Assignment 2: Feature Extraction and Matching

1. Detecting Interest Points (features)

An interest point can be defined as a point in an image that has a well-defined position, a distinctive description and have repeatability so it can be found in several images despite rotation etc.. Hence, interest points can be localized making it possible to estimate their location. One example of such is a corner; placing a small window over a patch of constant image values and moving the window in any direction, will not change the pixel intensity significantly. Likewise, if the window moves along the edge direction intensity does not change either. But at a corner, shifting the window in any direction should give a large change in intensity; making it possible to estimate its location (see figure 1).

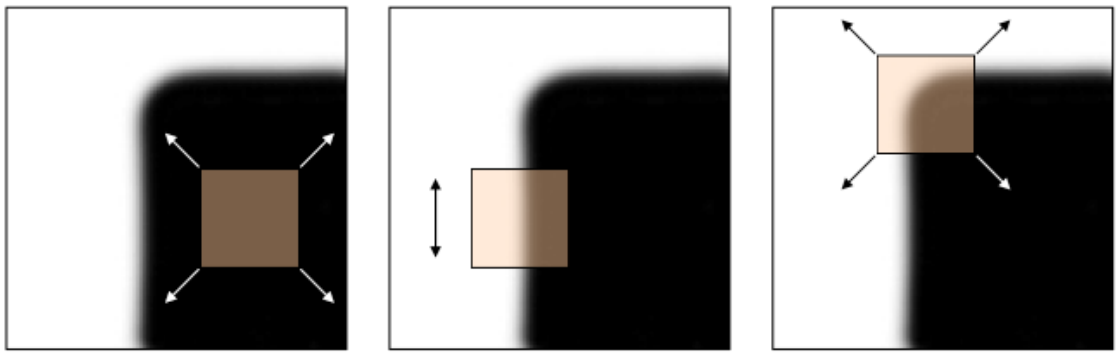


Figure 1. Corners

But an interest point is necessarily not a corner, it can also be an isolated point of local intensity minimum or maximum like line endings or a point on a curve where the curvature is locally maximal.

In this assignment we used a blob detector: Laplacian of a Gaussian (LoG) as our interest point detector (see figure 2,3 and 4). A blob is describe as a group of connected pixels (a region) in an image that share some common property and differs in this property, compared to the surrounding regions. Each blob are in our assignment marked with a black circle. By convolving the images with a Gaussian kernel and then applying the Laplacian operator, we used the LoG filter extrema to locate blobs: locating pixels that are bigger/smaller than all neighbors. When applying the LoG operator, the magnitude of the Laplacian response is strongly dependent on the relationship between the size of the blob

structures in the image and the size of the Gaussian kernel that is used for pre-smoothing. This means that the zero crossings of the filtered image come closer together and produce a peak in the response curve, when the scale of the Laplacian is matched to the scale of the blob. The maxima gives strong positive responses for dark blobs on light background when $\sigma = r/\sqrt{2}$ and vice versa for the minima that gives strong negative response: light blobs on dark background.

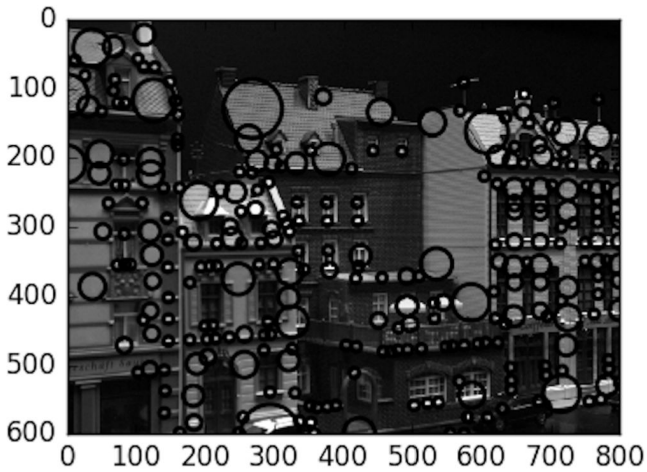


Figure 2: Feature detection with different radius for image 1.

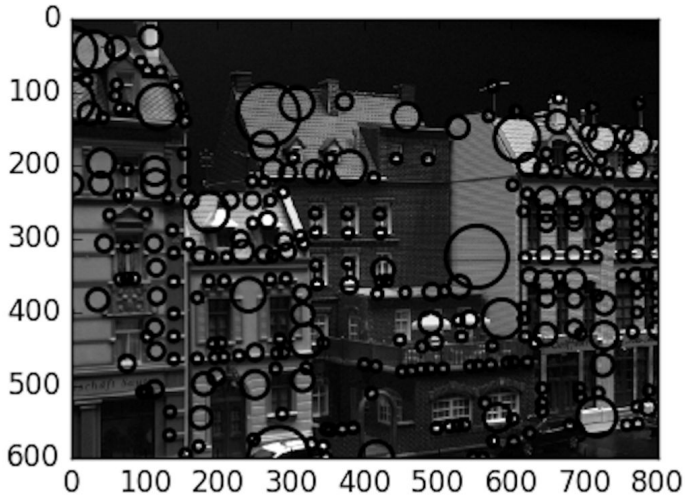


Figure 3: Feature detection with different radius for image 2.

