MO446 – Introduction to Computer Vision
Project 2

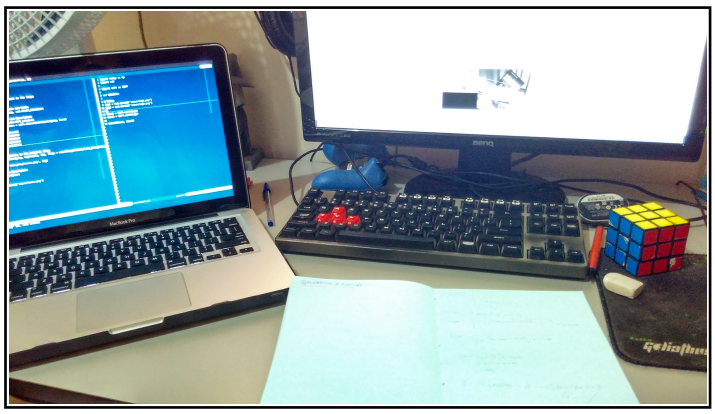
Breno Leite
Guilherme Leite

19/09/2017

Input Video

A video can be seeing as nothing more than a sequence of frames showed in a specific frequency to illude the human eye into perceiving movement, imitating the human vision. With this knowledge in mind it is feasible to abstract the problem of stabilizing an entire video into several sub-problems of stabilizing a pair of video frames at a time. This approach enables the experiments results to be presented in form of images, moving the video output to the very end of the report. The sub-problems experiements that compose the solution were performed on Figure 1a and Figure 1b.

(a) Sub image to be found in cluttered environment.



(b) Cluttered environment.

Figure 1: Initial images used on the experiments.

Important note: *The borders seen in the figures are not part of the image, they are figurative information about the starting and ending points of the image. Moreover, all the image scales in this report were changed in order to make the text more readable.*

Question 3 - Video Stabilization Algorithm

3.1) The SIFT descriptor was chosen to find the interest points and descriptors, we couldn't fully implement it due to the density of details, misleading information, time constrains and complex concepts. Since the remaining of the project requires SIFT's output we opted to use OpenCV implementation of the SIFT descriptor on the following questions. The SIFT descriptor has mainly two independent operations: detection of interest points, also called key points by Lowe, and extraction of a descriptor for each key point.

(a) Keypoints in sub image.

(b) Keypoints in cluttered image.

Figure 2: Keypoints found by the SIFT descriptor.

In Figure 2 all the key points in the image are shown in coloured circles. To detect these key points the algorithm creates a so called scale space which can be seem as a matrix, its columns are called octaves and its rows are called scales. The first octave consists of the original image and it is gradually blurred, by a factor of *sigma*, as the scales increase, the first image of the next octave is a subsample of the first image in the current octave, in our partial implementation the image is subsampled by half its size, along with the octaves and scales creation a Difference of Gaussian (DoG) is also generated. The difference of gaussian is generated within each octave and is done by the subtraction between two consecutive scales, the result is a black image with white spots denoting the edges in the image. With the DoG, for each octave, in hands a pre selection of the key points is executed, for every DoG in every octave, select a pixel and check its intensity against its neighbourhood in its scale and also in the above and below scales, if the select pixel is the greater or the lowest of all points, then it is a approximation of a key point, this is also as far as we got to implement SIFT in our code, the actual key point position is found with by the Taylor expansion. Of all the key points generated in the last step many of them are false positives, so they are filtered out. To remove the low contrast points a threshold is used on the point intensity, the threshold value to be used would be of 0.03 according to Lowe, but further testing would be needed to make sure it fits our data. To remove edges and keep the corners a second filter must be done, at each key point two gradients are calculated perpendicular to each other, in a corner both gradients would be big, since corners are key points they are kept and the rest is discarded, the threshold is done on the ratio between the two eigenvalues and its value is recommended of 5 by Lowe, at the end of the filtering only the actual key points remain.

