



MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Vávra Václav** Personal ID number: **316939**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Computer Science**
Study program: **Open Informatics**
Specialisation: **Artificial Intelligence**

II. Master's thesis details

Master's thesis title in English:

Two-view matching of image containing planar surface exploiting monodepth estimation

Master's thesis title in Czech:

Matching pohledů na scény s planárními povrchy

Guidelines:

1. Familiarize yourself with the problem of matching images, focusing on situations when they were acquired from very different viewpoints or where other acquisition conditions changed dramatically.
2. Consider mainly images containing planes, e.g with buildings and indoor images.
3. Familiarize yourself with monodepth algorithms and choose a suitable one.
4. Propose modifications of the standard view matching pipeline [2], exploiting the estimated depth information, to accelerate, simplify, robustify or extend the applicability of the pipeline.
5. Evaluate the proposed modifications or algorithm on the standard datasets [1,3].

Bibliography / sources:

- [1] Toft, Carl; Turmukhambetov, Daniyar; Sattler, Torsten; Kahl, Fredrik; Brostow, Gabriel J.; Single-Image Depth Prediction Makes Feature Matching Easier, ECCV 2020
[2] P. Pritchett and A. Zisserman. Wide baseline stereo matching. In ICCV, 1998.
[3] D. Mishkin, J. Matas, M. Perdoch. MODS: Fast and Robust Method for Two-View Matching. CVIU 2015.

Name and workplace of master's thesis supervisor:

Mgr. Dmytro Mishkin, Ph.D., Department of Cybernetics, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **21.02.2021** Deadline for master's thesis submission: **04.01.2022**

Assignment valid until: **19.02.2023**

Mgr. Dmytro Mishkin, Ph.D.
Supervisor's signature

Head of department's signature

prof. Mgr. Petr Páta, Ph.D.
Dean's signature

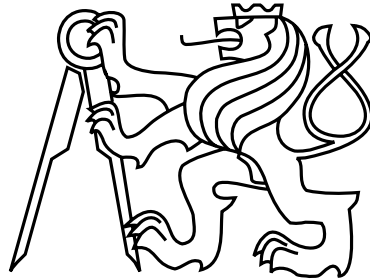
III. Assignment receipt

The student acknowledges that the master'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 master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Diploma thesis

**Extreme two-view matching of image containing planar
surfaces**

Ing. Václav Vávra

Supervisor: Dmytro Mishkin, PhD

Study Programme: Open Informatics, Master program

Field of Study: Artificial intelligence

January 4, 2022

Acknowledgements

I would like to thank my supervisor, Dmytro Mishkin, PhD, for his support. His advice has always helped me and made me think about what actually mattered. Also, he has always swiftly helped me with various tasks, be it setting up accounts, contacting the right people or providing the needed information. Last but not least he has been very time flexible and always there to help. I would also like to thank prof. Jiří Matas for his advice and that he provided me with this topic.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 4. 1. 2022

.....

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis.

In Prague on January 4, 2022

.....

Abstract

We consider the problem of matching images, which are taken under the significant viewpoint changes, by computing the rectification homography with the help of depth predictions from monocular images as recently proposed by Toft et al. [27]. We improve a method by Toft et al. by revisiting its components and validate the changes on the the dataset for Strong Viewpoint Changes, also introduced by Toft et al. Finally, we propose to replace normal-from-depth-based rectification via homography with an estimator exploiting local affine features coming from AffNet model [17], similarly to the method proposed by Rodríguez et. al [24]. The proposed method improves both in speed and accuracy compared to the original methods Rodríguez et. al [24] and Toft et al. [27].

Keywords - Local Feature Matching, Image Matching, Depth Estimation

Abstrakt

Uvažujeme problém matchování obrázků, které jsou pořízeny při velké změně zorného úhlu, a výpočet rektifikační homografie s pomocí predikce hloubky z monokulárních obrázků, jak bylo nedávno navrženo Toftem a kol. [27]. Metoda navržena Toftem a kol. byla zlepšena revidováním jednotlivých komponent a validována na datasetu navrženém a použitém tamtéž. Dále je navržena rektifikace afinní transformací pomocí lokálních afinních příznaků modelovaných AffNetem [17], podobně, jak bylo navrženo Rodríguezem a kol. [24]. Navržená metoda vykazuje lepší přesnost i vyšší rychlost než původní metody.

Klíčová slova - Matchování lokálních příznaků, Image Matching, Odhad hloubky

Contents

1	Introduction	1
1.1	Thesis contributions	2
1.2	Thesis structure	2
2	Feature rectification using the depth information	3
2.1	Depth estimation	3
2.2	Estimating the normals	4
2.3	Sky segmentation	6
2.4	Filtering based on singular values	6
2.5	Planes detection	6
2.5.1	Normals clustering	7
2.5.2	Detected planar surfaces as connected components	10
2.6	Algorithm tuning	10
2.7	Feature rectification	12
3	Rectification via affine maps using AffNet	15
3.1	Motivation	15
3.2	Covering the space of tilts	16
3.3	Combining AffNet shape estimators and depth maps	19
4	Experiments and results	21
4.1	Dataset	21
4.2	Experiments setting	22
4.3	Implementation	22
4.4	Results	23
4.4.1	Tuning the processing parameters	23
4.4.2	Different features	25
4.4.3	HardNet features and AffNet shape estimator	26
4.5	EVD dataset	31
5	Conclusion	33
6	Bibliography	35

List of Figures

2.1	Depth estimation of a scene by Megadepth CNN	4
2.2	Normals visualization	5
2.3	Sky filtering	6
2.4	Normals clustering	7
2.5	Pseudo-code for the clustering algorithm	9
2.6	Different quantiles for filtering based on singular values ratio	11
2.7	Different refinements of the initial bucket centers	12
2.8	Simple SVD and weighted SVD	13
2.9	Handling of antipodal points	14
2.10	Pseudo-code for the rectification via homography	14
3.1	Geometric interpretation of a linear map decomposition	16
3.2	Space of tilts and its covering	18
3.3	Pseudo-code for the new method using both AffNet shapes and the depth maps	19
4.1	The effect of SVD weighting on scene 1	23
4.2	Handling of antipodal points	24
4.3	Morphological variations	24
4.4	Cluster refinement variants	25
4.5	Different features (I) on scene 1	26
4.6	Different features (II) on scene 1	27
4.8	Rectification of HardNet features via affine maps with different parameters	27
4.7	HardNet rectified via affine maps and homography, SIFT rectified by homography	28
4.9	Original and new method based on AffNet shapes estimates	29
4.10	Computation complexity of the original and new AffNet based methods	30
4.11	Main methods on the whole dataset	30

Chapter 1

Introduction

Image matching is an important part of the 3D reconstruction [3, 25], SLAM(Simultaneous localization and mapping) [18], panorama stitching [5] and other applications. One of the key ingredients of the robust image matching pipeline is good local features [15]. SIFT features [13] set a high standard for all other local feature types. One of its core property is its rotation and scale invariance. The rotation invariance is achieved by normalizing according to the dominant direction, whereas the scale invariance achieved by simulation [24] in the scale space. However, SIFT features, as well as others [14, 10, 6] often fail under extreme viewpoint changes [28, 16].

A successful attempt in fixing this issue was done in ASIFT [28], where the invariance is augmented to all affine transforms. It is achieved by test-time augmentation technique of simulating a set of transforms, which effectively covers all possible affine transforms on a given granularity. This inspired many other works [24], some of them developing specialized methods for e.g. matching [16]. The idea of inverting a distortion caused by the camera seeing a planar surface from general angle was also used to account not only for affine, but for general perspective transformations, be it with the help of detected vanishing points [11] or repeating patterns [20].

These methods narrow the set of possible transformation using an additional information discovered in the input image. Other use cases permit to have real additional information, which is not present in the input image, such as when the 3D geometry of the scene is known [9]. On the other hand, an estimate of the scene geometry can be obtained from the input image. This became possible with the advent of depth estimating CNNs ([7], [12], [21]). Toft et al. [27] showed such a method using the depth estimates from a CNN. This method detects planar surfaces in the scene and undoes the effect of perspective distortion using rectifying homography. The rectified features then show an improved performance in the task of two-view matching.

Another CNN handling distortion of images is AffNet [17]. It is trained by synthetic data to estimate an applied random affine transformation. It can then be used to transform local features to canonical shape by HardNet feature descriptor, which itself is also based on a CNN. Here we abandon the realm of precise modeling and even explicability (what makes the shape canonical for AffNet? Which steps are taken by the description computation for HardNet?). Even more interestingly, the original idea from ASIFT, developed further in [23] uses AffNet shape estimates to more effectively cover the relevant parts of the space of tilts [24] of an image, which is covered completely in ASIFT. This thesis builds on top of the works by Toft et al. [27] and Rodríguez et al. [23] and attempts to improve them.

1.1 Thesis contributions

This thesis builds on the work by Toft et al. [27] and Rodrigues et al.[23]. It improves on the method by Toft et al. by revisiting the various implementation details and exploring possible improvements. The first contribution is the introduction of weighted SVD during the computation of surface normals, which better smoothes out the data and results in better local features as shown in the experiments. The normals clustering is improved by a faster and more accurate method based on bucket voting. Also, the method is made more general by abandoning the assumption that the planes are orthogonal, even though only to a certain extent. The addition of semantic segmentation to filter out areas corresponding to sky was very straightforward, yet useful. These are all improvements made in different steps before the rectification itself.

As far as the rectification is concerned, it was observed that the method works well with HardNet [14] features. Better yet, its affine shape estimator AffNet [17] was used to develop a new method based on rectifications via affine maps, which is more accurate than the method based on rectifications via homographies. This new method is a combination of methods based on covering the space of tilts [23] [24] and the original method using the information about the depth [27].

1.2 Thesis structure

The second chapter describes the original method described in Toft et al.[27] of local features rectification via homographies based on depth estimation. Its improvements are described there as well, one step of the pipeline at a time. It also shows experiments that helped tune the parameters used.

The third chapter covers the new method combining information from a depth estimating CNN and from AffNet[17] affine shape estimator used in HardNet[14] local features.

Experimental results are presented in the fourth chapter. It covers a large number of experiments measuring performance of the task of two-view matching on a dataset also used in Toft et al.

Finally, the conclusion is drawn in the last chapter.

Chapter 2

Feature rectification using the depth information

We first present a main idea of the method described by [27] and then focus on our improvement to its pipeline. The method estimates 3D planes in the image and the rectification transformation of the image, after which the planes are seen from a fronto-parallel view. This is to undo the effect of perspective projection when originally seen from a slanted view.

The rectification transformation is computed from the planes orientation and segmentation. Planes are detected and segmented by clustering surface normals, which are estimated from the pixel-wise depth information. Given that for many practical scenarios the depth information is not available directly, it is provided by the convolution neural network, trained for depth estimation from monocular images.

Parts of the image which do not belong to any patch corresponding to planar surfaces are handled as usual, that means the keypoints are found and their descriptors are computed unrectified.

Each of its steps is discussed in detail in the following subsections.

2.1 Depth estimation

Successful implementations of CNNs estimating depth from monocular images include MegaDepth [12], Monodepth2 [7] and MiDaS [21]. Toft et al. [27] used also their own implementation of depth estimating CNN with an architecture similar to the one used in Monodepth2. These CNNs usually predict unscaled depth per pixel, less frequently they predict normals or 3D plane equations. However, as noted in [27], normals are either used to regularize depth or trained exclusively on indoor scenes because supervised data are difficult to collect for outdoor scenes. Therefore it is further assumed that the depth map with unscaled depth per pixel is readily available.

