UNIVERSITY OF COPENHAGEN

COMPUTER SCIENCE

ADVANCED TOPICS IN IMAGE ANALYSIS

Exam project

Author:
Exam number: 13

Teachers:
Søren OLSEN
Sune DARKNER

November 5, 2021

1 Assignment 1

1.1 Baseline

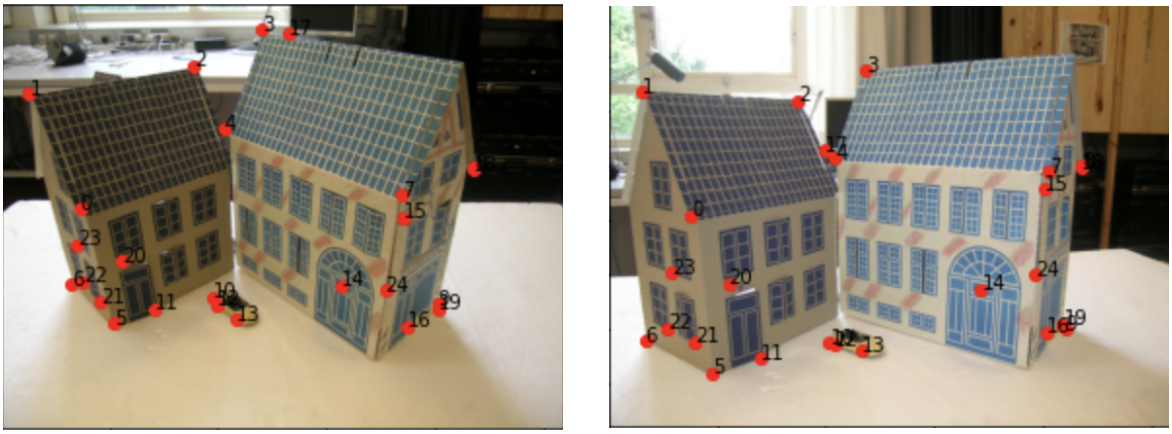


Figure 1: The 25 point correspondences between the two images found.

We will begin by establishing a baseline for comparison with the following experiments. This baseline will be the most naive / simple solution to estimating the fundamental matrix and bring the two images to scanline agreement. We begin by manually estimating 25 point correspondences between the two images which can be seen in Figure 1. We now normalize the points found in each image separately by subtracting the mean of the points and dividing with their standard deviation. This gives us T_A and T_B which are transformation matrices for image A (left) and image B (right), and we will use these to denormalize the fundamental matrix. We can now estimate the fundamental matrix using the 8-point algorithm and after normalizing it by dividing through with index (3,3), we can report it to be:

$$F = \begin{bmatrix} -2.35e-06 & -1.81e-06 & -1.50e-02 \\ 2.77e-06 & 2.43e-06 & -2.05e-02 \\ 1.99e-02 & 1.63e-02 & 1 \end{bmatrix}$$

With the fundamental matrix we can now use *opencv* and the function *computeCorrespondEpilines* to determine the epipolar lines and draw them on the images by supplying the function with the points found in the images and the estimated fundamental matrix. This is done in Figure 2. Using opencv is simply for convenience to transform the points and estimating and drawing the epipolar lines.

By computing the left and right null vectors of the fundamental matrix, F, we can get the coordinates of the epipoles. We find the left epipole, i.e. the optical center of the right camera projected into the left cameras view, to be at coordinates $[-104610.58, 127733.71]$, and the right epipole, i.e. the optical center of the left camera projected into the right cameras view, to be $[4577.86, -3312.85]$.
We can now bring the two images to scanline agreement using *opencv* which can be seen in Figure 3. As the 'scanline agreement error' is ambiguous to me, I have chosen to follow the 'rectification error' from [1] which measures the error by computing the mean and standard deviation of the *y-disparity* (the vertical distance) between the points in image A and image B after rectification. Here we find a **mean** of **8.53** and a **standard deviation** of **7.41** which tells us our epipolar lines on average are fairly close to go through our points in the images, but the variance is also fairly high, which implies some of our points are very close,

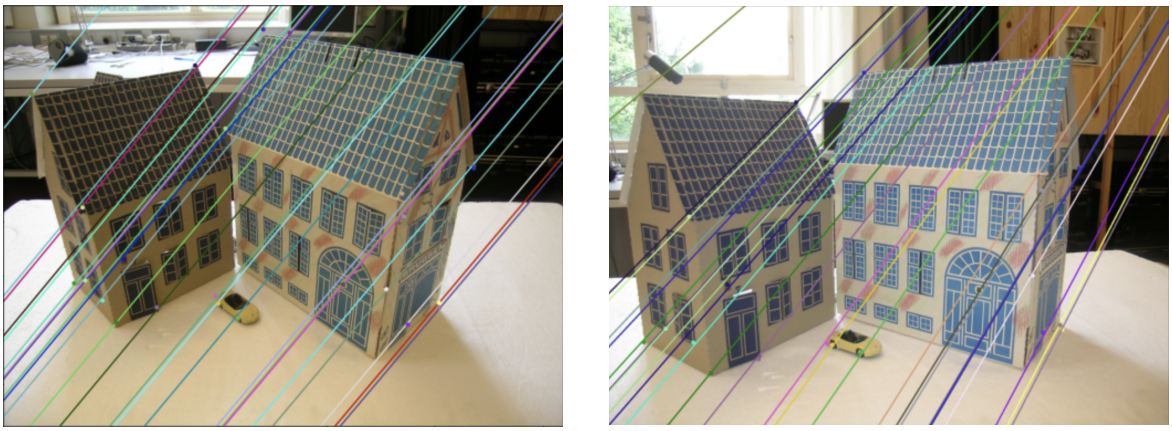


Figure 2: Epipolar lines drawn on top of the two images.

