Computer Vision, 1st assignement

Mathis Lamarre

Harris corner detection

In order to get rid of the noise, I blurred the image using a gaussian filter. As can be seen on Fig.1, the star is smoother, especially at the edges while maintaining the same shape and intensity.

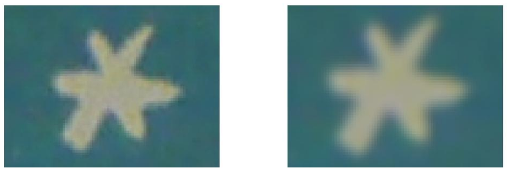

Fig. 1: Detail of picture of a book showing the raw unfiltered version (left) and the smooth filtered version (right).

Then, I computed the intensity gradients (on the grayscale version) in booth the x- and y-direction, namely I_x and I_y, using the gradient() Matlab function. To compute the Harris response, I used the matrix convolution with this 3x3 matrix:

$$C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

By convoluting C with with I_x^2, I_y^2 and I_xI_y, this will select only the values of the pixels in the neighbourhood. The Harris response can the be computed by doing:

$$K = \frac{I_x^2 \cdot I_y^2 - (I_xI_y)^2}{I_x^2 + I_y^2} \tag{1}$$

With this method, K is a matrix containing the response values for each point. To get the corners, we have to find the coordinates of the points whose value is above the threshold.

Non-Maximum-Suppresion

As corners are usually not one-offs and happen in a zone, we should only keep in each subset (the one with the biggest value). To do so, I implemented Non-Maximum-Suppresion (Fig.2). Using the Matlab function ordfilt2(), I checked that every corner has the highest value (the 9th in increasing order) of the 3x3 submatrix around it. In practice, the Non-Maximum-Suppresion and the thresholding were implemented at the same time.

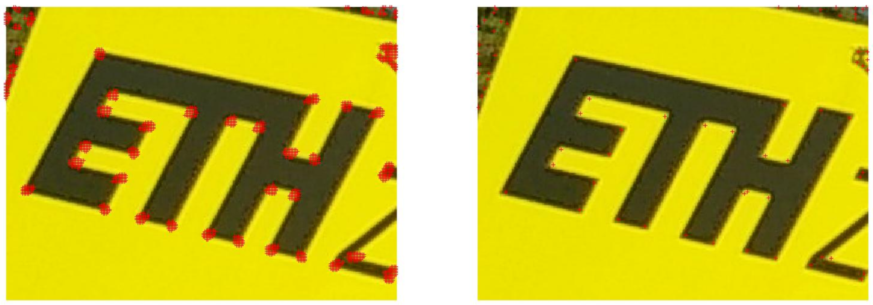

Fig. 2: Detail of picture showing zones with lots of corners without (left) and with (right) Non-Maximum-Suppression. The version on the right only keeps one corner in each region.

Feature descriptor

The feature descriptor is a 9x9 patch centered around each corner. To extract, I used the corners coordinate and selected the surround pixels. The only issue is when the corner is in a edge region. If the patch exceeds the dimension of the image, there will not be any pixel to select from. To remedy this, I performed zero-padding around the image. I added 4 rows and 4 lines of zeros on each side as this is the maximum length of patch that could exceed if the corner were centered on an edge pixel. This displacement has to be taken into account when working with the coordinates of the points.

SSD Feature Matching

In order to match the features, I computed the sum of squared differences (SSD) between the descriptors of keypoints in the two images. For every pair of descriptors, I did the difference of intensity of each pixel and put it to square then summed the entire vector. For each value under a certain threshold, that is to say high affinity, I kept the index of both keypoints as a match. Results can be seen on Fig.3.