Formally this means that for each pixels at image coordinates (i, j) the depth d^{ij} is known, which is the length of the projection ray between the camera center and a point X^{ij} in 3D space:

$$\begin{aligned} d^{ij} &= \|X^{ij}\|, \text{ s.t.} \\ PX^{ij} &\cong (i, j, 1)^T, \end{aligned} \tag{2.1}$$

where P is the projection matrix. Knowing P the 3D points X^{ij} can be reprojected at each pixel, which is in fact done during estimation of the surface normals. The depth estimates is given up to an unknown scale factor, same for all pixels in an image. This is equivalent to not knowing



Figure 2.1: Depth estimation of a scene by Megadepth CNN. The original image on the left and the depth map visualized as a grayscale image on the right.

the scale of the length unit of the projected 3D space. No further computations depend on knowing this scale. Note that this notion of depth is really the length of the projection ray and hence it is different to z-buffer value.

In all experiments described here the MegaDepth CNN [12] was used for the depth estimation. The input image fed to this CNN needs to have a maximum dimension of 512 pixels. The images as the inputs to the CNN - if too large - were hence downscaled to 512 pixels along the larger dimension with the other dimension chosen as the multiple of 32 that best preserves the original aspect ratio (exactly in accordance with the original method[27]). The depth map is then kept downscaled during the computation of the surface normals. See Figure 2.1 for a visualization of a depth map computed by MegaDepth. Note that Toft et al. trained their own neural net, which performed better than the aforementioned ones (see Figure 5 of the Supplementary material in [27]), but as it was not made publicly available, MegaDepth was used in the implementation described here.

2.2 Estimating the normals

The surface normals at each pixel are estimated by fitting a plane through 5x5 pixel window backprojected into the 3D space (also in accordance with [27]). This is done via SVD decomposition. For a pixel with coordinates (i_0, j_0) , for which the normal is computed (center of the window), there are 25 row vectors $X^{ij\top}$ stacked in the matrix A

$$A_k = X^{ij\top} \quad (2.2)$$

For the indices it holds

$$\begin{aligned} k &= (i - i_0 + 2) * 5 + j - j_0 + 2, \\ |i - i_0| &\leq 2, \\ |j - j_0| &\leq 2. \end{aligned} \quad (2.3)$$



Figure 2.2: Visualization of surface normals as computed via the weighted SVD. The (x, y, z) components of the unit normals correspond in order to the blue, green and red channel respectively.

We compute SVD of the centered data points, i.e. where the average over rows is deducted from each row:

$$\begin{aligned}\hat{A}_i &= A_i - \frac{\sum_{j=1}^{25}(A_j)}{25}, \\ \hat{A} &= U\Sigma V^\top.\end{aligned}\tag{2.4}$$

The estimated value of a unit normal vector to a surface at (i_0, j_0) is then the last column of V - the right-singular vector corresponding to the smallest singular value (assuming singular values in Σ in descending order):

$$\hat{n}^\top = V_3^\top\tag{2.5}$$

We propose to suppress the influence of the pixels further away from the window center by do a weighting. The data points in \hat{A} are scaled with weights $(w_i)_{i=1}^{25}$:

$$\begin{aligned}\hat{A}^w &= \text{diag}(w_1, \dots, w_{25})\hat{A} \\ \hat{A}^w &= U\Sigma V^\top.\end{aligned}\tag{2.6}$$

The weights w_i were set to the values of Gaussian probability density function centered at (i_0, j_0) :

$$w_k \propto \exp\left(\frac{((i - i_0)^2 + (j - j_0)^2)}{-2\sigma^2}\right).\tag{2.7}$$

The normalization constant does not affect the result, as it just scales the singular values, not the singular vectors (therefore it neither affects the singular values ratio, see further). The experiments showed that the weighting before SVD improves the performance both in terms of accuracy of the estimated normals (see Figure 2.8, left), but also in terms of the accuracy of estimated relative pose in matching (see Figure 4.1). Also, as observed on the accuracy of the estimated normals, the best value for σ was 0.8 (See Figure 2.8, right), which was also used for further experiments.

2.3 Sky segmentation



Figure 2.3: Sky filtering and filtering based on the singular values ratio. From left to right: original image, sky mask and mask based on singular values ratio on top of the sky mask ($\alpha_q=0.6$ used). Yellow pixels are considered for further processing (normals clustering), ultra violet pixels are ignored.

It was observed that the regions of the images corresponding to the sky for outdoor scenes are very noisy in the depth data (as also pointed out by the authors of Megadepth [12]), while at the same time they can easily be detected by semantic segmentation. Therefore it was used to filter out regions of the images detected as sky prior to clustering of the normals. For this the segmentation model by MIT CSAIL Computer Vision [29] was used. This filtering simply narrows the regions of the image for which the normals clustering and possibly the rectification is considered.

2.4 Filtering based on singular values

To filter out parts of the scene which do not correspond to planar surfaces, we have tried to filter out pixels, for which the points of backprojected to 3D space from the 5x5 window did not have the smallest principal axis significantly smaller than the others. For each pixel the ratio between the two smallest singular values were computed (assuming singular values are sorted in descending order):

$$r = \frac{\Sigma_{33}}{\Sigma_{22}} . \quad (2.8)$$

The case when the denominator is zero can simply be ignored as this would mean that the backprojected points from a square window all lie on a line. Only those pixels with r within α_q -quantile of its distribution over the image were considered. This step, however related to the SVD decomposition, was placed only after the sky filtering, considering the values of r only on the sky mask, so that these two steps would not filter out the same pixels. This filtering is conceptually promising, but did not show consistent improvement and therefore was not used in further experiments (see Figure 2.6).

2.5 Planes detection

Once the surface normals per pixel in the scene have been estimated, contiguous regions of the image with similar normals are searched for as they are assumed to correspond to planar regions in

the scene. The unit normals corresponding to the image pixels are distributed on the unit sphere. If we take the normals as coming out of the surfaces ($\mathbf{n} = (0, 0, -1)$ if on a planar surface orthogonal to the camera axis), the normals take up parts of the unit sphere with $z \leq 0$ and only a subset of the unit sphere with positive z . This is affected by the direction of view. The normals defined in this way would only form obtuse angles with any projection ray originating in the camera center. Surfaces with normals forming acute angles with the projection ray cannot be seen as they are on the reverse side of the seen object. See Figure 2.4 for a visualization of the distribution of normals estimates on the unit sphere.

2.5.1 Normals clustering

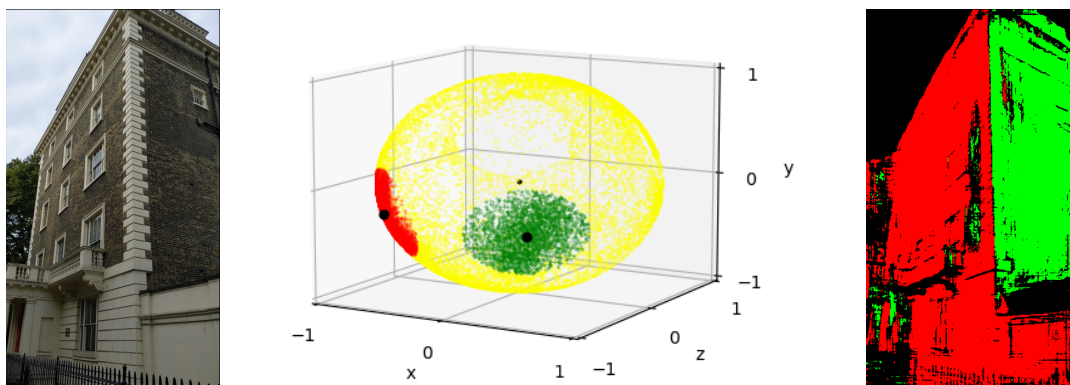


Figure 2.4: Normals clustering. Left: original image. Center: normals distributed on the unit sphere – only every 10th normal is visualized. Yellow points correspond to normals outside the clusters, green and red points are normals of the detected planes. Right: detected planes corresponding to the green and red cluster in the image domain.

K-means normals clustering in Toft et al.[27]. Normals are clustered in up to three clusters or dominant directions, while also enforcing orthogonality between these clusters. Each cluster also includes its antipodal point. The antipodal normals would correspond to e.g. opposing parallel walls. Even though the method is not stated explicitly, a natural way to handle the antipodal points together would be to duplicate the data and consider both estimated normal and its negative for each pixel. Then the normals will be symmetrical about the origin. Care then must be taken to avoid detecting the opposing clusters of these duplicated datapoints twice - one on each side of the sphere around the origin. This can be done by for example considering only cluster centers with positive z coordinate.

It should be noted however, that this duplication affects the size of the detected clusters. Clusters of surface normals which can appear in the scene along with its negatives (e.g. the aforementioned opposing walls) would appear to have equal size as others even though the individual surfaces are smaller - two opposing walls projected together on the same number of pixels as one wall faced straight on would form clusters of the same size.

On the other hand, such normals duplication naturally handles the cases where the SVD decomposition give the singular unit vector forming an acute angle with the projection ray as again it handles this vector and its negation the same way. If the duplication is not done, the correct way to handle these cases would be to check if the angles with the respective projection rays are indeed

obtuse and if not, the sign of the normal would be flipped. However, for simplicity this is not done here even if the handling of the antipodal points is disabled.

In the implementation described here the spherical k-means clustering was not used as it neither achieved a good speed nor a good accuracy. There are as many data normals as there are pixels in the depth map (denoted by $|D|$), or twice as many after the duplication allowing to handle the antipodal points. The normals are clustered into k clusters. This translates to $|D|.i$ operations (i denotes the number of iterations), where there are usually at least tens of iterations and the parallelization can only be done across D as the operations across iterations cannot be parallelized. Furthermore, it is not clear how to quickly estimate the number of detected planes. To estimate the number of clusters for k-means the algorithm is usually run n times for each possible number of clusters. This further increases the computation complexity.

As for the inferior accuracy, our hypothesis is that it is because k-means reflects the global distribution of the data, which is most likely not desired here. Normals too divergent from any cluster center probably do not belong to any planar surface and should not be part of any cluster. Within standard k-means however, they would end up in one of the clusters and negatively affect one of the cluster centers and thus one of the detected plane normal. Rather than global clusters, modes of the distribution of the unit normals should be searched for.

Our approach. The clusters of normals are found by voting among a set of buckets, represented by their centers, a set of N points distributed on the unit hemisphere with $z \leq 0$ roughly equidistantly with the spiraling algorithm [1]). The number of normals closest to each bucket center is computed. Then the bucket center with the highest count is selected as the initial guess of the plane normal. All unit normals forming an angle smaller than the threshold angle α_θ are considered to be part of the cluster. This initial guess is then improved either by simply taking a mean of the points of the initial cluster or by starting a mean-shift algorithm from this initial guess. Again all normals within the threshold angle α_θ from the resulting cluster center will belong to the cluster. Note that this will be a typically larger number of normals than the number of initial votes, as the bucket centers are quite close to each other (forming angles smaller than $2\alpha_\theta$). The search continues and the next candidate point is selected, however only those cluster centers are considered which form an angle greater than $c\alpha_\theta$ with any cluster center that has already been found. The parameter c is chosen to be greater than 2, which avoids the possibility of normals being part of multiple clusters. The search continues as long as the next bucket has higher votes than some threshold p_θ , which is given as the fraction of all the normals. This threshold is set to:

$$p_\theta = \frac{0.13}{30} \alpha_\theta \alpha_q, \quad (2.9)$$

where α_q is the parameter for the filtering based on the singular value ratio. This is done in attempt to make the threshold have the same effect for different values of the parameters (the surface of the cluster grows roughly quadratically with respect to α_θ , but the number of normals within a cluster is assumed to grow slower than linearly with the growing area of the cluster on the unit sphere; the number of the normals would grow roughly linearly with respect to α_q). The numerical values were chosen to make $p_\theta = 0.13$ degrees for $\alpha_\theta = 30$ degrees and $\alpha_q = 1.0$ (filtering based on the singular values ratio is disabled), which were the values for the initial experiments. The values for the rest of the parameters were $N = 300$ and $c = 2.5$. See Figure 2.5 for pseudo-code of this algorithm.