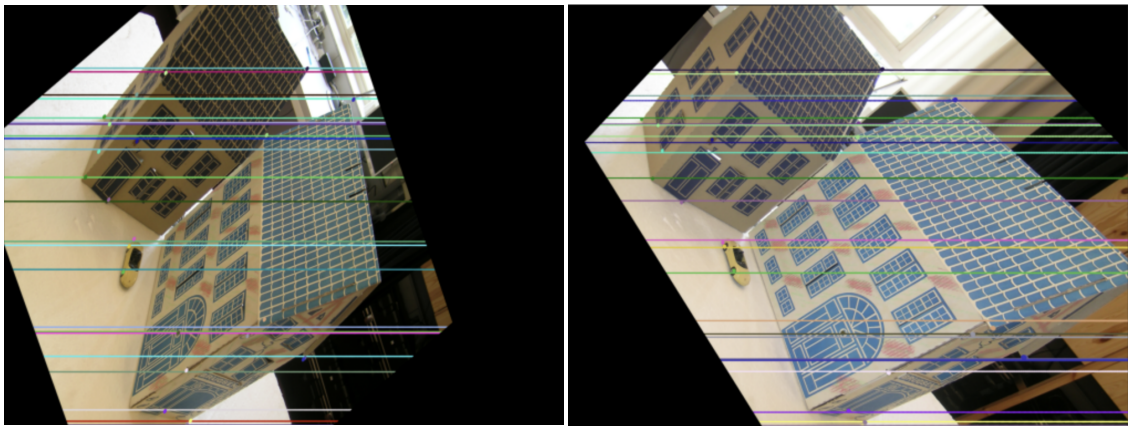


Figure 3: The baseline estimation brought to scanline agreement.

and some are quite far off.

1.2 Ransac with baseline

Ransac is a procedure which can be used to detect outlying data. We will use Ransac to sort out the correspondences we have found which are too imprecise to contribute to the estimation of the fundamental matrix. Using Ransac makes our estimation of the fundamental matrix more robust, and so we will also look at how robust our estimation becomes by adding random correspondences and then attempt to remove false correspondences with Ransac.

1.2.1 Problems with Ransac

Ransac can be implemented in a few different ways. One could simply use the largest inlier set found and return this. One could also estimate the error, and although a bigger inlier set might be found, it will only be accepted if the error is smaller than the previous best set (the set with the smallest error and largest number of inliers). Then, should the points used to compute the model be included when computing this error, or only points not used in the model estimation? In the latter we avoid bias, but the points used for model estimation will most likely be quantified as inliers. And perhaps most importantly, how does one estimate the thresholding used to determine inliers from outliers?

After some experimentation I have found using the largest inlier set gave the most consistent results and have thus chosen to use this implementation. For thresholding a constant threshold of 20 have been used.

As the threshold is often simply chosen empirically, and the scanline agreement error from the baseline shows decent results, a threshold of 20 seems reasonable. The threshold could also have been estimated based on the standard deviation and a Chi distribution. To determine whether a correspondence is within this threshold, the *Sampson distance* have been used as this seems to the most natural way to estimate the error, and it only involves the fundamental matrix.

1.2.2 Error as function of falsely added correspondences

We will now take a look at how robust the fundamental matrix estimation is when using Ransac. For this we take our 25 manually found correspondences and add a number of false correspondence before running Ransac. We then measure the mean and standard deviation of the scanline agreement error. As Ransac is a non-deterministic algorithm we might get rather different errors each run, and because we only are interested in the tendency of how the error behaves when adding false correspondences, we make 10 runs of Ransac and report the median mean and standard deviation found. This helps us avoid lucky or unlucky runs where the error reported is 'uncharacteristic' for the number of false correspondences added, and to avoid the skewness the average would give if one run reports something completely different than the other runs. The results can be seen in Figure 4.

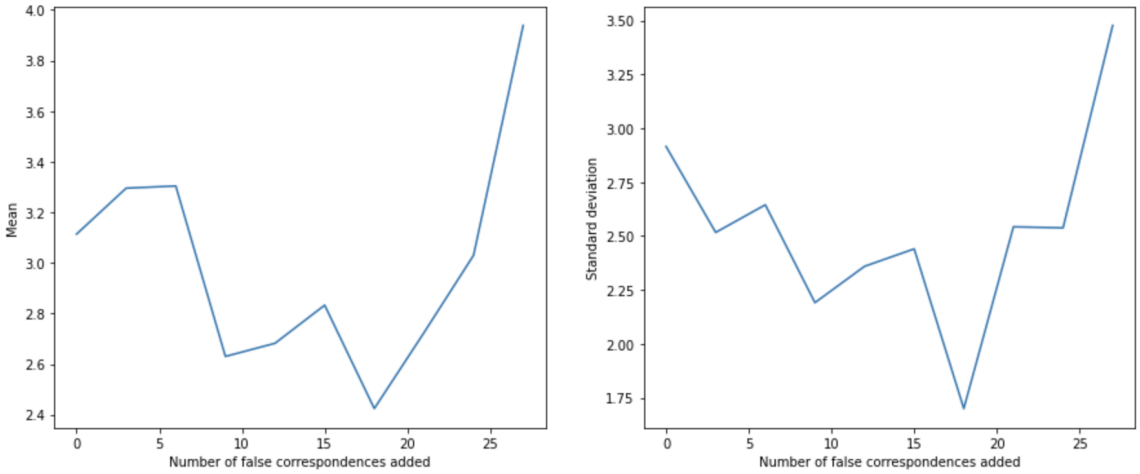


Figure 4: The median mean and standard deviation after 10 Ransac runs with increasing number of false correspondences added to the manually found correspondences.

As seen the median mean and standard deviation only changes slightly, however, both show a tendency to grow quite rapidly when adding more than 18 false correspondences. This would equate to when the data consists of more than 42% outliers Ransac begins to break down. This shows using Ransac makes the estimation of the fundamental matrix *very* robust. We do however need to keep in mind that the false correspondences added here are random points in the two images, and so they with high probability fall *completely* out of line with our inliers. This might be a reason why we can include so many false correspondences.

1.2.3 Scanline agreement using Ransac

Taking a visual look at using Ransac, we can report the result of running Ransac on the 25 correspondences manually found, and from the inliers, estimate the fundamental matrix. This can be seen in Figure 5. We here

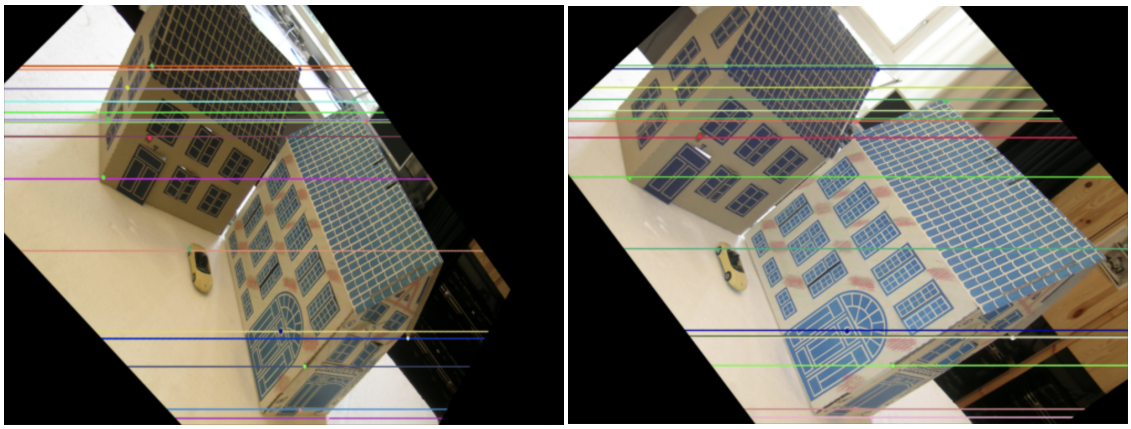


Figure 5: Result of using Ransac on the original 25 correspondences before estimating the fundamental matrix and then bringing the images to scanline agreement.