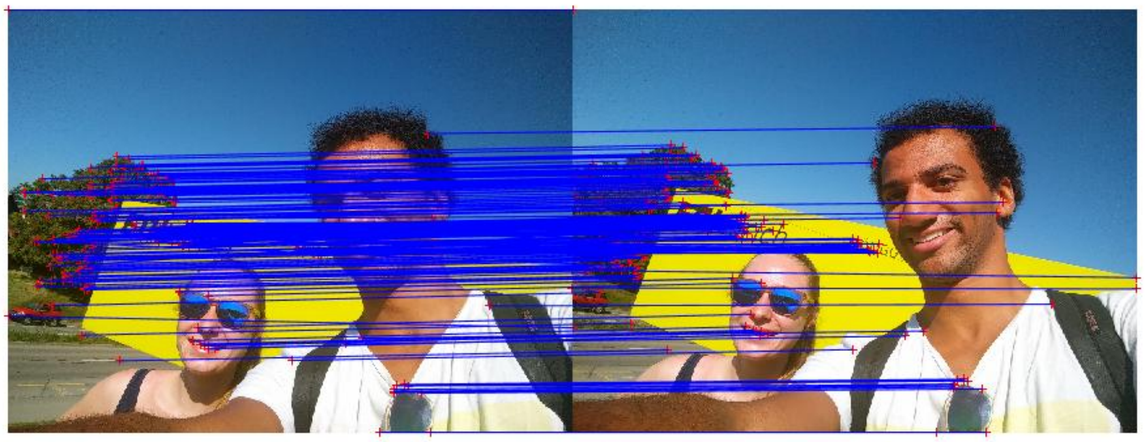

Fig. 3: Matches of two slightly different pictures with Harris Corners and SSD matching of 9x9 descriptors. Lots of keypoints are detected in the trees or in the letters. The matching is accurate.

SIFT features

I used the SIFT feature extractor, descriptor and matcher from VLFeat. See Fig.4.

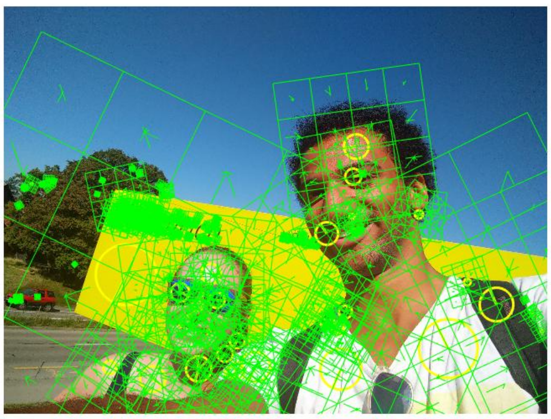

Fig. 4: SIFT keypoints and descriptor. The features have different sizes and orientations.

When comparing the two results (Fig.5 for SIFT, Fig.3 for Harris), we notice that the keypoints of the SIFT descriptor are more diverse. The Harris Corners methods gives lots of keypoints in the tree region, although that is not very relevant. The SIFT method, on the other hand, extracts several keypoints in the face region like the eyes, the mouth, the nose. The matching is also more accurate: all the lines are almost horizontal which means they end up on the corresponding keypoint. For the Harris Corners and SSD matching method however, we can spot a few mistakes, like a keypoint at the front of the car on the left being matched with a keypoint on the grass on the right. This is due to the fact that SIFT keypoints can have different sizes (not only pixels as in Harris Corners) and orientations, allowing the SIFT method to recognize entire regions as an entity.

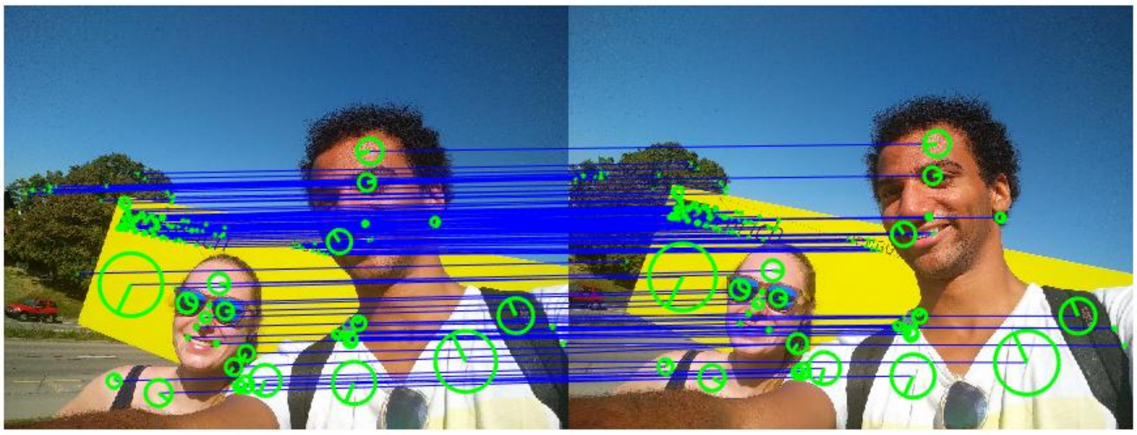

Fig. 5: Matches of two slightly different pictures with the SIFT extractor, descriptor and keypoints. Lots of keypoints are detected in the letters, but also quite a few on the trees and on the faces. The matching is very accurate.