As can be seen in Figures 2.6 to 2.8, that the rather surprisingly high value of 35 degrees for α_θ consistently resulted in the based accuracy in terms of the estimated normals. This value was then used in all further experiments including the two-view matching experiments. However, with

```

def compute_clusters(normals, mean_shift_type, max_clusters):

    size = normals.height * normals.width
    # minimal number of normals in the cluster
    p_th = (0.13 / 30) * alpha_th * alpha_q * size

    candidates = equidistant_points_on_hemisphere(N)

    clusters = {}

    for c in candidates:
        c.count = closest(normals, c, candidates)

    candidates = candidates.sort(key=lambda c: c.count, order=DESCENDING)
    for c in candidates:
        if c < p_th:
            break
        elif algorithm == Alg.mean:
            c = cluster_mean(c, normals, alpha_th)
        elif algorithm == Alg.mean_shift:
            c = mean_shift(c, normals, alpha_th)

        if max_distance(c, clusters) < c * alpha_th:
            continue
        else:
            clusters.add(c)

    return clusters

```

Figure 2.5: Pseudo-code for the clustering algorithm. Parameter values used: $N = 300$ (number of initial clusters), $\alpha_{th} = 35$ (angle defining the extent of the cluster), $\alpha_q = 1.0$ (quantile used for filtering based on singular value ratios), $c = 2.5$ (factor defining the inhibition area around detected clusters), $algorithm = Alg.mean$ (simply taking a mean of the initial cluster.)

$c = 2.5$, the minimal angle between clusters is $c\alpha_{\theta} = 87.5$ degrees, which makes the angles close to the right angle just with this condition and to an extent serves as the explicit enforcing of orthogonality used in Toft et al.[27]. However the figures also show that ruling out this high value of 35 degrees, comparatively good accuracies are also achieved with $\alpha_{\theta} = 25$ degrees, which is the optimal value in the range of 0 to 30 degrees (values of $\alpha_{\theta} < 15$ degrees lead to even worse accuracies than for $\alpha_{\theta} = 15$). $\alpha_{\theta} = 25$ degrees may then be the right parameter value when non-orthogonal planes are frequently to be detected.

To summarize, the clustering algorithm a) finds the normals of the dominating planes as the detected cluster centers and b) assigns each pixel an index of the corresponding plane normal or possibly some default value indicating no assigned normal.

2.5.2 Detected planar surfaces as connected components

The next step is to find the contiguous regions in the image belonging to the same cluster. These regions will finally be considered to be part of one planar surface in the scene. The regions are found as connected components of pixels assigned to the same normal. Both 4- and 8-point connectivity was tried for this step. The patches of connected components are discarded if they do not cover some minimal number of pixels (r_θ). This threshold value is expressed as the ratio of the image surface. The value of 0.03 was used in the experiments.

There are possible extensions to this step, which aim at finding larger or more compact patches of pixels corresponding to one planar surface. Firstly, morphological closing with a circle-like kernel was tried to enlarge the detected patches (this may fill holes in the patches as well as merge close patches). Secondly, to make the patches simply connected (without holes), flood filling algorithm was employed. This is done by first filtering out the bounding box of the whole image from the patch, then running flood filling on the patch complement and then adding again the pixels from the patch which were filtered in the first step.

However it was observed that these extension didn't bring any consistent improvement. See the results in Figure 4.3. The flood filling may also fill out regions which are indeed not part of the detected plane or, worse yet, are actually part of another plane. The morphological closing, when the kernel is too big, may expand the regions of detected planes so that the overlap, which may then interfere with further steps of the method. The same goes for flood-filling. Therefore these refinements were abandoned and not used in further experiments.

These steps so far, even though there are multiple, remain fairly simple. Note that for example the cases when two collinear planes at different depths are projected to one contiguous patch are ignored, as they would result in one plane detected by this method. Even though this could be handled by taking into account the depth during the clustering, it is not necessary as these two collinear planes would be the rectified the same way anyway, so such cases can safely be ignored.

2.6 Algorithm tuning

For the steps described so far experiments were done on the scene 1 of the dataset used by [27] – see Chapter 4 for its description. The absolute difference between the right angle and the angle between two normals of the largest detected planes is used as a metric, as the most of images contain orthogonal planes as part of the buildings. The two largest planes were considered because various settings may detect more than two planes. They may also detect one or none, which is the reason why the average of these differences across 200 images from scene 1 was compared. The selection of images was random but fixed for all the experiments here.

This makes sense as scene 1 of the dataset contains exclusively orthogonal dominating planes - the images depict a street with a rectangular shaped building without much clutter. The original method in Toft et al.[27] enforced the orthogonality explicitly, which is considered to be a specific feature making it not very general. Therefore it was observed whether the orthogonal planes are also detected as orthogonal even without explicitly enforcing the orthogonality. Hopefully it will be clear that the methods tried to achieve this were generally applicable strategies, i.e. not something that would only artificially achieve the desired goal (i.e. render the detected planes as orthogonal). An exception to this may be the high value for the α_θ parameter.

The experiments are organized as follows: for each experiment, all parameters were set to the default (best performing) values, except two: one that was being tested and α_θ , which iterated through the set {15, 20, 25, 30, 35} (the values are in degrees) so that the observed parameter with

combination of different α_θ values. The best performing values of parameter (i.e. the fixed ones) are as follows: singular value ratio quantile $\alpha_q = 1.0$ (filtering effectively disabled), cluster refinement method - taking a simple mean, SVD weighting parameter $\sigma = 0.8$ pixels, handling of antipodal points - disabled, SVD weighting - enabled.

SVD ratio filtering. Figure 2.6 shows the accuracy depending of different values of α_θ and α_q . Rather counter-intuitively it shows that the filtering hurts the performance, usually the more significantly the more restrictive the filtering was (lower values of α_q). Therefore the filtering based on the singular values ratio was effectively disabled (which corresponds to the value of 1.0) in other experiments.

Cluster refinement strategies Figure 2.7 shows the plane orientation accuracy depending on the cluster center refinement strategy. No refinement means that simply the centers of the buckets with most votes were chosen as the cluster centers. The next option is to simply take a mean of all the normals within the initial cluster to get a refined cluster center (the cluster membership as refined as well according to the threshold value of α_θ). The last strategy is to start a mean shift algorithm from the initial cluster center. The uniform kernel was chosen spanning all normals within $\alpha_\theta/2$ degrees from current the center of the cluster. Simply taking the mean was the best performing option. Because of that and also because its simplicity and speed it was chosen as the refinement strategy for all other experiments. It is interesting to mention the absolute best accuracy attained with no refinement at $\alpha_\theta = 35$ degrees. This was considered to be rather an outlier, as among the 200 chosen images at least two planes were detected only in 121 of them, whereas most of the well performing choices detected at least 2 planes fairly consistently in around 170 images.

Figure 2.8, left, shows the influence of the weighted SVD on the accuracy. The weighted SVD clearly outperformed the unweighted variant with higher values of α_θ . Therefore it has been used for the rest of experiments.

Figure 2.8, right shows the results of tuning of the SVD weighting parameter σ . The best performing value was 0.8 pixels, with both higher and lower values performing worse. It is important to note that the 5x5 window on which the weights according to the normal distribution probability

Different quantiles for filtering based on singular values ratio

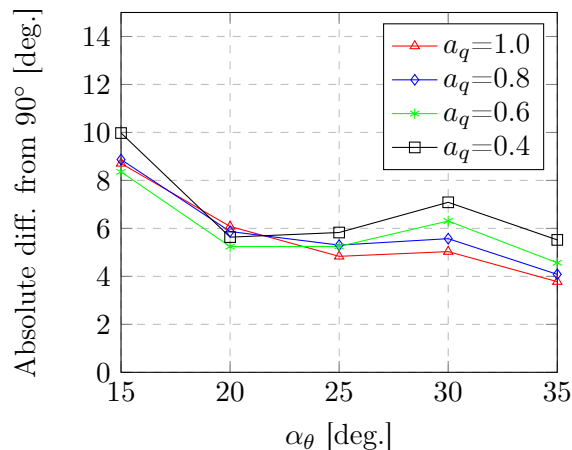


Figure 2.6: Different quantiles for filtering based on singular values ratio. The lower the value, the more restrictive the filtering is. The optimal value is $\alpha_q=1.8$, which effectively turns the feature off.

density function are applied only spans maximally $2\sqrt{2}$ pixels from the center for the pixels on the corners of the window.

On Figure 2.9 the accuracy was measured for handling of the antipodal points enabled and disabled respectively. The performance is very similar for both choices, which the option turned on slightly outperforming the case with the option turned off. However, as shown in 4, disabled handling of the antipodal points performs better in the task of estimating relative pose in two-view matching. Therefore the feature was turned off in the rest of experiments.

2.7 Feature rectification

In this phase the detected plane normals, each pixel membership to a patch and each patch membership to one plane normal have been computed. Each of the patches are now rectified one by one. As done in Toft et al.[27], this is achieved by applying a homography that rotates the camera so that its optical axis becomes parallel to the patch plane normal (i.e. the patch will be transformed to a fronto-parallel view). The smallest such rotation is used. In the original method, not the entire patch is rectified, but only parts of the planar surface seen at not too glancing an angle. A threshold value of 80° was used there as the maximum angle between the viewing ray from the camera and the surface normal and the rectified patch was cropped to contain only points seen at smaller angles. In the method described here a simplification has been made. The entire patch is ignored if the surface normal forms an angle with the camera optical axis greater than a threshold value. A threshold value of 75° was used here.

A heuristic that normalizes the size of the resulting rectified patch was used. The rectified patch is scaled by a factor of $\sqrt{\frac{2 \cdot |p|}{|bb|}}$, where $|p|$ are the number pixels of the original patch and $|bb|$ is the size of the smallest rectangular bounding box aligned with x and y axes including the whole transformed patch (before the scaling). This makes the resulting bounding box have exactly twice as many pixels as the original patch (of an arbitrary shape). This constant of one half was empirically found to keep

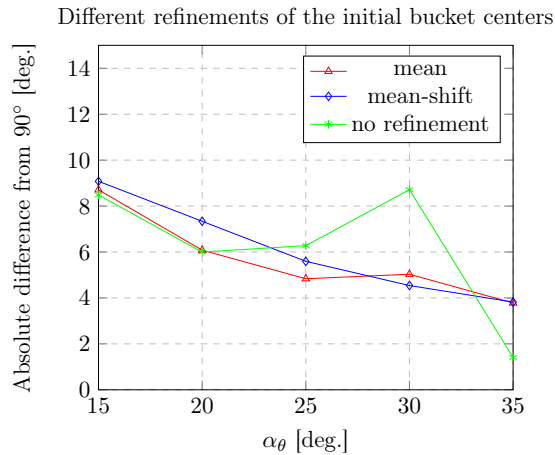


Figure 2.7: Different refinements of the initial bucket centers. The initial bucket centers are either not refined at all, a simple mean of all normals is computed to refine the initial value, or mean-shift algorithm is started from the initial guess. Note that while the best value was achieved with no refinement, the number of images with at least two detected planes was in this case much lower (121) than in other cases (around 170).