Before extracting the descriptors for each key point the algorithm ensures rotation invariance by setting a rotation for each key point. A gradient magnitude and orientation is calculated for every pixel around the key point, a histogram of orientations is created discretizing the orientation

angles into 36 bins, the amount to be added is proportional to the magnitude of the gradient at that point, the size of the surrounding to be analyzed is based on the gaussian kernel by a factor of *sigma*. To extract the descriptors a area of 16x16 pixels around the key point is taken in consideration, this area is divided again in 4x4 areas and within this area gradient orientation and magnitude are calculated weighted with a gaussian weight function, discretized to a histogram of 8 bins. This will result on 128 values, gathered in a feature vector, that are normalized and now describe that single key point.

3.2) At this point the SIFT transform has given back all the key points it has found in both images, as seen in Figure 2, and all the descriptors also found in both images. But still there is no relation between the key points in Figure 2a and the ones in Figure 2b, and to relate those key points a matching process is performed.

The goal of the matching is to find the same descriptor, or the closest ones, in both Figure 2a and Figure 2b. To do so an euclidian distance is calculated between any two descriptors, one in each image, and if they are the same then the distance result will be minimal. The distance is calculated between every descriptor in Figure 2a and every descriptor in Figure 2b.

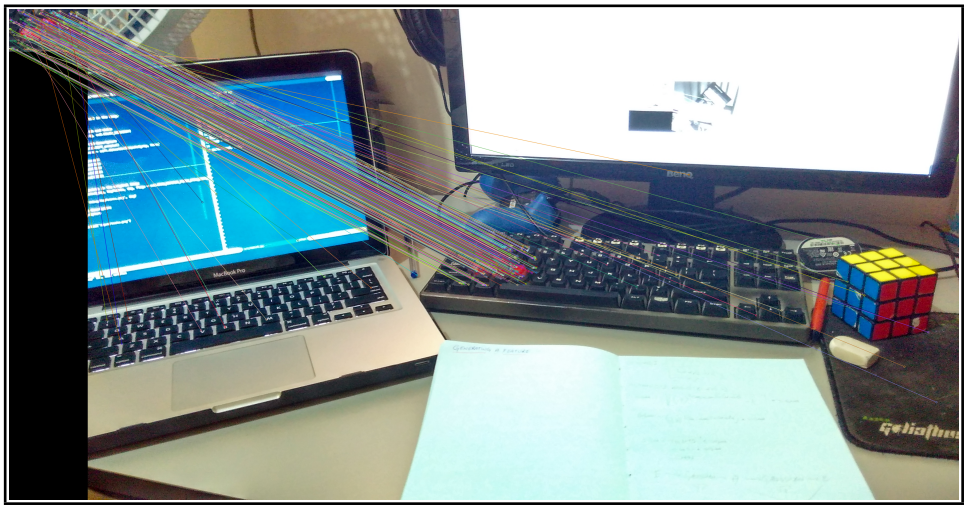
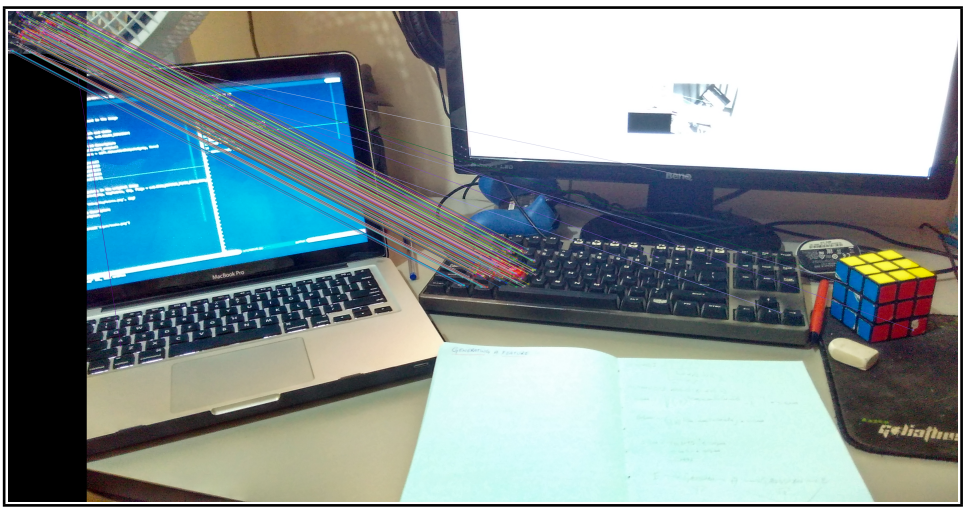


