Computer Vision: Image

UNIVERSIDADE D
COIMBRA

José Lopes Paulo Cortesão

March 31, 2023

Contents

1 Introduction

This report aims to describe the process undertaken in the implementation of an image mapping system where the Harris pipeline is used to detect corners in images, extract relevant features and map the location of objects in different images. The implementation details, experimental procedure and results are included here.

2 Implementation

2.1 Harris corner detection

In order to build this pipeline, the various components had to be implemented. Firstly, the corner detection algorithm was written. A filter with the derivative of a 2-D Gaussian was obtained and was used to filter the image, producing a horizontal gradient and then a vertical one, when transposed. The element-wise square of each derivative and the product between them was computed, to allow for the determination of the Harris matrix for each relevant pixel, and for the respective coefficient of "corniness":

$$H(i,j) = \begin{bmatrix} \nabla_x(i,j)^2 & \nabla_x(i,j)\nabla_y(i,j) \\ \nabla_x(i,j)\nabla_y(i,j) & \nabla_y(i,j)^2 \end{bmatrix}$$

$$C(i,j) = det(H(i,j)) - 0.1 * trace(H(i,j))^2$$

After computing this matrix, the threshold for considering a pixel the center of a corner was determined as a fixed fraction of the maximum corniness coefficient throughout the image. Finally, for each coefficient, non-maximum suppression with a window of NMS_{size} was applied: only the corners whose value was maximum in that window were considered for feature extraction. Their coordinates were stored in a matrix for future use (an offset was added due to the processing).

2.2 Scale and orientation determination

With the coordinates of the corners determined, it is necessary to discover their orientation and scale. The orientation is obtained by passing a gaussian filter on the image and calculating its derivative on both axes using a Sobel filter; and then calculating an angle based on the coefficient between the sum of the vertical derivatives and the sum of the horizontal derivatives, considering a 3-by-3 neighbourhood window centered on each pixel.

To obtain the value for the scale, various possible sizes for filters were defined. Then, scale-normalized Laplacian of Gaussian filter with the selected σ values were to adequately-sized windows centered on each corner, and the σ corresponding to the highest response was chosen as the feature scale.

After the determination, all the properties regarding corners, including coordinates, orientation and scale were saved in the data structure *Pts* to be used later.

(a) $\sigma = 35$ (b) $\sigma = 57$ (c) $\sigma = 91$

Figure 1: Corner activation with different Laplacians

Figure 1 exhibits the response in each pixel with three different scales. It was constructed using the HSV color scheme: the value corresponds to the grayscale value of the pixel, and the hue corresponds to blue for a null response and to red for a maximum one. It is possible to see that smaller features have greater response in smaller σ values and vice-versa.

2.3 Descriptor extraction

In order to extract features to allow the mapping of corners between images, two main descriptors have been implemented: a simple one and the S-MOPS one. In the Simple descriptor, the 5-by-5 pixel window centered on the corner was extracted as a feature. It can be seen that, because of the way it is formulated, this descriptor is very sensitive to rotation and scale, and performs optimally when considering translation. As well as this, the S-MOPS descriptor was extracted. This more complex descriptor is designed to be scale and rotation-invariant: considering the scale and orientation determined in the previous test, the corresponding window centered on the pixel was rotated in the contrary direction, so that its orientation became 0º. After this, the central slice of the rotated square, considering the initial window width, was taken and resized to an 8-by-8 descriptor, which was normalized by the z-score method.

2.4 Feature matching

With the image descriptors calculated, we must now have a way to compare them and match the appropriate descriptors. During the development of this project, two alternatives were implemented. One is the SSD comparison test, which simply computes the pairwise Euclidean distance between the two descriptor matrices. To compute the list of matches, we first need to find the distance of all pairs of descriptors between the two images and then try to compare those that fall below the predefined threshold.

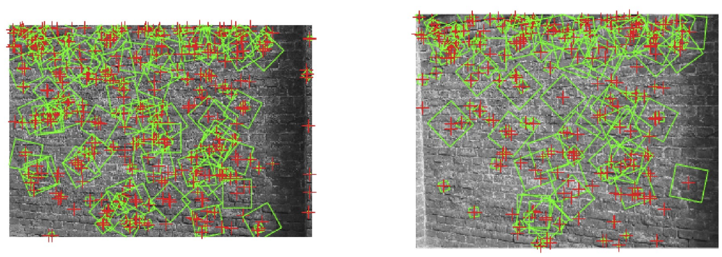
The second alternative is the ratio test, which considers the second best match for each possible match. By calculating the ratio between the best match and the second best match for a given descriptor, we can determine how discriminative that descriptor is. If the closest match is much smaller than the second best match, the ratio is very small, and we can use the threshold to filter out the most discriminative matches.

3 Results and Discussion

After the implementation of the whole pipeline, testing was conducted to verify the quality of the results. Overall, it is considered that they are satisfactory since there is a good portion of the corners that are correctly mapped. Still, this is achieved by fine-tuning the ideal parameters for each image. Some conclusions could be drawn from varying the several parameters involved:

3.1 Corner detection results

One of the first steps in the development of this project was the detection of image corners. After calculating the Harris matrix and using non-maximum suppression we were able to find the image corners.