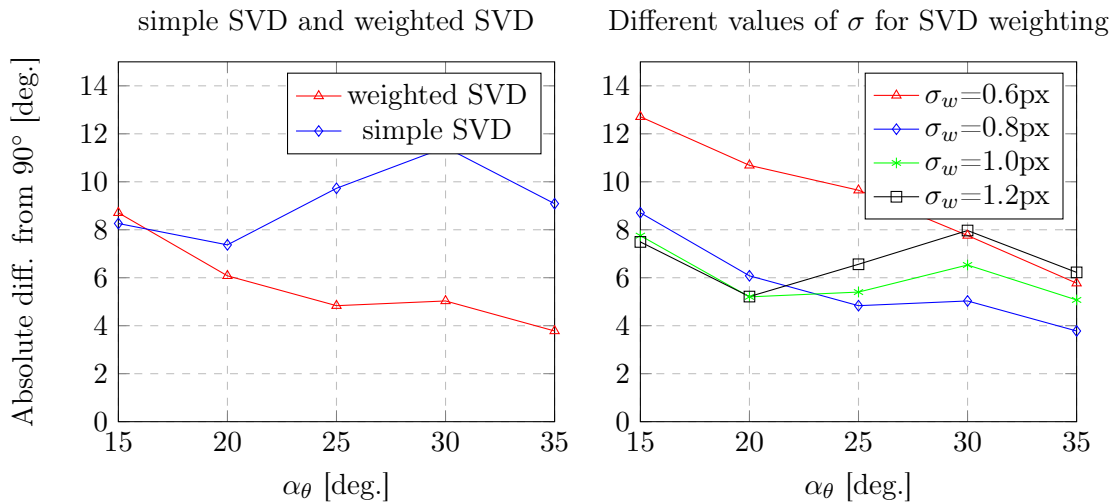


Figure 2.8: Left: simple SVD vs weighted SVD for plane orientation estimation. Right: Different values of σ_w for SVD weighting. The value of 0.8 pixels for this parameter outperforms both higher and lower values.

the size of the resulting rectified patch reasonable.

The resulting bounding box is also computed after the scaling to account for the final translation, which moves the rectified patch to the field of view, and for cropping the parts of the transformed image which were not parts of the original patch. Finding the rectifying homography is summarized in Figure 2.10. Note that transforming only the minimal parts of images containing the patches and not the entire image greatly reduces the memory footprint as well as reduces the computation complexity. This is a very important advantage of the whole method.

Once the patch is rectified, the features are detected and their descriptors computed. The locations of these features are transformed back by the inverse homography. This is done for each patch one by one. Lastly, keypoints in the regions of the image not belonging to any patch are computed unrectified. Note that if no plane is detected in the image, then the method automatically fallbacks to the standard keypoints computation.

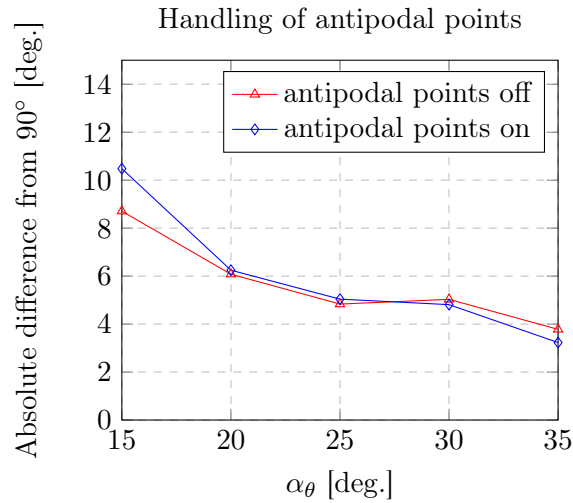


Figure 2.9: Handling of antipodal points. The performance is similar. The feature turned on better here - in accuracy of estimated normals. However it was the other way around in accuracy of estimating the relative pose in two-view matching.

```

def rectify_img(K, normal, img, patch_mask):
    """
    :param K: calibration matrix
    ...
    :return: rectified image, rectified patch mask
    """

    # check the sign
    rot = cross_product([0, 0, 1], normal)
    R = rodriguez_formula(rot)
    H = K @ R @ inverse(K)
    bb_size = bounding_box(H @ patch_mask).size
    scale = math.sqrt(2 * patch_mask.size / bb_size)
    H[2] = H[2] / scale
    (t_x, t_y) = -min(H @ patch_mask)
    T = translation(t_x, t_y)
    H = T @ H
    bb = bounding_box(H @ patch_mask)
    rect_img = warp_perspective(img, H, bb)
    rect_patch = H @ patch_mask
    return rect_img, rect_patch

```

Figure 2.10: Pseudo-code for the rectification via homography

Chapter 3

Rectification via affine maps using AffNet

3.1 Motivation

In the previous Section the rectification via homography was used to simulate transformation of the projected planes to a fronto-parallel view, undoing the effect of perspective projection. As already mentioned, an advantage of the approach taken by Toft et al. [27], as opposed to e.g. methods based on vanishing points, is that only a part of the image needs to be rectified at a time, which results in lower time complexity and memory consumption.

Another possibility examined in various works ([28], [16], [24], [23]) is to use affine transformations for rectification or simply simulation of all possible transformations (up to given granularity). Affine transformations form a subgroup of homographies, they are simpler to model. Image transformation via an affine map results typically in a smaller resulting image when transformed by a homography.

Affine maps can be seen as local approximations to homographies. Indeed, let H be a homography. The homogeneous vector $x \cong (x_1, x_2, 1)$ will be mapped to $Hx \cong (y_1, y_2, 1) = (\frac{(Hx)_1}{(Hx)_3}, \frac{(Hx)_2}{(Hx)_3}, 1)$. Considering the function $\eta : x \rightarrow (y_1, y_2)$, its Taylor expansion gives

$$\eta(x+z) = \eta(x) + L(z) + o(|z|), \quad (3.1)$$

where

$$L = \begin{bmatrix} \frac{h_{11}-y_1h_{31}}{h_{31}x_1+h_{22}x_2+h_{33}} & \frac{h_{12}-y_1h_{32}}{h_{31}x_1+h_{22}x_2+h_{33}} \\ \frac{h_{21}-y_2h_{31}}{h_{31}x_1+h_{22}x_2+h_{33}} & \frac{h_{22}-y_2h_{32}}{h_{31}x_1+h_{22}x_2+h_{33}} \end{bmatrix}, \quad (3.2)$$

which is nothing more than the linear part of the affine approximation.

3.2 Covering the space of tilts

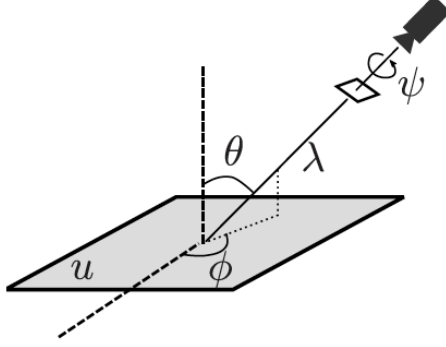


Figure 3.1: Geometric interpretation of a linear map decomposition. First the camera is rotated by ϕ . It is then tilted by $\theta = \arccos \frac{1}{t}$ and rotated by ψ . Scaling by λ commutes with any of those operations. The image taken from [24].

The digital image of a flat object obtained by any camera at infinity is modeled as ([28], [24], [23]):

$$u = S_1 G_1 A T u_0, \quad (3.3)$$

where u is the digital image and u_0 is an (ideal) infinite resolution frontal view of the flat object. T is a translation and A is a linear map with positive determinant. G_1 is a Gaussian convolution modeling the optical blur, and S_1 is the standard sampling operator on a regular grid with mesh 1. As stated in [24], the Gaussian kernel is assumed to be broad enough to ensure no aliasing by the 1-sampling, therefore with a Shannon-Whittaker interpolation I , the continuous image can be recovered from its discrete version: $IS_1 G_1 A T u_0 = G_1 A T u_0$. S_1 will therefore be ignored as well as the translation component as it does not add up to the deformation of the image. The model thus simplifies as:

$$u = G_1 A u_0. \quad (3.4)$$

The optical blur is not easily invertible (see further). The linear map A with strictly positive determinant, which is not a similarity, can be uniquely decomposed ([28]) as:

$$A = \lambda R_1(\psi) T_t R_2(\phi), \quad (3.5)$$

where R_1 and R_2 are rotations and

$$T_t = \begin{pmatrix} t & 0 \\ 0 & 1 \end{pmatrix} \quad (3.6)$$

is a tilting transformation with $t > 1$, $\phi \in [0, \pi)$, $\psi \in [0, 2\pi)$. Similarities are carefully left out from this proposition as their decomposition requires $t = 1$ and is technically not unique (ϕ and ψ

can either both be equal to 0 or to π). This decomposition has also a nice geometric interpretation - see Figure 3.1. It is the tilting component that local features typically cannot handle. SIFT, as a canonical and well performing local feature type, is made robust to the other groups of linear transformation in this decomposition, i.e. to scaling and rotation. In [24] it was defined the space of tilts, whose members are equivalence classes of linear maps with the same last two components of this decomposition (classes of all similarities has identity as their representative):

$$\Omega = [Id] \cup \left\{ \bigcup [T_t R_\phi] \right\}, \phi \in [0, \pi), \psi \in [0, 2\pi) \quad (3.7)$$

Furthermore a metric ρ is defined on this space:

$$\rho([A], [B]) = \log \tau(AB^{-1}), \quad (3.8)$$

where $\tau(A) = t$ for non-similarity $A = \lambda R_1(\psi) T_t R_2(\phi)$ and 1 for similarity. The distance between two classes $[T_{t_1} R(\phi_1)], [T_{t_2} R(\phi_2)]$ can be written as:

$$\rho([T_{t_1} R(\phi_1)], [T_{t_2} R(\phi_2)]) = r \iff G(t_1, t_2, \phi_1, \phi_2) = \frac{e^{2r} + 1}{2e^r}, \quad (3.9)$$

where

$$G(t_1, t_2, \phi_1, \phi_2) = \left(\frac{\frac{t_1}{t_2} + \frac{t_2}{t_1}}{2} \right) \cos^2(\phi_1 - \phi_2) + \left(\frac{\frac{1}{t_1 t_2} + t_1 t_2}{2} \right) \sin^2(\phi_1 - \phi_2). \quad (3.10)$$

According to [24], local features like SIFT can realistically cope with two tilted transformation, if their distance ρ is less than threshold $r = \log 1.7$. While ASIFT[28], an affine invariant version of SIFT, accounts for the variations in tilts and the rotations by ϕ by experimentally covering the space of tilts with the set of values for t and ϕ , [23] shows how to cover the space of tilts with (approximately optimal) covering of r -balls. See Figure 3.2. Covering the space of tilts for all $t < \Lambda$ with a set of simulated tilts account for up to Λ^2 transition tilt, which is the ρ distance between two transformations applied to the original image (e.g. in two different views). For the proof of the previous proposition see [24], theorem 19.

ASIFT, justified by this formalization, allows for this greater variations in transition tilt thanks to all the combinations of simulated tilts applied to the query and train image. On the other hand, methods described in [23] try to cover all or significant part of normalizing transformations in the space of tilts, which are given by shape estimator AffNet used for HardNet features. When covering these transformations with a given r -ball centered in $[T_t R(\phi)]$, the image is transformed with $T_t R(\phi)$ along with the Gaussian blur - see further. Union of features computed after these transformations (their locations unwarped as in the case of rectification via homography) for all these covering r -balls is computed and used for matching. Note that unlike in the case of homography rectifications, the same part of the image will typically be rectified more than once, meaning the method may result in multiple features located at the same point in the image as each of them were obtained with a different simulated transformation. The most progressive and fastest algorithm described in [23], called "greedy selection", greedily finds an r -ball covering most of the data points given by AffNet that remain to be covered, until at least some given fraction (θ) of datapoints are covered. This is formalized in Algorithm 1.