find a **mean** of **4.14** and a **standard deviation** of **2.57** of the scanline agreement error, which is a significant improvement over not using Ransac.

1.3 Improved correspondences

We will now improve our precision of our correspondences by applying the Harris corner detector. This has been done by extracting a small patch around our initial point location, and then run the Harris corner detector to find a more 'robust'/'easier to find again' location. Once these points have then been found, we can estimate the fundamental matrix and scanline error to see if these improved points have an effect on the fundamental matrix estimation.

1.3.1 Problems with Harris corner detection

It is not obvious what the parameters for the Harris corner detector should be to give optimal results. Because a vast majority of the work in finding the improved points already was manual, I have tweaked the parameters for each original point until the Harris corner detector found a point which I would reckon would be easy to also find in the other image after applying the corner detector.
Another approach would have been to keep the parameters of the Harris corner detector constant and assume it is consistent. That is, if it does not find a perfect location, it will not find it in the other image either, but still find the same point in both image. In other words, if it makes a mistake, it make the same mistake in the other image. This would then have allowed us to discuss how well automatic detection could perform, whereas the first (and used) approach instead finds a 'lower bound' on the error we can get.

1.3.2 Scanline agreement error

If we simply estimate the fundamental with the improved correspondences we now find a **mean** of **1.92** and a **standard deviation** of **1.40** of the scanline agreement error. This is a fair bit better than running Ransac on the original correspondences, but as mentioned earlier, this is not representative for how an automatic detection would perform. We can now run Ransac on the improved correspondences and see the result in Figure 6.

We here find a **mean** of **1.52** and a **standard deviation** of **1.40** of the scanline agreement error, which is a slight improvement.

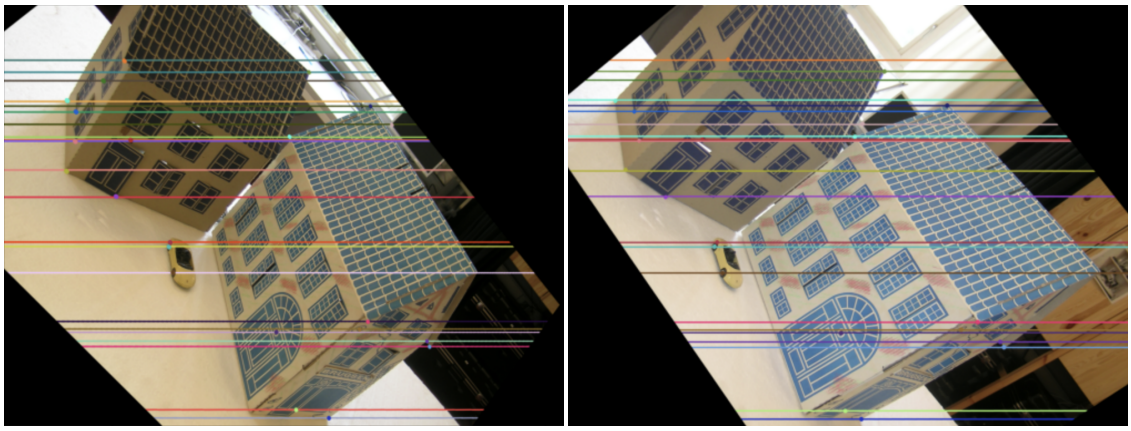


Figure 6: Result of using Ransac on the improved 25 correspondences before estimating the fundamental matrix and then bringing the images to scanline agreement.

1.3.3 Error as function of falsely added correspondences

Looking at the error as a function of falsely added correspondences, now using the improved correspondences we get the results seen in Figure 7.

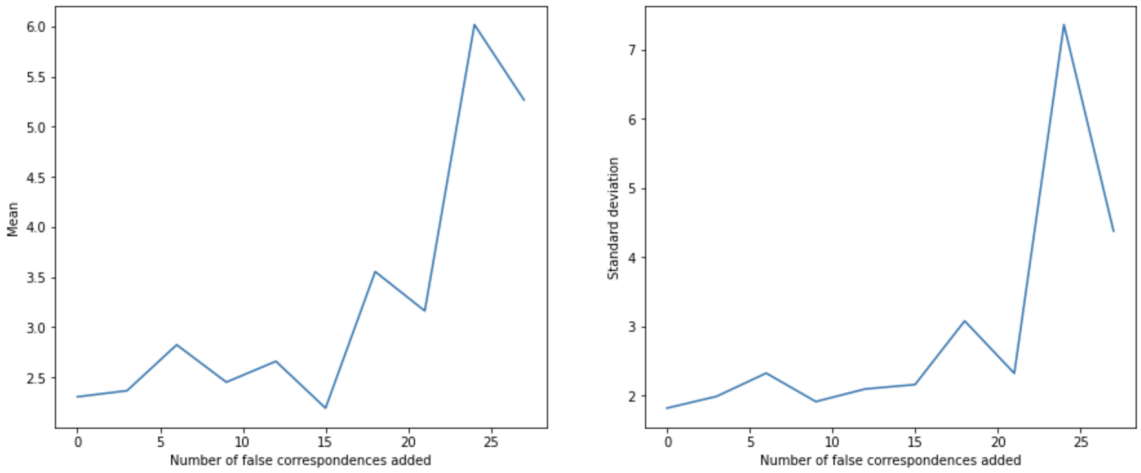


Figure 7: The median mean and standard deviation after 10 Ransac runs with increasing number of false correspondences added to the improved correspondences.

We here see similar results as with the original correspondences, except a more clear tendency of increased error with increased number of false correspondences added. The more clear tendency might be a coincidence based on more similar Ransac runs or due to the run with 24 added false correspondences having a high error, and making it harder to see the small differences in mean and standard deviation.