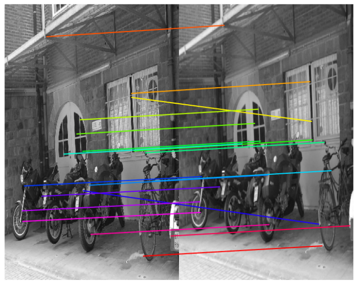


(a) Corner detection with the 'wall' dataset

In Figure 2a, the red crosses depict the locations of the maximums in the image following suppression. Filtering the optimal candidate points for descriptors is crucial for the project's functionality. The experimental part of the project involves a tradeoff between the number of generated corners and the selectivity of the feature-matching process, which is essential for its success.

3.2 Descriptor results

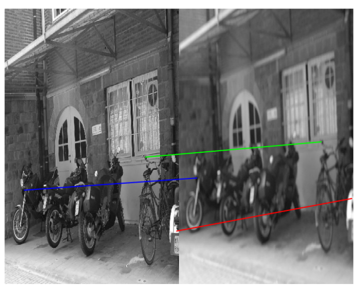
One of the cornerstones of this project are the descriptors because they define and characterize each patch around the key point. Therefore, it is of utmost importance to understand in which scenarios each descriptor works best. The first test was performed on the 'bike' dataset, which contains several images taken at slightly different locations with different blur values.



(a) S-MOPS descriptor result

(b) Simple descriptor result

Figure 3: Both descriptors with a low blur image



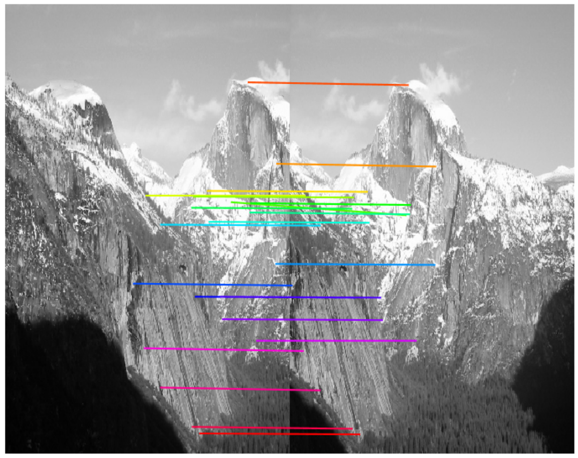
(a) S-MOPS descriptor result

(b) Simple descriptor result

Figure 4: Both descriptors with a high blur image

As can be seen from the previous images, the S-MOPS descriptor provides much more relevant and correct mappings, while the Simple descriptor tends to have more errors. It can also be noted that the S-MOPS descriptor has a much higher tolerance to the blur effect, which in turn makes it more robust.

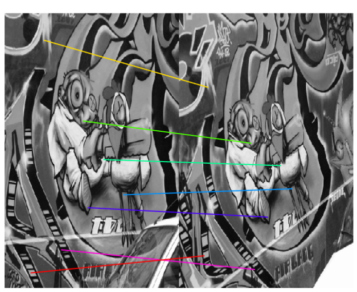
Since there is a considerable number of transformations such as rotation, translation, blurring and others to test, it is difficult to make an overall conclusion about the descriptors. Even though the S-MOPS descriptor gives promising results, the Simple descriptor is more robust in situations where the images are related by translation.



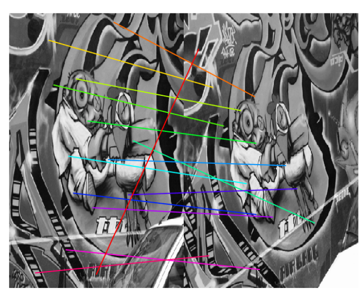
(a) Simple descriptor result with the Yosemite dataset

3.3 Feature matching results

Even if the results obtained with the SSD test are satisfactory, this method presents major problems. In applying this test, it is assumed that each element of the descriptors is equally important for the comparison, i.e., it is not a weighted sum. In contrast, the ratio test considers the second best match, which allows for an easy discrimination of the features. In addition, the ratio test is much easier to parameterize, since the ratio can only be in the interval [0, 1], whereas the SSD threshold can vary depending on the pixel intensity of the image. Furthermore, the range is much wider and harder to understand intuitively.



(a) Ratio metric test



(b) SSD metric test

Figure 6: Metric test with the 'graf' dataset

The ratio metric was tested with the set of parameters $sigma_i = 2$, $sigma_d = 2$, $TreshR = 0.05$, $NMS_{size} = 7$, $Tresh = 0.3$, $Descriptor = simple$ while the simple metric was tested with $sigma_i = 2$, $sigma_d = 2$, $TreshR = 0.05$, $NMS_{size} = 7$, $Tresh = 4500$, $Descriptor = simple$.

As we see in Figure 6 both metrics give very satisfactory results but fine-tuning was needed in order to set the SSD metric correctly with a $Tresh = 4500$.

The main takeaway from the tests is that it is very difficult to find a set of parameters that work well on all data sets. Since different algorithms respond differently to different image transformations, fine-tuning is required to achieve the desired results. A possible alternative to this problem is to set these parameters to vary according to the image properties.

It was also found that there is a tradeoff between the number of keypoints and the selection of the best fitting descriptors. In some scenarios, it was better to generate many potential keypoints and be very selective in the matches (a lower or higher threshold value), while in other cases generating fewer keypoints was beneficial, suggesting that the patterns or gradient variations of the image may play a role in tuning the parameters.

4 Conclusion

This work allowed the authors to become familiar with the corner detection pipeline that allows for mappings between images. Considering the mappings obtained in the empirical part of the project, it is considered that the goals proposed for this activity were achieved.