact transform-domain variance for improved shrinkage and patch matching," *IEEE Transactions on Image Processing*, vol. 29, pp. 8339–8354, 2020.

[64] I. Jazayeri and C. S. Fraser, "Interest operators for feature-based matching in close range photogrammetry," *The Photogrammetric Record*, vol. 25, no. 129, pp. 24–41, 2010.