1.4 Conclusion

In conclusion we have seen the importance of precise correspondences when we want to estimate the fundamental matrix, and we have seen using Ransac make the estimation very robust to random noise. An interesting experiment could have been to add the original correspondences as noise to the improved correspondences to see if this would result in more or less error than using random correspondences. The idea would be it would be harder for Ransac to differentiate between inliers and outliers if the correspondences are not purely random. Probably some of the original correspondences would be classified as inliers and

would then most likely make the estimation worse.

We end by summarizing the reported scanline errors of the baseline and the improved correspondences (IC):

	Baseline	Baseline and Ransac	IC	IC and Ransac
Mean	8.53	4.14	1.92	1.52
Standard deviation	7.14	2.57	1.40	1.40

2 Assignment 2

Throughout this assignment we will use image A (Figure 8a) and image B (Figure 8b) to measure similarity. Both image have been downsampled and then blurred with a Gaussian kernel with sigma being 1. For the three similarity graphs, the results have been normalized such that all optima touch 1.0.

(a) Image A.

(b) Image B.

Figure 8

2.1 Self similarity

We begin our examination of the four similarity measures by looking at self similarity. Here image A is compared to itself. The results can be seen in Figure 9.

From the results we note that all similarity measures should be maximized with the exception of P-norm, and thus the P-norm goes above 1.0. We can also see that the similarity measure with the biggest relative difference is NMI, which means NMI both have the greatest gradients, but also is the most sensitive to displacements. In comparison, we see that MI is almost completely flat. However, because MI is so flat, it makes the curves more smooth when we get close to 1.0, whereas for NMI we get some uneven sudden changes as we approach 1.0. For all similarity measures we see them 'peak' at 1.0 as expected as the images should be most similar when not displaced.

2.2 Multi modal similarity

We now take a look at multimodal similarity. Here we have used image A and image A inverted, that is, the intensities have been multiplied by -1. The results can be seen in Figure 10.

Although it is hard to see, we have that the P-norm is slightly above 1.0 except when the displacements are greatest where it touches 1.0, and we have MI is slightly below 1.0 except when we have 0 displacement where it touches 1.0. This can be seen in Figure 11. We note that NCC now finds the optimal similarities

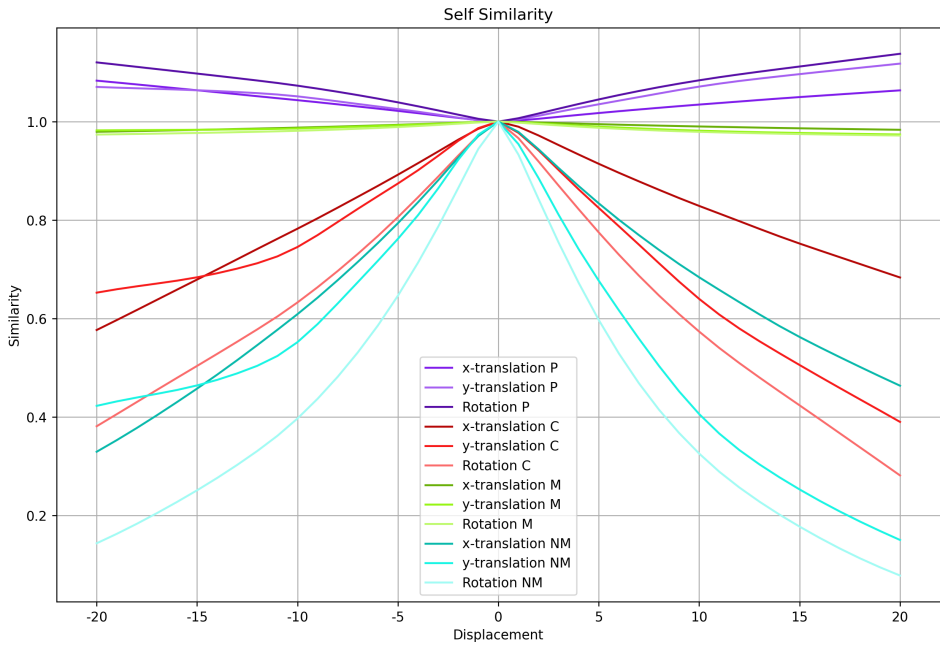


Figure 9: Self similarity results normalized such that all optima touch 1.0. Both images have been blurred by a Gaussian kernel with sigma being 1. Translation displacement is measured in pixels and rotation displacement is measured in degrees. P = P-norm with P being 2, C = normalized cross correlation, M = mutual information, NM = normalized mutual information.

when the displacements are greatest and the smallest similarity when the displacements are 0 as we have normalized the similarities such that the optima touches 1.0. This means NCC is not invariant to intensity changes, and thus is not suitable as a similarity measure for multi modal images. The same is the case for P-norm which also has its optima where the displacement is greatest. We see that MI behaves correctly and thus is invariant to intensity changes, however, MI become extremely flat, and so its gradients are almost 0. This would make it very hard to optimize the similarity measure with a gradient based method, but it is smooth. Lastly we have NMI which also shows invariance to intensity changes and, of the correctly behaving similarity measures, shows the greatest sensitivity to the displacements. This makes the gradients bigger and thus the function easier to optimize. We also see the curves being more smooth than during self similarity. The reason for P-norm and NCC not being invariant to these intensity changes is they both rely on intensities to compute the similarity, and thus, when we invert one of the images we get negative values as part of the similarity measures, which seemingly flips everything around the x-axis. As MI and NMI works on the distribution of the intensities, and not the intensities themselves, it does not matter if the values have been inverted, it only matter whether we observe as many inverted values as non-inverted values, i.e. that the distribution of pixels are the same in the two images.

2.3 Object similarity

We now look at the similarity between image A and B. The results can be seen in Figure 12.