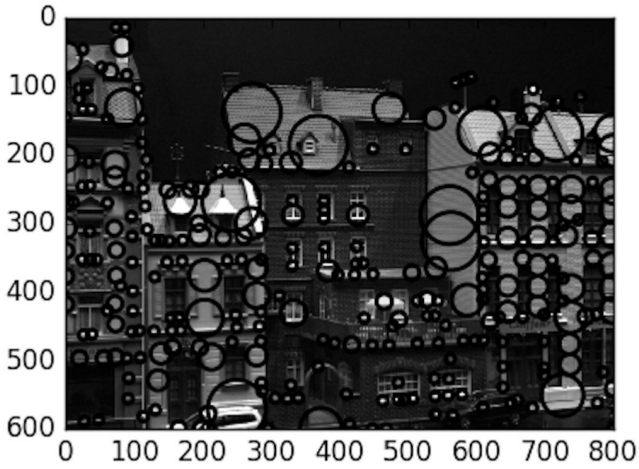


Figure 4: Feature detection with different radius for image 9.

Since we don't know the size of the blobs and want to identify blobs at different sizes, we varied the σ-value of the Gaussian kernel in the Laplacian using the skimage function blog_log.

The σ-value of the Gaussian kernel ranged between min_sigma = 5 and max_sigma = 30. The skimage function has a parameter 'num_sigma' that we sat to 5. This represents the number of intermediate values of standard deviations of the Gaussian Kernel to consider between the min_sigma and max_sigma. Therefore it can be observed in figure 2, 3 and 4, that we have blobs at different sizes (σ between 5-30).

Being able to compare all the responses we got at the different scales, we normalized the response to each Gaussian kernel by multiplying by sigma and each Laplacian by σ^2 . When convolving with many different sizes of Laplacian of Gaussian kernels, the blob_log function returns the coordinates and the standard deviation of the Gaussian kernel, that detected the blob if the max response was above the threshold (0.1). The blobs detected in figure 2 and 3 are quite similar in comparison; almost all of the same blobs has been detected when the perspective/angle of the building change (from image 1 to image 2). Figure 4 differ more from the two other figures in the detection of blobs, but still a lot of the same blobs are detected even though the angle changes. Overall, we think that our feature detector were able to detect the blobs with high repeatability.

Part 2: Simple matching of features

By using the Laplacian of Gaussian to extract points of interests (blobs) we can exploit these to compare features between two pictures. Comparing features solely based on their pixel values works on pictures of similar objects from similar angles, but if the pixels change due to lighting, angles or similar the features will never match. A good comparison should be robust in terms of noise, perspective (such as rotation, skewing or lense distortion), luminance and size of the feature. To compare across images we use the notion of image descriptors. In theory a descriptor is normalized across images and can describe a feature independent of the pixel values. In practice we have use a simple square pixel patch with NxN pixels where $N \in \mathbb{Z}_+$. To compare two patches we the dissimilarity measure d to get a value between two patches (F_1 and F_2):

$$d(F_1, F_2) = \sqrt{\sum_x \sum_y (F_1(x, y) - F_2(x, y))^2}$$

The smaller the d, the closer the patches are to each other. To determine the value of N we tried and compared values ranging from 5 to 40. It became clear that more features was matched and vice versa. This can be explained by the fact that larger patches increase the radius of the areas being examined, decreasing the significance of the actual features. In other words the larger the patch, the more pixels who are not directly related to the feature will be tested. This will give a higher value of d and result in more, but less accurate, matches. We found the value 20 to give a good result. The feature matching using the above description between images 1, 2 and 9 are shown below in figures 8, 9 and 10.

To avoid selecting too many matches we implemented a filter for the dissimilarity search. When comparing two images, we iterate through all descriptors in the second image, for each descriptor in the first, calculating and ordering the patches using their dissimilarity scores. If the similarity of the best match is very close to that of the second-best match, it can mean that the patch is too generic, resulting in not just one, but many likely matches. Given our images a patch can only match one other patch, so such generic patches should be removed.
This has been implemented by looking at the ratio of the two best dissimilarity scores and ignoring them, if the ratio is above a certain threshold t.

Figures 5, 6 and 7 below shows histograms of the feature dissimilarity between the three images. The Y-axis describes number the features and the X-axis describes their dissimilarity score. The lower the score, the better is the match between the two features, therefore it is preferable to have more occurrences with a low score than a high one. This means that the matches found has a higher chance of being unique. Note that the dissimilarity score goes to 1, which means everything is included in the figures and the threshold hasn't yet been taken into account. Looking at these graphs it's clear that dissimilarity score is much more evenly distributed in figure 5 and also has more values with a low dissimilarity score than figure 6 and 7. This means that there has been found more similar features between image 1 and image 2 than between them and image 3, which obviously makes sense since image 1 and 2 are visibly more similar. This is also confirmed by the standard deviation, which for fig 5 is 0.17 and for fig 6 and 7 is between 0.6 and 0.7.