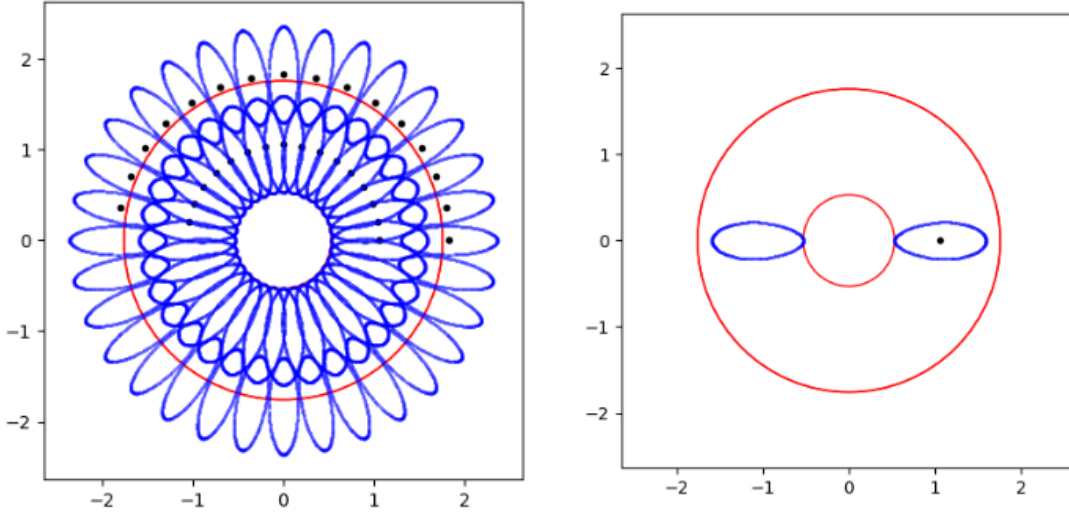


Figure 3.2: Space of tilts and its covering. Left - all sets of the covering. Right - one set of the covering (except of the inner red circle) along with its "antipodal point". The covering covers the area within the outer circle and corresponds to tilts $t \leq 5.8$. Covering sets are in this case 25 $\log 1.7$ -balls shown as blue shapes similar to ellipses, whose centers are depicted as black dots. These ellipses are more elongated the further away they are located from the center. One of the covering shape is actually the inner red circle, which contains the class of similarities, which corresponds to the origin in the space of tilts. The other covering sets displaced from the center also cover an area opposite to them across the origin. This can be seen on the right, where only one covering blue set is depicted - with its center as a black dot. The space is parametrized via radial coordinates, with the radius corresponding to $\log t$ and the angle to ϕ . Shown is an approximately optimal covering called as optimal $\log 1.7$ covering in [24]. Here it is called sparse covering as opposed to dense covering, which has 192 covering sets.

Finally, to address the Gaussian blur in 3.4, it is shown in [24] that the effect of the optical blur can be reversed by applying a Gaussian blur with $\sigma = c \cdot \sqrt{t^2 - 1}$ along the x axis after the rotation by ϕ , i.e. $T_t G_{x, \sigma} R(\phi)$ instead of $T_t R(\phi)$ is applied. c is a parameter advised to be set to 0.8 [28].

Parameter : r - tilt radius (set to 1.7)

Parameter : θ - cover threshold (set to 0.95)

Parameter : \mathcal{S}_G - space of tils covering

Input: \mathcal{A} affine normalizing maps provided by AffNet

$\mathcal{A}_0 = \mathcal{A}, \mathcal{S} = \emptyset$

while $|\mathcal{A}_0| \geq (1 - \theta)|\mathcal{A}|$ **do**

$S = \arg \max_{S \in \mathcal{S}_r} |\{ \rho([S], [A]) \leq \log r \mid A \in \mathcal{A}_0 \}|$

$\mathcal{S}_G = \mathcal{S}_G \cup S$

$\mathcal{A}_0 = \mathcal{A}_0 \setminus \{ A \in \mathcal{A}_0 \mid \rho([A], [S]) \leq \log r \}$

end

Algorithm 1: Greedy covering selection algorithm

3.3 Combining AffNet shape estimators and depth maps

It the greedy selection algorithm shape estimates from AffNet CNN are used and subsequently covered in the space of tilts. AffNet was trained to output a canonical shape (normalizing transformation) for features that undergo random affine transformations. It was developed along with a CNN-based feature descriptor, called HardNet. AffNet effectively serves as a component deciding on the normalizing transformation, which is applied before HardNet computes the feature descriptions, even though AffNet was trained separately – as a CNN simply predicting affine transforms.

It was observed, that HardNet features perform very well on the dataset from [27] and its performance is further significantly improved by the homography rectification - see Figure 4.5. The idea to combine a method like greedy selection algorithm and the method based on information about depth is simple: perform the clustering into normals and detect the planar patches the same way as for the homography rectification. Then, instead of applying the homographies, apply the greedy selection algorithm independently on each patch. Compute only unrectified features on parts of the image not belonging to any detected planar patch. The whole algorithm is described in 3.3. Note that in the first step the HardNet features are computed on the whole input image. This a) finds the unrectified features, which are then used in regions outside of any planar patch as well as in each planar patch b) also provides the shape estimates given by AffNet automatically for each feature, which will then be covered in the space of tilts in each planar patch separately.

```
def aff_net_and_depth(img, depth_map):  
    patches = detect_planar_surfaces(img, depth_map)  
    kpts, desc, shapes = HardNet.detectAndCompute(img)  
    for patch in patches():  
        to_cover = restrict(shapes, patch)  
        kpts_p, desc_p = greedy_cover_selection(patch, to_cover)  
        kpts = kpts.union(kpts_p)  
        desc = desc.union(desc_p)  
    return kpts, desc
```

Figure 3.3: Pseudo-code for the new method using both AffNet shapes and the depth maps.

Chapter 4

Experiments and results

4.1 Dataset

The method was tested on the same dataset that was also used and first presented in Toft et al. [27]. It was created to provide a dataset suitable for evaluating the performance of methods based on single-image depth prediction. It captures typical outdoor scenes from a city. The dataset consist of 8 scenes, each containing images of one building taken from a wide range of different viewpoints. The images will typically differ in other aspects (lighting, weather, etc.). The dataset was created from video sequences (up to 5 sequences per scene, see table 4.1). Colmap ([25], [26]) was used to reconstruct these sequences and then to extract the poses for the images chosen from the sequences and used in the dataset. The dataset contains intrinsic parameters for each image, which are actually estimates also computed by Colmap - this is due to the camera’s autofocus, which may change the effective intrinsic from image to image.

The performance on this dataset is evaluated as the accuracy of relative pose estimation between pairs of images. For each scene, a list of image pairs is provided, which is split into 18 different categories depending on their relative rotations - the pairs in the k -th category (called difficulty) have relative rotation in the range of $[10k, 10(k + 1)]$ degrees. Each difficulty typically consists of up to 200 pairs of images. For some scenes the higher difficulties are missing altogether (see Table 4.1). Within a single scene the images would repeat in different pairs (typically an image would occur in 1-6 pairs within the scene), which allowed for caching of image processing data, which sped up the experiments.

Scene	1	2	3	4	5	6	7	8
difficulties	17	17	17	14	17	16	16	17
pairs	3590	3600	3600	2612	3428	3031	2893	3312
sequences	4	4	3	4	3	5	3	3

Table 4.1: Strong ViewPoint Changes Dataset statistics

4.2 Experiments setting

The performance of the method was measured as the accuracy of the estimated relative camera rotation between the image pair. This was done by first computing the tentative corespondences (performing cross check and using the Lowe’s ratio of 0.85) and then running RANSAC given the tentative correspondences and the calibration matrices $\mathbf{K}_{1/2}$ for both images. Parameter values for RANSAC were set as follows: threshold = 0.5 pixels, confidence = $1 - 10^{-4}$, iterations = 10^5 .

After computing essential matrix using OpenCV RANSAC (flag `cv2.RANSAC` used) the pose estimation was considered accurate, if the difference between the estimated relative rotation and the ground truth relative rotation was smaller than given threshold (usually 5°). For each experiment the ratio of accurately estimated image pairs was measured (typically across the whole dataset or one scene). This is exactly the same performance task and metric as used in [27]. Therefore the results obtained here and there can be compared, however the exact numerical values obtained from there were not presented, only approximate values from graphs ¹.

4.3 Implementation

The image processing before the rectification (normal estimates, normals clustering, connected components) was mainly implemented using the PyTorch library [19], which allowed for GPU acceleration through CUDA. However, SVD, an important part of the pipeline, is known to be extremely slow in PyTorch for CUDA². Therefore the data are copied back and forth between CPUs and GPUs around the operations related to SVD. This obviously affects the performance of these steps as well as the overall performance. It is one of the reasons not to compare or observe the performance solely based on the elapsed times (see the discussion of the performance of various approaches for covering the space of tilts under the section with results).

The sky filtering was done using a semantic segmentation project by CSAILVision [29]. The rectification via homography and matching were implemented using OpenCV [8] as it provides many useful components out of the box (BRISK and SIFT features, matching API - matchers, RANSAC, etc.). Kornia [22] was used for HardNet features and the related AffNet shape estimator. For SuperPoint features code forked from the original SuperPoint repository [6] [2] was used. Adapters for these non-OpenCV features were written to comply with the OpenCV API. The rectification via affine maps was implemented using Kornia. For the RANSAC homography and essential matrix estimation, we have used OpenCV MAGSAC++ [4] algorithm.

The computation times of various steps, even though measured quite precisely, vary greatly depending on the exact parameters, hardware and whether the computation was done on GPUs or CPUs. If run on GPUs, the bottlenecks are clearly all the steps for were not implemented for CUDA, most notably the rectification via homography and computation of normals via SVD. The rectification via homography takes around 1 second per rectified patch, the computation of normals takes under 0.5 seconds. In comparison, the normals clustering take about 0.05 seconds and finding the connected components about 0.1 seconds. It is clear that in terms of performance there is a big room for improvements in the current implementation.

Part of the code base is attached to the thesis as an electronic attachment, but as the size is limited to 50MB in size, the submitted version is not complete (e.g. not containing minimal playground data or demo jupyter notebooks). The full version is published under https://github.com/vicsyl/extreme_two_view_matching.

¹We have contacted the authors of [27], but they responded that the data was lost

²See <https://github.com/pytorch/pytorch/issues/41306>

4.4 Results

As already noted the performance for the matching task on the dataset is measured as the ratio of image pairs for which the error of estimated relative rotation compared to the ground truth was less than 5° . If not stated otherwise, the default image processing settings were used: SVD weighting enabled, sky filtering enabled, handling of antipodal points disabled, $\alpha_\theta = 35^\circ$, cluster refinement by taking mean and filtering based on singular value ratio disabled.

4.4.1 Tuning the processing parameters

First those results are presented, which show the effect of various improvements during the image processing before rectification. These experiments measured performance of matching using rectification via homography with SIFT features.

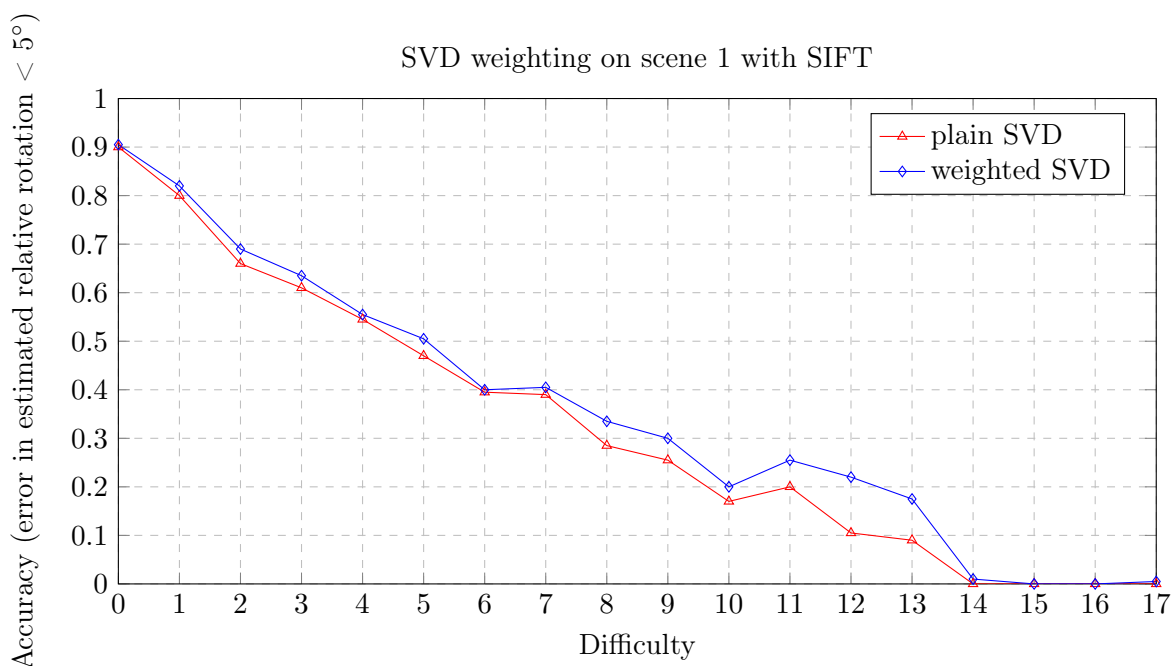


Figure 4.1: The effect of SVD weighting on scene 1. SVD weighting improves the performance in matching consistently throughout scene 1. The data was measured for homography rectification using SIFT features for both cases.