We here see that all similarity measures again behaves correctly, however, compared to the self similarity results in Figure 9 the graphs have become more 'pointy' which would indicate the images are less similar as we need a smaller displacement to make the images dissimilar. By looking at the un-normalized similarity values it can also be reported that the similarity at optima has dropped for all similarity measure. However,

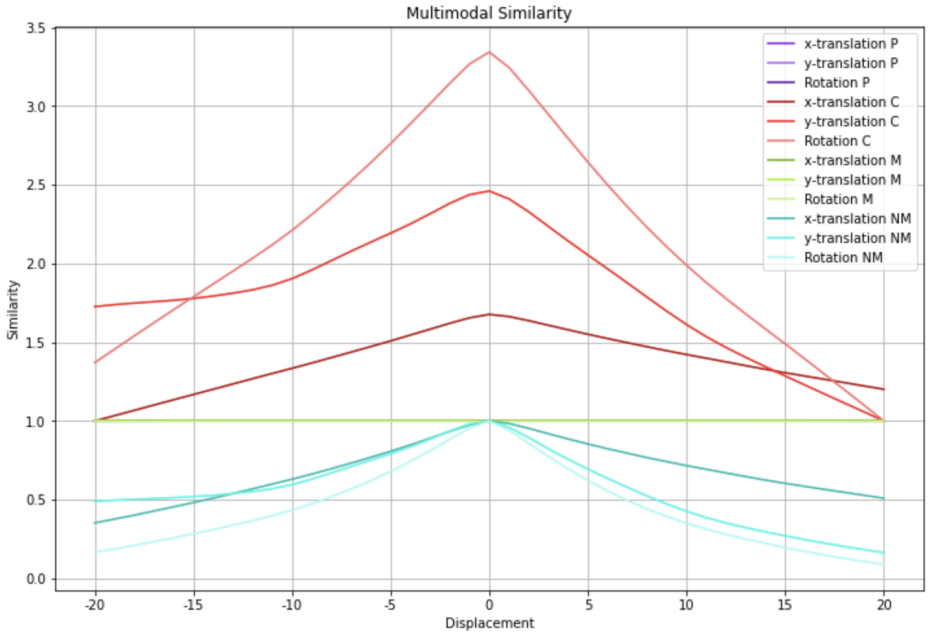


Figure 10: Multi modal similarity results normalized such that all optima touch 1.0. Both images have been blurred by a Gaussian kernel with sigma being 1. Translation displacement is measured in pixels and rotation displacement is measured in degrees. P = P-norm with P being 2, C = normalized cross correlation, M = mutual information, NM = normalized mutual information.

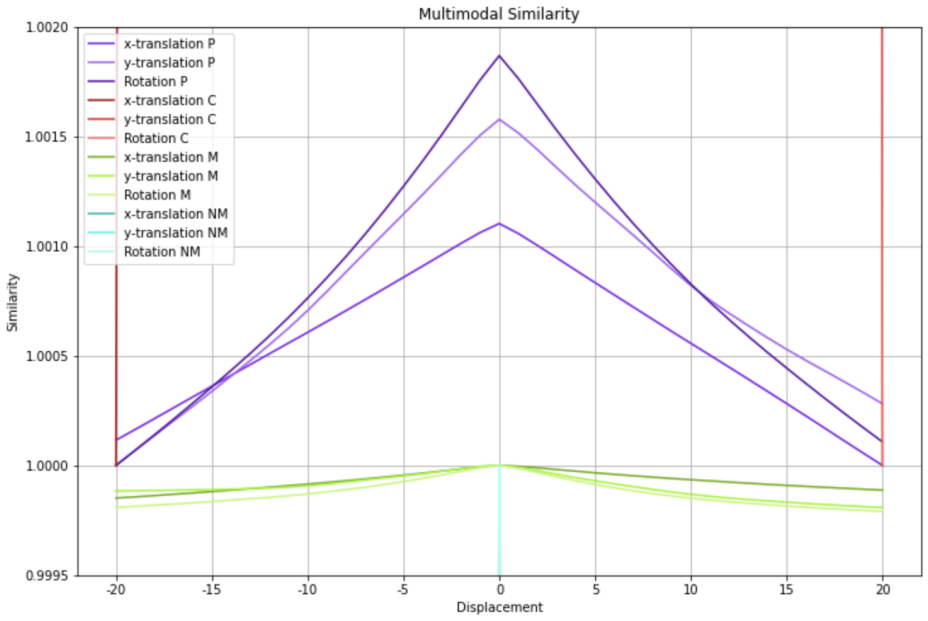


Figure 11: Zoomed multi modal similarity results normalized such that all optima touch 1.0. Both images have been blurred by a Gaussian kernel with sigma being 1. Translation displacement is measured in pixels and rotation displacement is measured in degrees. P = P-norm with P being 2, C = normalized cross correlation, M = mutual information, NM = normalized mutual information.

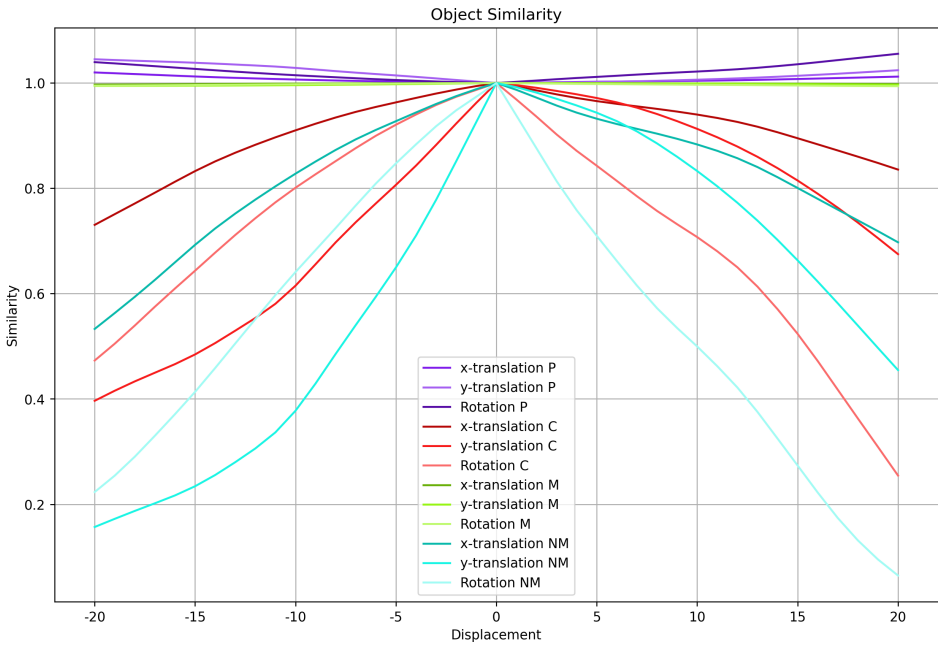


Figure 12: Object similarity results normalized such that all optima touch 1.0. Both images have been blurred by a Gaussian kernel with sigma being 1. Translation displacement is measured in pixels and rotation displacement is measured in degrees. P = P-norm with P being 2, C = normalized cross correlation, M = mutual information, NM = normalized mutual information.

we still see the optima being at 0 displacement. This might be due to the foreground and background being the most 'dominant' features in the images, and thus translating and rotating these out of the images might result in a bigger dissimilarity than aligning the images on the motif. We also see for NCC and NMI that the curves have become rather uneven / wavy and not as smooth as for self similarity. This is especially true around 1.0. The more wavy shapes of the curves could be an issue if we would like to optimize the functions with a gradient based method as the waves could form local minima and maxima. We see that the gradients of P-norm and MI is even more flat than for self similarity, making them hard to optimize.