Figure 5

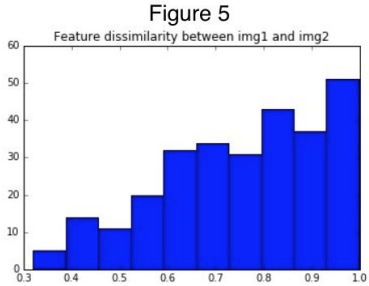


Figure 6

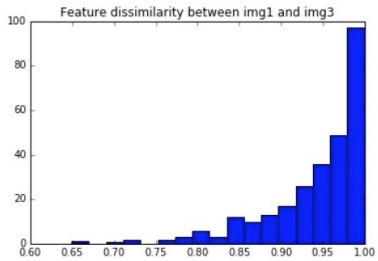
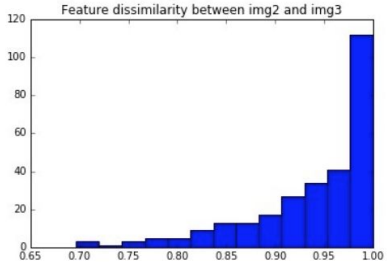


Figure 7



Therefore we decided to set a lower threshold value for the dissimilarity score between image 1 and image 2, since we had a lot more occurrences and were able to pick the best matches. We set the threshold to 0.7 between image 1 and 2 and ended up with a standard deviation of 0.1. We set the threshold to 0.9 between the two others and ended up with a standard deviation at around 0.5 for both comparisons.

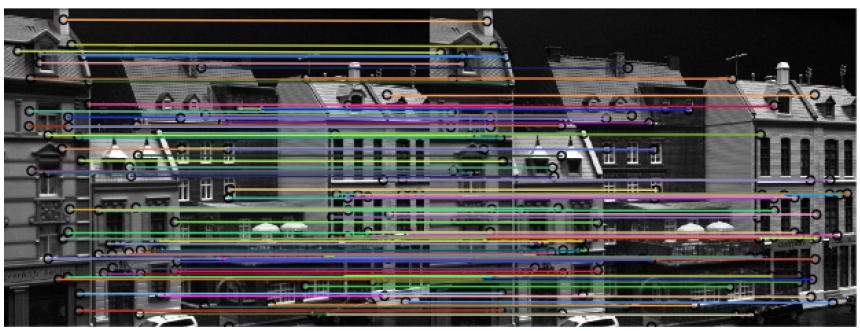


Figure 8: Image feature comparison between image 1 and 2.

N = 20, t = 0.7. Number of matches: 100.

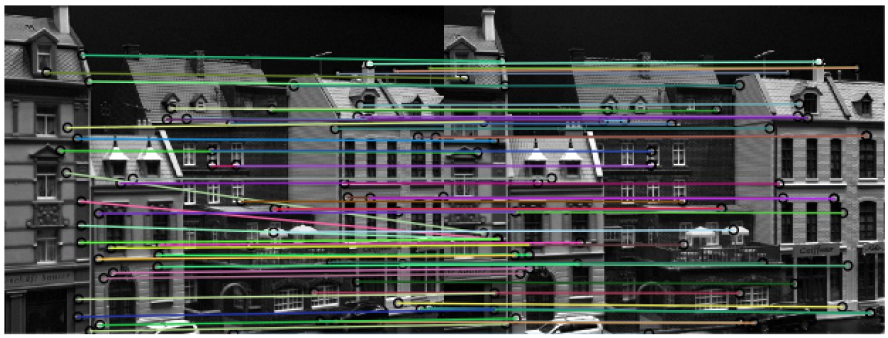


Figure 9: Image feature comparison between image 2 and 9. $N = 20,\ t = 0.9$. Number of matches: 64.

Figure 10: Image feature comparison between image 1 and 9. $N = 20,\ t = 0.9$. Number of matches: 54.

Figure 8 shows 100 matches between the first and the second image. Looking closer most matches look to be correct despite the high value of N. Figures 9 and 10 only has 64 and 54 matches respectively. However, looking at figures 8, 9, and 10, it is clear that figure 8 and 9 are more similar than 9 and 10. This, in turns, makes it easier to match the same features from 8 and 9, than from 9 and 10. The lower match from the second to third image is thus expected. However, the algorithm still captures the most distinct features, despite the change in the viewing angle. Because the error-rate increases when the pictures are less similar, we increased the error threshold t in the latter comparisons as mentioned before.

Conclusion

In conclusion our features matching was a success. We found that a patch size around 15-20 gave the best results. Many of the matches are lines which extend well beyond 20 pixels, which could explain the large patch size.

There are several approaches to further the accuracy that we did not investigate. One could be to normalise the patches with regards to illumination. Despite showing the same feature, two patches taken from two different images might have different grayscale values. To avoid lighting playing a role, the pixel values can be normalised, giving a more uniform distribution of values from 0 to 255. This should increase the chance of matching similar features across different lighting scenarios, as well as limit the number of wrongly matched features due to uniform variances in pixel values.