Figure 4.1 shows the improvement in performance when the SVD weighting is turned on. The accuracy was improved consistently on all difficulties throughout scene 1 of the dataset, except on those where it was 0 both with and without the SVD weighting and was improved significantly for difficulties 8-13.

Another parameter to observe is whether to handle the antipodal points together. Figure 4.2 shows that antipodal points can be handled together without significant degradation in performance, however without this feature the matching performance was slightly better across the whole scene 1. Even though the occurrence of e.g. opposing walls should be plausible in real world scene, here the option was disabled in other experiments.

Different morphological operations were applied before the connected components were found in order to make the resulting patches bigger and more compact (ideally simply connected). Figure 4.3

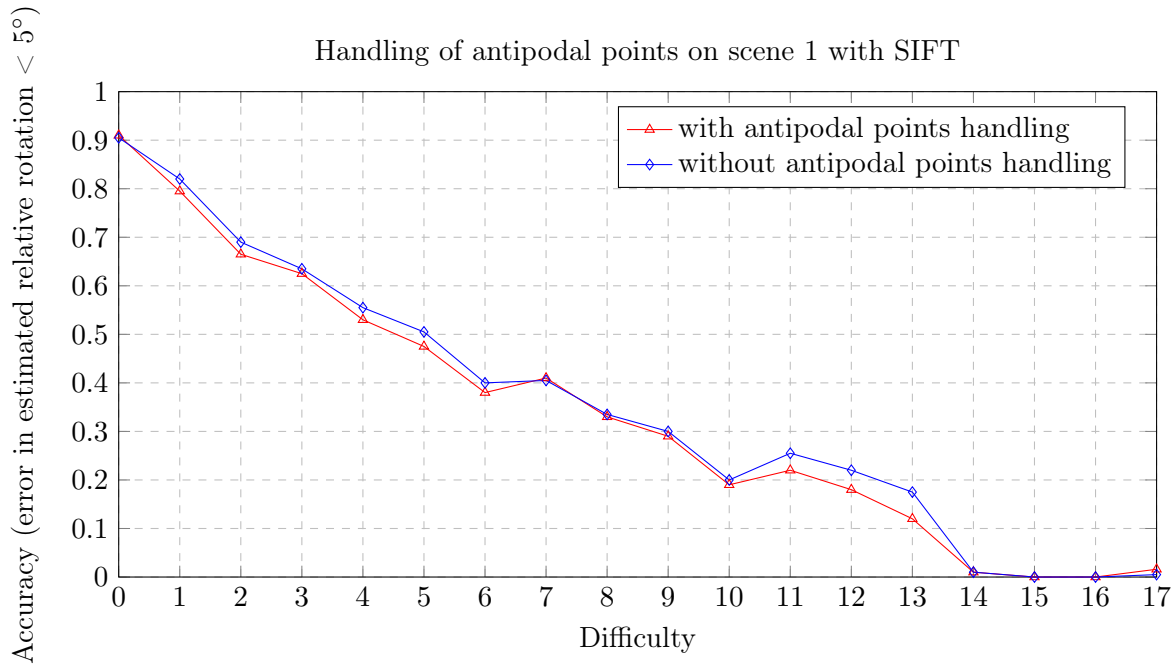


Figure 4.2: Handling of antipodal points on scene 1. The data was measured for homography rectification using SIFT features for both cases.

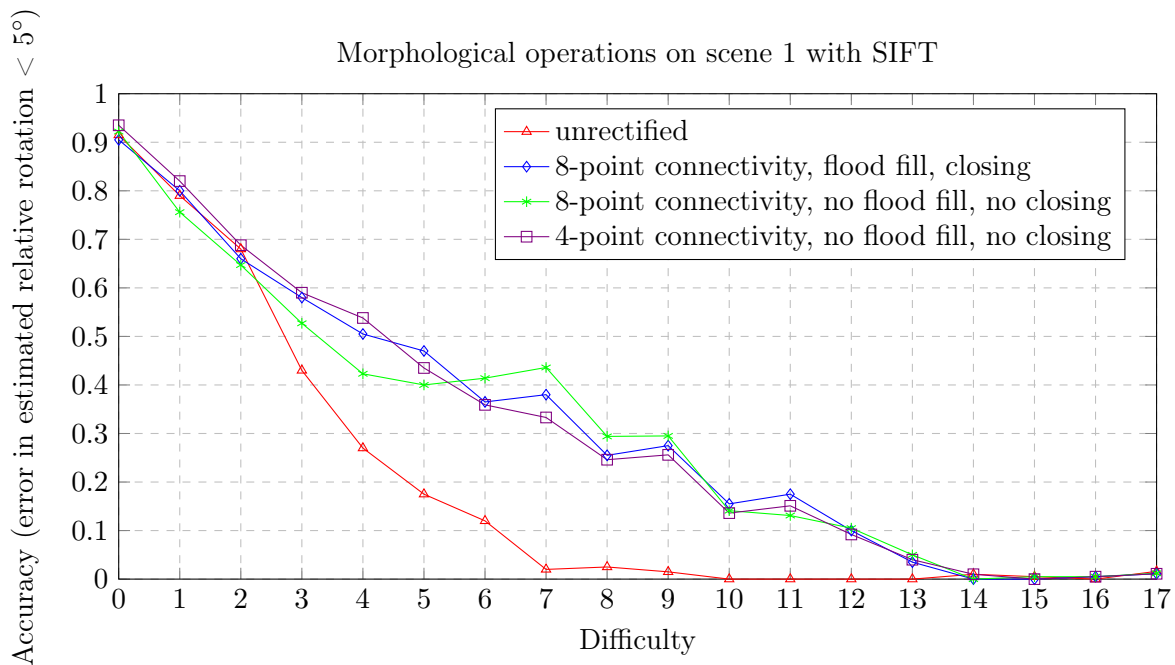


Figure 4.3: Morphological variations - 4/8-p con. denotes 4/8-point connectivity respectively

shows the different morphological operations did not improve the overall accuracy throughout scene 1 and therefore they were not used in all other experiments.

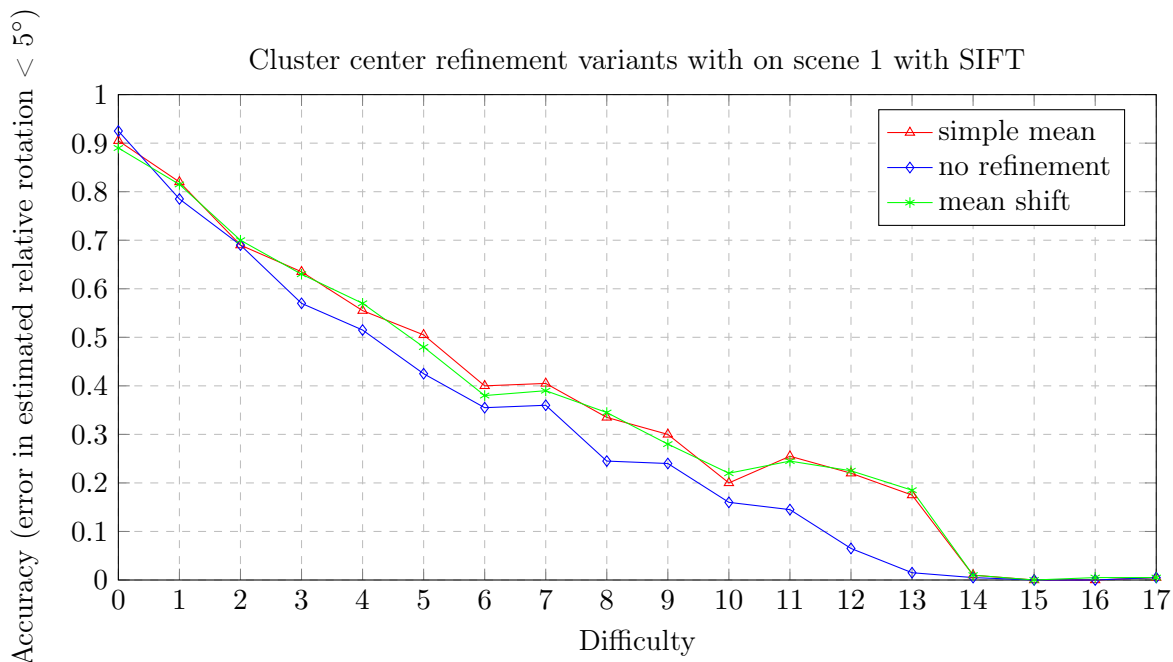


Figure 4.4: Cluster refinement variants. Simple mean and mean shift are almost identical in performance. Simple mean is the preferred option for its speed and simplicity.

4.4.2 Different features

Performance of different features was measured on scene 1 both rectified and unrectified. Figures 4.5 and 4.6 present the performance of SIFT, RootSIFT, HardNet, SuperPoint and BRISK features, each both rectified and unrectified. For all feature types the rectification significantly improved the performance. Rectified variant of HardNet performed significantly better than all others. Rectified version of SuperPoint features also performed better than the unrectified one – note that this is a different result than showed in Toft et al.[27], where the performance of rectified and unrectified variant was almost identical for SuperPoint.

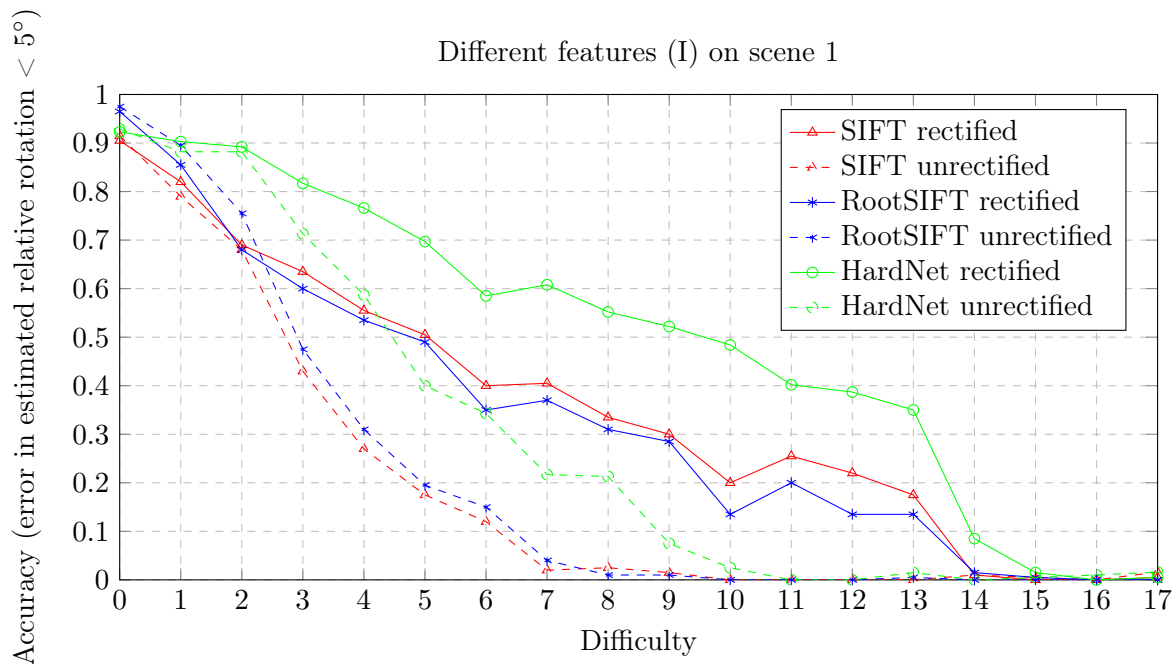


Figure 4.5: Different features (I) on scene 1. The graph shows accuracy for SIFT, RootSIFT and DoG-HardNet, each both rectified and unrectified. HardNet performs significantly better than any other feature type, both rectified and unrectified. All rectified variants are better than the unrectified.