2.4 Registration

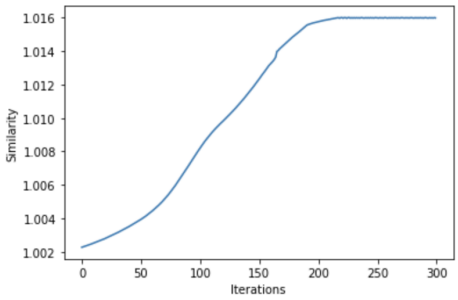
We can now look at how our similarity measures perform when we make a rigid and affine registration. In the simplest case we can make a rigid self registration (image A with image A) using NMI. Because we know an optimal transformation matrix will have $\theta = 0$, $t_x = 0$ and $t_y = 0$ we can initialize the transformation matrix to have $\theta = 0.2$, $t_x = 5$ and $t_y = 5$ and see if it returns to all zeros. If we let our gradient descend optimizer run for 200 iterations with a rotation learning rate of 1 and a translation learning rate of 100 we get that the final transformation matrix have $\theta = 0.0023$, $t_x = -0.039$ and $t_y = -0.014$ which is very close to our expectation. We can now perform a rigid transformation using NMI again, but this time using image A and B. We begin by initializing $\theta = 0.3$, $t_x = 5$ and $t_y = 5$. The results can be seen in Figure 13.

We report that the optimized transformation matrix goes towards 0 for all parameters which means it finds the optimal transformation to be no transformation. This is also consistent with the results seen in the object similarity section. In Figure 14a we see the similarity at each iteration of the optimization, and we see we have found an optima. As discussed earlier, the reason for the optimal parameters being 0 are probably because the foreground is dominant in the images, and optimizing the amount of foreground is probably

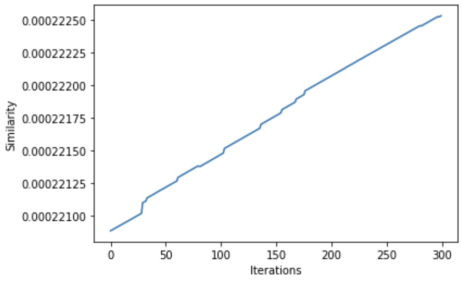
(a) Transformed image.

(b) Target image overlaid with transformed image.

Figure 13: Result of rigid registration using NMI on image A and B.



(a) Similarity using NMI at each iteration of optimization with translation learning rate of 100 and rotation learning rate at 1.



(b) Similarity using NCC at each iteration of optimization with translation learning rate of 10 and rotation learning rate at 0.1.

Figure 14

better than optimizing the overlap of buildings.

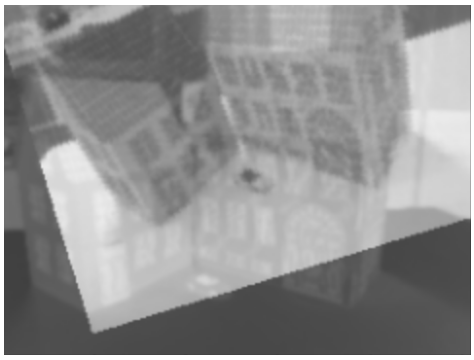
We can now perform a rigid registration using NCC, image A and image B inverted. Here we initialize with the same parameters, but use a rotation learning rate of 0.1 and a translation learning rate of 10. The results can be seen in Figure 15, and the similarity at each iteration can be seen in Figure 14b.

We here note the motif is getting turned out of the resulting image. This is likely again because the foreground is attempted matched, and rotating the image introduces black pixels into the foreground. This illustrates that NCC is *not* intensity invariant. We also note from Figure 14b that we have not yet reached an optima, which means if we ran more iterations or used bigger learning rates, the image would rotate further.

We can now perform an affine registration using NMI and image A and B. We initialize the transformation matrix with the same θ, t_x and t_y values as before and initialize $s_x = s_y = 1.2$. The rotation learning rate is set to 0.7, for translation it set to 100, and for scaling it is set to 1. The results can be seen in Figure 16 and the similarity at each iteration can be seen in Figure 17.

We here note it is attempted to align the foreground of the images, however looking at Figure 17 we see some volatile oscillation towards the end indicating we are using too big learning rates. This is likely the reason the the transformation matrix is not returning to 0 displacement. However, we see the houses and the car overlap slightly more than before.

(a) Transformed image.



(b) Target image overlaid with transformed image.

Figure 15: Result of rigid registration using NCC on image A and inverted B.

(a) Transformed image.

(b) Target image overlaid with transformed image.

Figure 16: Result of affine registration using NMI on image A and B.

2.5 Conclusion

In conclusion we have seen that the four similarity measures are smooth on self similarity, but becomes rather uneven on object similarity. We have also seen that the most dominant features of the images are most likely foreground and background as the optima of the object similarity is at 0 displacement for all similarity measures. We have also seen that NCC and NMI have the greatest gradients and are thus the easiest similarity measures to optimize, whereas P-norm and MI have gradients very close to 0. We have also seen none of the similarity measures are translation or rotation invariant, however MI and NMI are intensity invariant due to how they use distributions to measure similarity rather than the intensity values. Lastly we conclude affine registration is harder to optimize due to the increased number of parameters, but aligns the

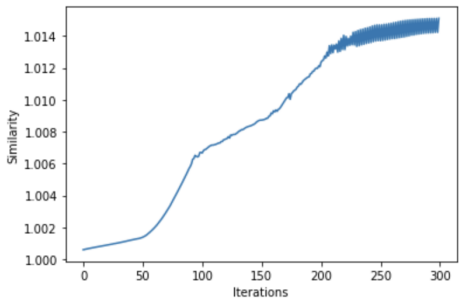


Figure 17: Similarity at each iteration for affine registration between image A and B using NMI.

motif in the images slightly better than a rigid registration, although this might be a coincidence.

3 Assignment 3

3.1 Design of experiments

For this assignment a subset of the 'HPatches' data set has been chosen, namely the first 10 series with illumination changes and the 10 fist series with view point changes. Three image descriptors, namely SIFT, ORB and SURF will then be evaluated on the aforementioned data. A consideration for the experiments have been, that an image descriptor both chooses a location in an image which it finds fitting, and it describes this location. That is, we both get key points and descriptors for each found feature in the image. To evaluate the key points we use *mean localization error*, to evaluate descriptors we use *nearest neighbour mean average precision*, and to evaluate the image descriptor as a whole, we use *homography estimation*. Before going into details with these error measures, we will first look at how we classify true positives, false positives and false negative detections.

For each pair of images considered, we apply the image descriptor to both images, and the detections made in *image 1* are considered ground truth whereas the detections made in *image 2* are considered candidate points. If there is a perspective change between the two images, the ground truths are warped into the perspective of the candidate points. To detect false positives, we go through each detection in our candidate points, and each candidate point which does not have a similar ground truth is considered a false positive. To detect false negatives we go through each ground truth and if it does not have a similar candidate point it is a false negative. A true positive can be measured as either each candidate point which has a similar ground truth, or each ground truth which has a similar candidate point. This 'similar' / ϵ term is measured as the L2 norm, and can be increased / shrunk in order to create precision / recall curves. We will compute precision and recall by starting with $\epsilon = 0$ and increase it. As we might have more ground truths than candidate points and vice versa, the recall and precision is rescaled to always go from 0 to 1. To ensure correct curves, 1 - recall will be used to compute precision / recall curves. An example of a precision / recall curve can be seen in Figure 18. Each ground truth or candidate point can only be matched once, that is, if we have two candidate points around a ground truth, only one of them will be considered a true positive whereas the other will be considered a false positive.

opencv implementations of the image descriptors have been used, and all image descriptors have their default parameters.

3.2 Error measures

We can now discuss the definition of each error measure.

Mean localization error is defined in accordance to [2] where we only use true positive detections and find the average distance between their nearest neighbours. This measure tells us about how close the key points are to each other in the two images, that is, if the image descriptor finds the same points in the two images. The error ranges between 0 and ϵ, where ϵ is the maximum distance between the two key points there can be for them to be considered true positives. As it is unclear which epsilon is used in [2], I have chosen to report the mean localization error for a range of epsilon values. To avoid rewarding image descriptors which find 0 features, the maximum error (ϵ) is used for a run where 0 features are found.

Nearest neighbour mean average precision is defined in accordance to [2], and takes all descriptors of an

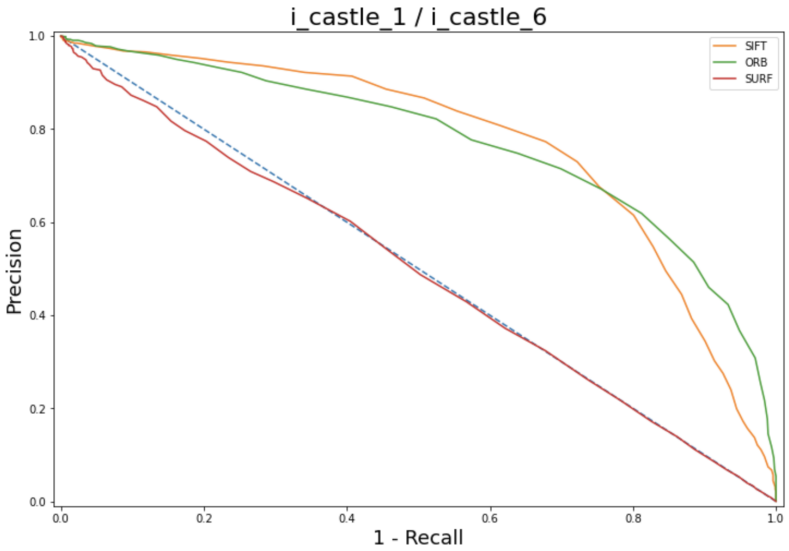


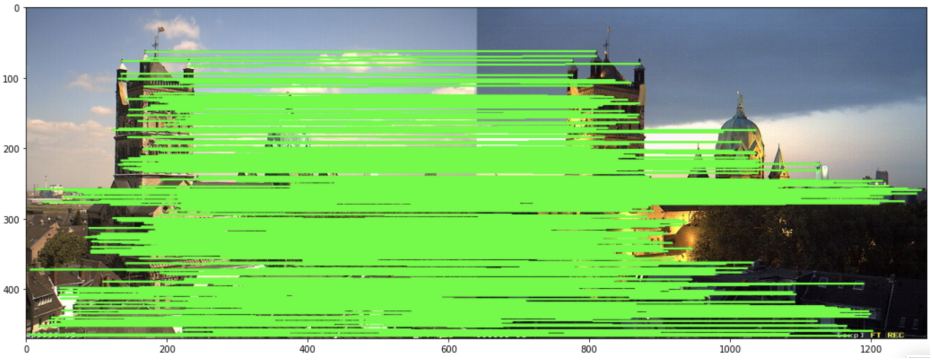
Figure 18: Precision / recall curve for two images in the castle series.

image pair and finds their nearest neighbour. Then creates a precision / recall curve and measures the area under the curve for this image pair. Then the average area under the curve is estimated across all image pairs. This measure estimates how well the image descriptor detects true positives without introducing false positives. The measure ranges between 0 and 1. As we are comparing the same scene, the descriptions made should describe the same image patches, and thus we do not perform any perspective warping for this measure. As it is not clear from [2] which range of ϵ values are used, I have chosen to use 100 evenly spaced ϵ values between 100 - 800 for SIFT and ORB as the distance between all found descriptors in the manually inspected images seem to be matched in this range. For SURF the range 0 - 1 have been used. To make this measure computational feasible, only the first 1000 descriptors found are used. These are likely in the same part of the image, but sampling 1000 descriptors across the image might result in a lot of descriptors which are not measured at the same key points.