4.4.3 HardNet features and AffNet shape estimator

HardNet features work very well in combination with the rectification via homography. They perform even better, when rectified using a combination of information about depth and information from its shape estimator AffNet as described in Chapter 3.

Figure 4.7 shows accuracy of the best performing setting for HardNet features rectified via affine maps - which is sparse covering with $\theta = 0.95$. This setting is compared to the performance of HardNet and SIFT features rectified via homography. HardNet features rectified via affine maps outperforms the other two versions in all difficulties except difficulty 0 and 1, where it is slightly worse than HardNet features rectified via homography.

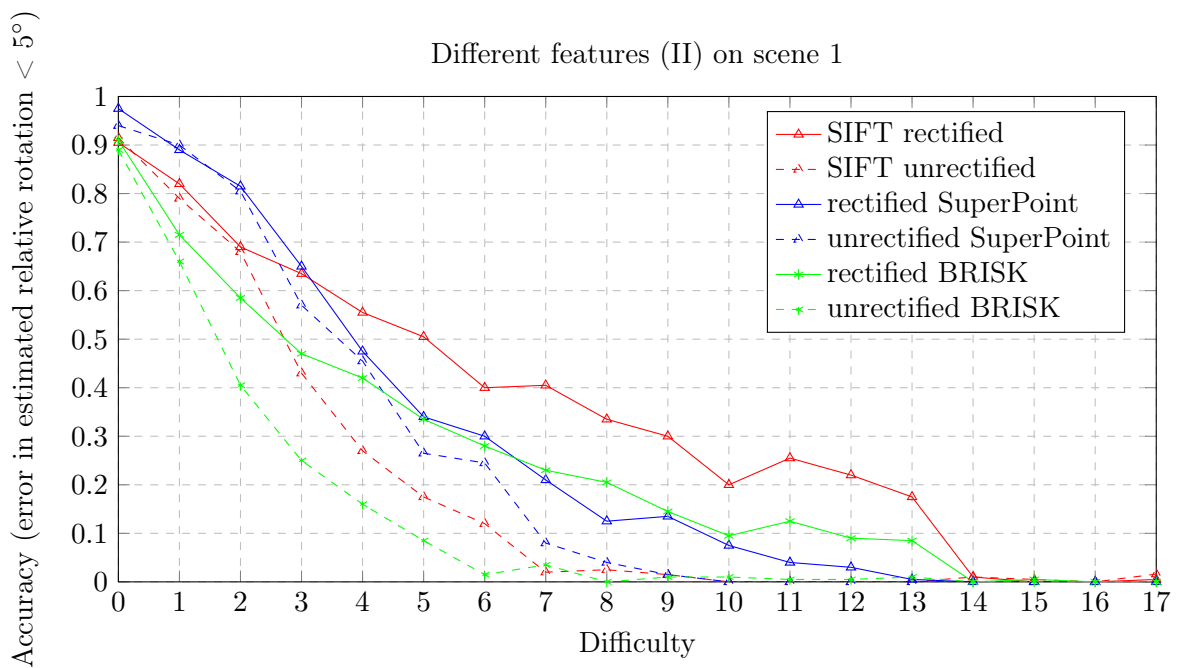


Figure 4.6: Different features (II) on scene 1. The graph shows accuracy for SIFT, SuperPoint and BRISK, each both rectified and unrectified. All rectified variants are better than the unrectified.

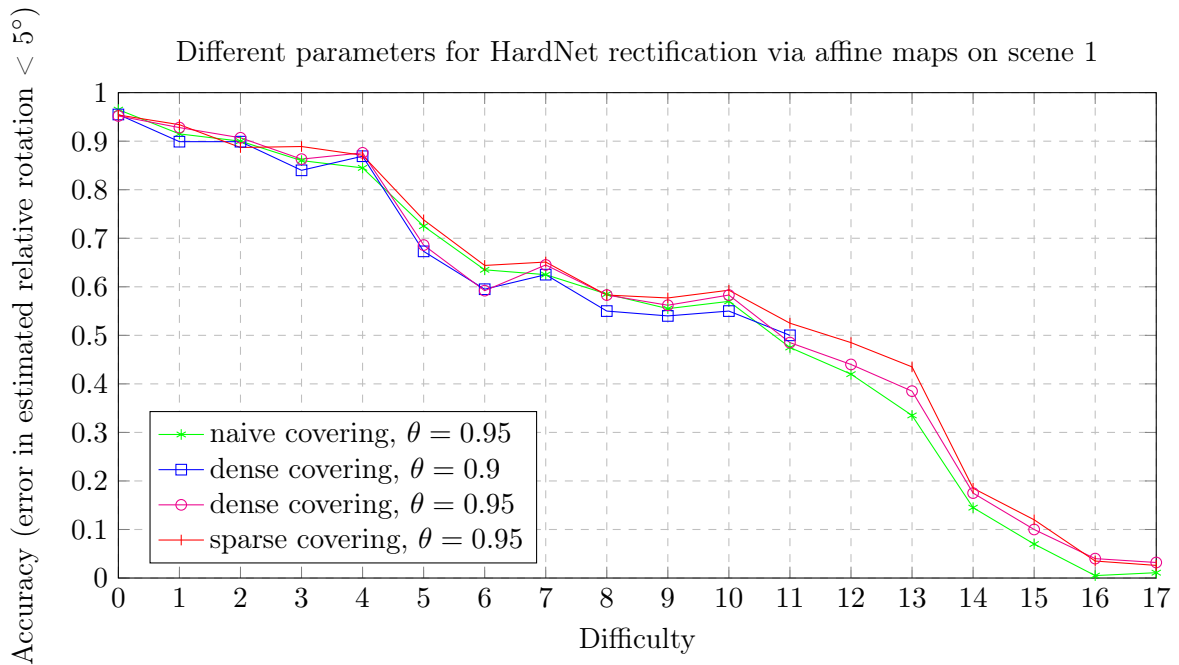


Figure 4.8: Rectification of HardNet features via affine maps with different parameters. The performance of all of these is very similar. Nevertheless, the variant with sparse covering with $\theta = 0.95$ performs slightly, yet consistently better than the rest.

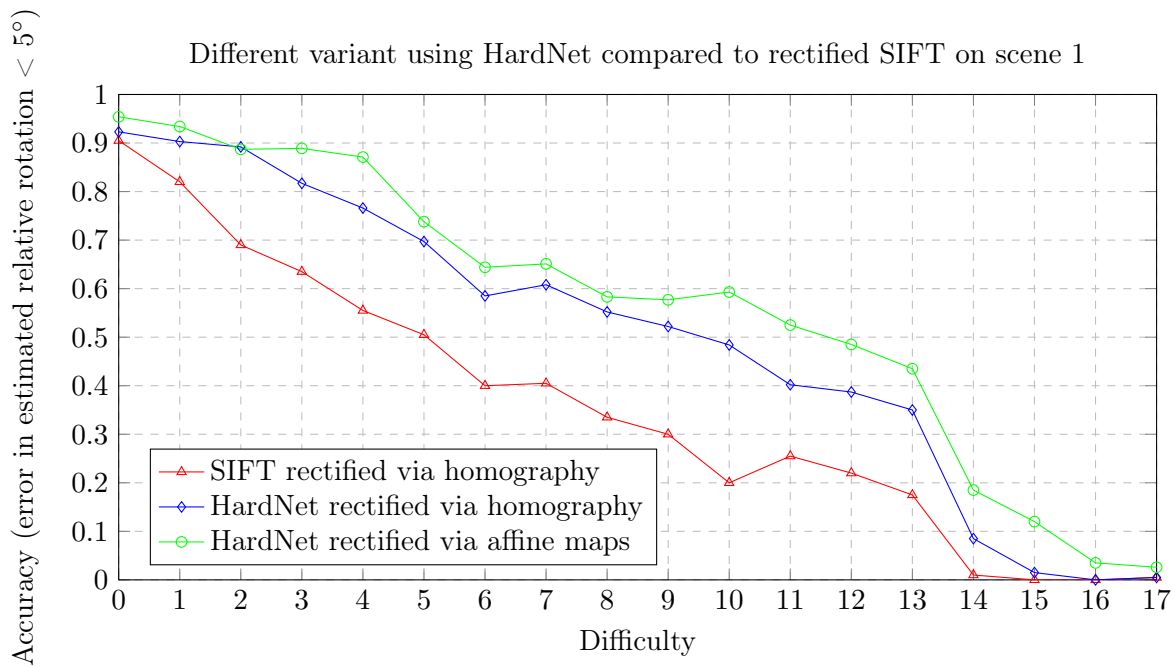


Figure 4.7: HardNet rectified via affine maps and homography compared to SIFT rectified by homography

Figure 4.8 shows accuracy of HardNet features rectified via affine maps with different parameter values. Different coverings (dense, sparse, naive - see Chapter 3 for their descriptions). Realistic values for θ to try are 0.9 and 0.95. Sparse covering with $\theta = 0.95$ is slightly better than the rest in all difficulty categories.

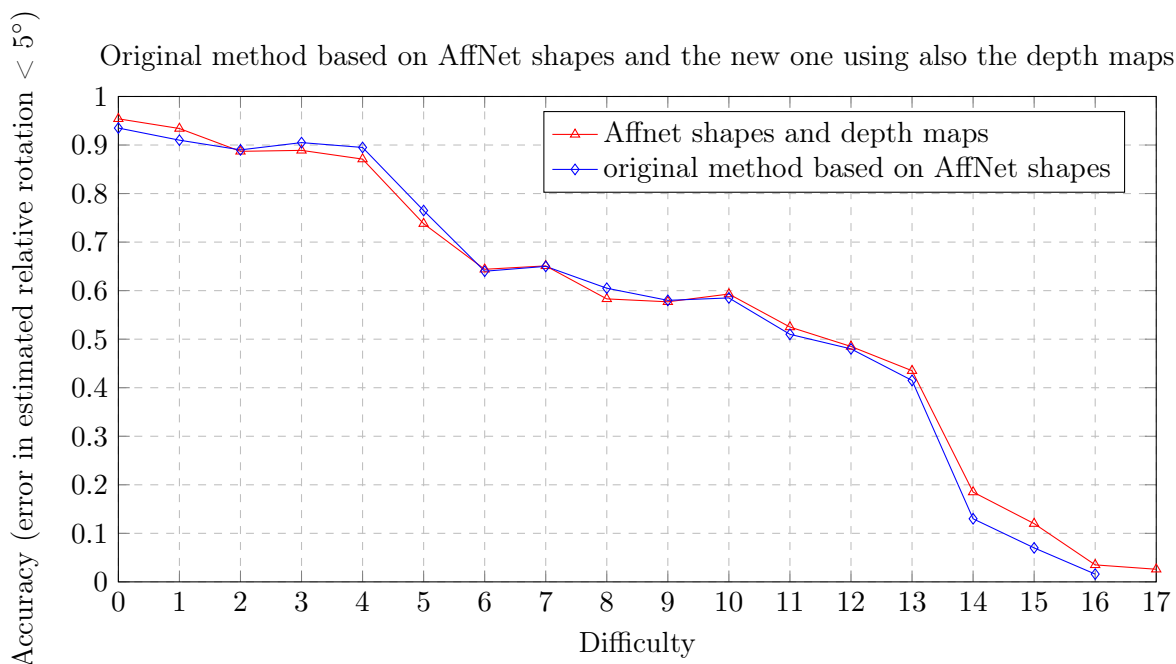


Figure 4.9: Original method based solely on AffNet shapes estimates and the new method based on AffNet shapes and depth maps. The new method almost exactly matches the performance of the original method.