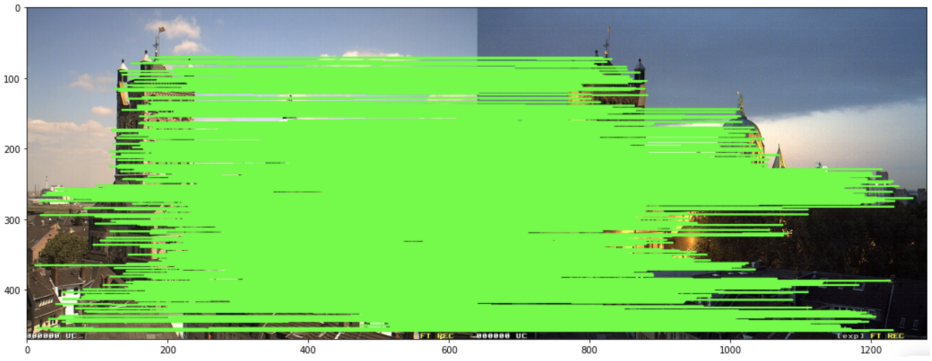
Homography estimation is defined in accordance to [2] and takes the four corners of *image 1* and first measures the average distance between each corresponding corner transformed by the true homography matrix and the estimated homography matrix found by taking nearest neighbour matches found by the image descriptor. If the average distance is less than some ϵ then it is reported as a correct homography estimation, otherwise it is reported as a wrong estimation. The average number of correct estimations over all image pairs is then reported as the *homography estimation* measure. This measure evaluates the image descriptor as a whole by taking points matched in the two images to estimate a homography. The result ranges between 0 and 1. As SIFT and SURF returns *a lot* of features, a Lowe's ratio at 0.7 is used to filter the number of features used to estimate the homography.

3.3 Results

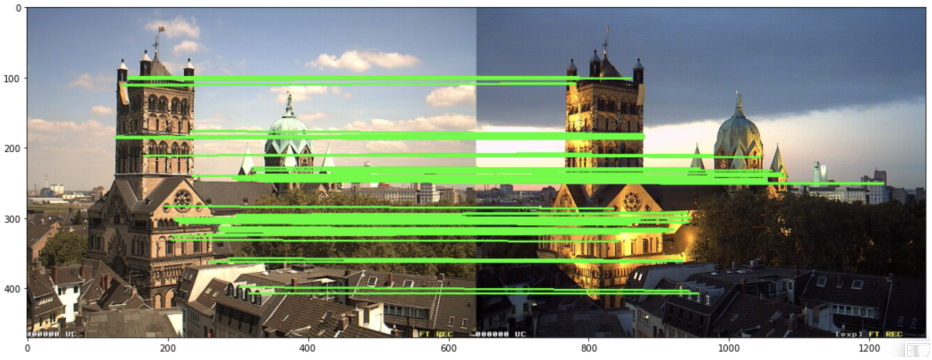
To get an idea of how many features each image descriptor produces, we will begin this section by taking a look at three image pairs where each image descriptor have been run. The found features can be seen in Figure 19, and only shows correct correspondences. By manual inspection, it can also be reported that in general (when not constrained) SIFT and SURF produces more matches than ORB.



(a) The detected SIFT features.



(b) The detected SURF features.



(c) The detected ORB features.

Figure 19: Each image descriptor run on the same image pair to give an indication of how many features each finds.

As seen, SIFT finds about as many features as SURF whereas ORB finds a considerable amount less.

In Table 1 we see the results of the experiments conducted. When comparing SIFT and ORB we see their ability to estimate correct homographies varies greatly. ORB estimates as many correctly homographies when epsilon is 5 as SIFT does when epsilon is 1, that is, SIFT is much more accurate when estimating homographies. SURF lies in between the two, performing closer to SIFT than ORB. However, when we compare the mean localization error SIFT, SURF and ORB obtain similar errors. This means their ability to choose the same places in the image pairs are equivalent. However, looking at the actual errors, we see the error is about halfway between the minimum and maximum possible which means we are fairly far away from the actual location we were looking for. When it comes to nearest neighbour mean average precision SIFT and ORB performs similar and fairly well, whereas SURF performs quite poorly.

	Homography estimation			Mean localization error			NN mean average precision
	eps = 1	eps = 3	eps = 5	eps = 1	eps = 3	eps = 5	
SIFT	0.34	0.61	0.71	0.56	1.46	2.41	0.72
ORB	0.05	0.21	0.31	0.54	1.54	2.41	0.73
SURF	0.29	0.53	0.66	0.56	1.48	2.27	0.50

Table 1: Results of the experiments.

3.4 Discussion of results

A reason for the big difference in homography estimation performance between SIFT and ORB might be how many feature points each find and are able to match. SIFT does find a considerable amount more features than ORB does, and because SIFT have more points, and point matches, it is able to estimate the homography better. RANSAC is used to eliminate outliers, which means even though SIFT might find more false positive matches, some are eliminated. It is likely SURF performs better than ORB for the same reason, however, because SURF produces worse descriptors (seen from nearest neighbour mean average precision) than SIFT, it is worse at estimating the homographies than SIFT.

The fact the mean localization error seems to be fairly high for all image descriptors might be due some aliasing or other approximation of the new location after moving the ground truth point locations into the coordinate system of the candidate points. If the new point location is off by 1 or 2 pixels it will make a quite big error in the mean localization error.

We see ORB and SIFT performs fairly well on nearest neighbour mean average precision, which is likely because they both are scale and rotation invariant, however, SURF is not rotation invariant, and this might be the reason it performs a lot worse. Another consideration when comparing nearest neighbour mean average precision is the length the descriptor vectors. SIFT creates a vector with length 128, ORB's descriptor vector length is 32 and SURF produces descriptors with length 64. When we measure the distance between these vectors the higher dimensional vectors are more likely to lie far away from each other. Having this in mind, SIFT seems to describe its features more accurately than ORB although their score is about the same.

3.5 Conclusion

In conclusion we have seen SIFT and SURF detects roughly the same amount of features where ORB detects considerable less features. All three image descriptors perform somewhat bad when it comes to mean localization error, but this is likely due to aliasing or approximation error when transforming the images. When it comes to nearest neighbour mean average precision we see SIFT and ORB performs similar and fairly well, however SURF performs quite bad. This might be due to SIFT and ORB being both scale and rotation invariant, whereas SURF is only scale invariant. Lastly we have seen SIFT performs better than both SURF and ORB when it comes to homography estimations, which is likely because SIFT both detects a lot of features and describes them well.

If we are to determine which of the three is better, it would depend on the task at hand. Although it is clear SIFT performs the best in the these experiments, it is also the slowest of the three. If the task requires fast feature matching it would not be possible to use SIFT, and we would need to trade performance for speed.

4 References

[1] Monnase, P., et. al. 'Three-step image rectification'. British Machine Vision Conference. Aug 2010.

[2] DeTone, D., et. al. 'SuperPoint Self-Supervised Interest Point Detection and Description'. Unknown publisher. Apr 2018.

5 Collaboration

Throughout the course I have been collaborating with Marcus Hansen on both code and discussion of course content.