Figure 4.9 shows accuracy of HardNet features rectified via affine maps as described in 3 and with the original method as described in [23]. Both versions use the same parameters, namely sparse covering and $\theta = 0.95$. The new method use the same method of rectification as the original one, but it also takes advantage of the information from depth map and it applies the rectification only on detected planes, for which it covers the space of tilts one by one. It almost exactly matches the performance of the original method, but it’s much faster. Figure 4.10 shows the relative ratio of various metrics related to computational complexity. The average number of warps, already observed in [23], is almost the same in both method. The average warped are per component is however much smaller with the new method. Here component means a planar surface corresponding patch for the new method and the entire image for the original one (as there is no partition of the image). As the time dominating operation of the whole method is the warping and unwarping during the rectification, the overall running time is also shorter by the new method - by the factor of 2.52.

Figure 4.11 shows the results of unrectified SIFT, our re-implementation of Toft et al. [27] with out proposed improvement to their pipeline and finally the our proposed hybrid algorithm with using AffNet data for affine-based rectification and depth-based segmentation together with HardNet descriptor. Scene 1 was excluded from the evaluation to avoid testing on the training set.

Computation complexity of the original and new method using AffNet shape estimates

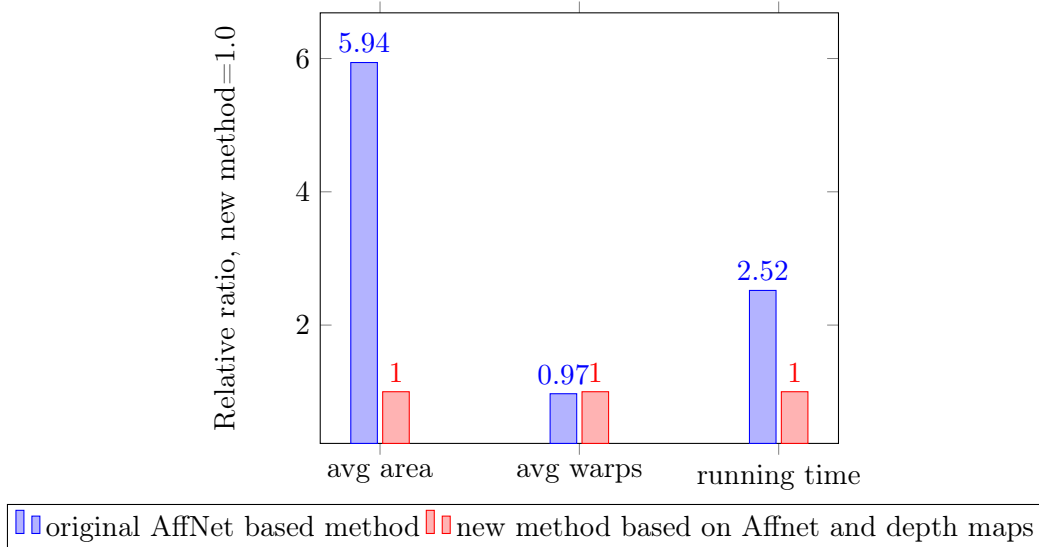


Figure 4.10: Computation complexity of the original method using AffNet shapes and the new method using AffNet shapes and the depth maps. The new method is 2.52 times faster.

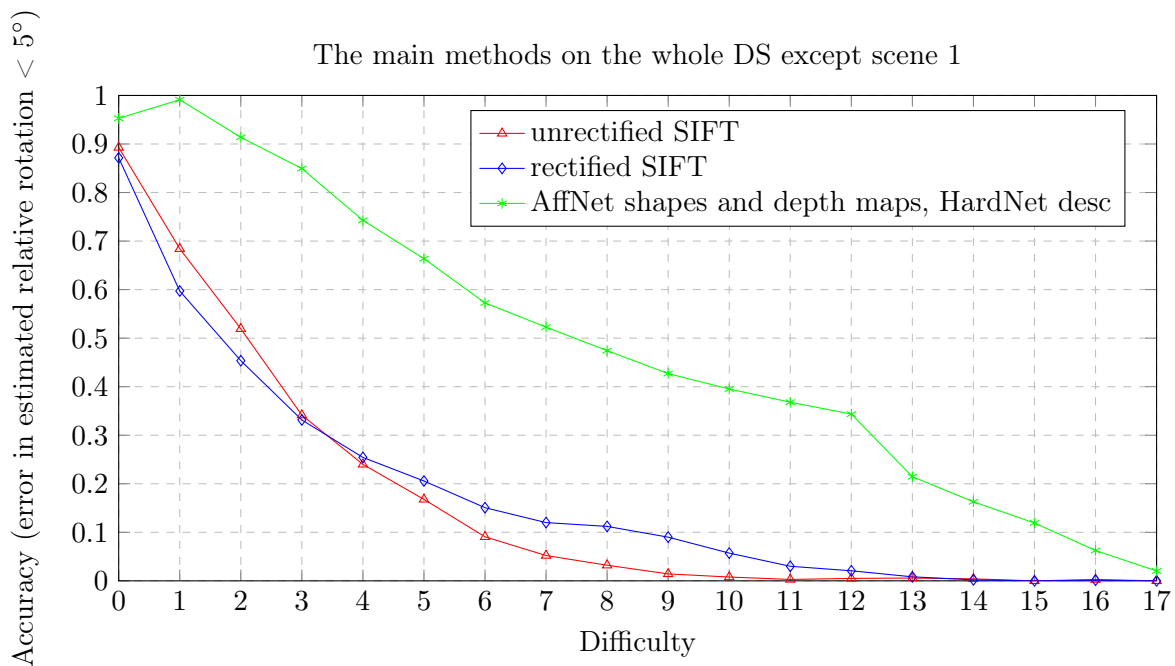


Figure 4.11: Main methods on the whole dataset except scene 1. In this experiment the rectified SIFT features did not perform as well as in other experiments, even though it still has better overall accuracy than the unrectified variant. The method using AffNet shape estimators and depth maps with HardNet performs much better.

4.5 EVD dataset

Finally the performance of the various methods on the EDV dataset is presented. The EDV dataset contain just 15 challenging image pairs with a single dominating plane. The task is homography estimation. The mean absolute reprojection error is computed for each pair given the estimated and ground truth homography. Outliers are frequently encountered therefore the observed metric as the accuracy given difference accuracy thresholds, specifically $\{1, 2, 3, 5, 10, 20\}$.

MAE th.[px]	SIFT unrect.	SIFT rect.	HardNet rect.	AffNet + depth	AffNet covering [24]
20	0	0	2	9	10
10	0	0	2	7	9
5	0	0	1	4	8
3	0	0	0	2	5
2	0	0	0	1	2
1	0	0	0	0	0

Table 4.2: Results on Extreme View Dataset [16]. Explain

Chapter 5

Conclusion

The thesis presents improvement to a method originally described in Toft et al [27] by revisiting the various implementation details and tuning the parameters. These improvements are tested in detail in various experiments described in chapter 2. New method was then presented in chapter 3 as a combination of the approach presented in Rodríguez et. al [24] and exploitation of the information from CNN provided depth maps. Both improvements are then tested on the task of two view matching in various experiments, which are presented in the fourth chapter. Especially all approaches based on HardNet excel in performance measured as the accuracy of estimating relative pose between image pairs. The best of these are the two methods using the AffNet shape estimates – the one originally described by Rodríguez et. al [24] and the new one using also the information about depth, which is even more focused and is also much faster.

Our method is still has limitations, mostly because its key component is monocular depth estimator, which fails for the paintings, graffiti, and similar structures. It also assumes existence at least of the one plane in the image, which is not always the case. Finally, it is not always outperforms more brute-force approach by on Rodríguez et. al [24], which could be seen on the example of the EVD dataset (even though it still performed very well).

Throughout the work on the thesis I have tried countless experiments, vast majority of which did not work nearly as much as I hoped for. Some of them may have failed because the method tested was simply inferior in its design, by some of them I may have missed an advantageous parameter setting or an important implementation detail. In any case it has been a great experience, which hopefully taught me something about the research work.

Chapter 6

Bibliography

- [1] Spiraling algorithm. <https://web.archive.org/web/20120107030109/http://cgafaq.info/wiki/Evenly_distributed_points_on_sphere#Spirals>.
- [2] Superpoint github repository. <<https://github.com/magicleap/SuperPointPretrainedNetwork>>.
- [3] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Rick Szeliski. Building rome in a day. *Communications of the ACM*, 54:105–112, 2011.
- [4] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiri Matas. Magsac++, a fast, reliable and accurate robust estimator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [5] M. Brown and D. Lowe. Automatic Panoramic Image Stitching Using Invariant Features. *IJCV*, 74:59–73, 2007.
- [6] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR Deep Learning for Visual SLAM Workshop*, 2018.
- [7] Clément Godard, Oisín Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth prediction. October 2019.
- [8] Itseez. Open source computer vision library. <https://github.com/itseez/opencv>, 2015.
- [9] Kevin Köser and Reinhard Koch. Perspectively invariant normal features. In *International Conference on Computer Vision*, pages 14–21, 2007.
- [10] Stefan Leutenegger, Margarita Chli, and Roland Siegwart. Brisk: Binary robust invariant scalable keypoints. pages 2548–2555, 11 2011.
- [11] Haoang Li, Ji Zhao, Jean-Charles Bazin, Wen Chen, Zhe Liu, and Yunhui Liu. Quasi-globally optimal and efficient vanishing point estimation in manhattan world. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1646–1654, 2019.
- [12] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [13] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

- [14] Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor’s margins: Local descriptor learning loss. *CoRR*, abs/1705.10872, 2017.
- [15] Dmytro Mishkin. Learning and crafting for the wide multiple baseline stereo, 2021.
- [16] Dmytro Mishkin, Jiri Matas, and Michal Perdoch. MODS: fast and robust method for two-view matching. *CoRR*, abs/1503.02619, 2015.
- [17] Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Learning discriminative affine regions via discriminability. *CoRR*, abs/1711.06704, 2017.
- [18] R. Mur-Artal, J. Montiel, and J. Tardós. ORB-Slam: A Versatile and Accurate Monocular Slam System. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.
- [19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
- [20] James Pritts, Zuzana Kukelova, Viktor Larsson, and Ondrej Chum. Rectification from radially-distorted scales. *CoRR*, abs/1807.06110, 2018.
- [21] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.
- [22] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Winter Conference on Applications of Computer Vision*, 2020.
- [23] M. Rodríguez, G. Facciolo, R. Grompone von Gioi, P. Musé, J. Delon, and J.-M. Morel. Cnn-assisted coverings in the space of tilts: Best affine invariant performances with the speed of cnns. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2201–2205, 2020.
- [24] Mariano Rodríguez, Julie Delon, and Jean-Michel Morel. Covering the space of tilts. application to affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 11, 01 2018.
- [25] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [26] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.
- [27] Carl Toft, Daniyar Turmukhambetov, Torsten Sattler, Fredrik Kahl, and Gabriel J. Brostow. Single-image depth prediction makes feature matching easier. In *European Conference on Computer Vision (ECCV)*, 2020.
- [28] Guoshen Yu and Jean-Michel Morel. ASIFT: An Algorithm for Fully Affine Invariant Comparison. *Image Processing On Line*, 1:11–38, 2011. <<https://doi.org/10.5201/ipol.2011.my-asift>>.

-
- [29] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal on Computer Vision*, 2018.