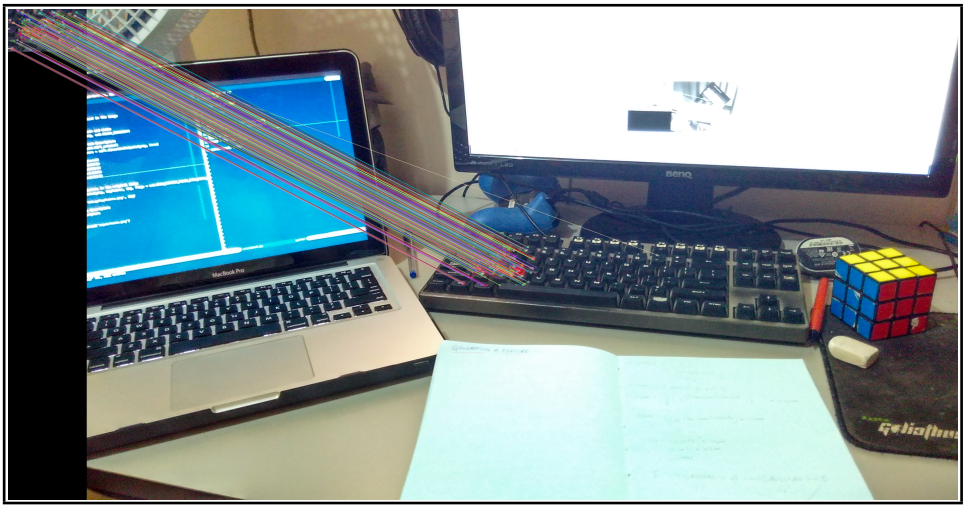
Figure 3: Matching process ignoring ratios.

This algorithm isn't perfect, as seen in Figure 3, some false positives arise because a key point might have a similar one in a far away position of the image, this is common in images with repeating structures, to eliminate these incorrect relations a new filtering occurs, additional to the closest descriptor match, the second closest descriptor is also retrieved and a ratio between both is calculated, if the ratio between both is greater than a threshold the relation is ignored. In his article Lowe suggests a ratio of 0.8 but it didn't performed well with our data, so we used a 0.75 value.

Figure 4 shows the difference between thresholds. Figure 4a shows that at ratio 0.9 the filter still allows some false positives to go through, and Figure 4b at ratio 0.75 was the best result we had before the filter started cutting out correct matches, according to Lowe a ratio of 0.8 should eliminate 90% of false matches and discard only 5% of the correct ones. We also performed a sandbox experiment, Figure 3 ignoring ratios completely, as expected there are many false positives matches because of these points neighbourhoods. We believe that our results difference from Lowe's are caused by distinct cameras and scenes, the keyboard has too many repeating structures that confuses the algorithm.



(a) Ratio of 0.90.



(b) Ratio of 0.75.

Figure 4: Ratio between descriptors distance experiments.

We also computed the runtime difference between two implementations of our algorithm, one using a list as data structure to keep all the key points and thus running over the entire list at every check, and the second one using a tree structure performing a searching in a tree structure on every check. The results can be seem in Table 1, it is noticeable how faster the tree structure is specially when dealing with big images, that is because it doesn't need to check every node in the tree to find the best match.

	Time (seconds)	
Data Structure / **Images Size**	**List**	**Tree**
160x295 | 1836x3264	67.4	13.95
50x50 | 500x501	0.1485	0.0589

Table 1: Comparison between our implementation of the matching with a list and a tree structure.

3.3) To define the number of iterations from RANSAC, we tested different values looking at the performance vs results. The value of 1000 iterations showed a good result within a short period of time, so we have used 1000 for the video stabilization process. If we knew the probabilities on our images, we could choose a better number, however trial and error gave us good results.

Our RANSAC uses a threshold of 100, with have shown success. Decreasing too much this numbers could influences too much on the number of inliers which was a problem. To our model be accepted at least 5 inliers must be matched, this parameter was also found by trial and error.

The matrix A is obtained from these samples and is used on all the remaining points, the error from these new tests are accumulated and later used to choose the best fitting model, this values are called affine parameters.

3.4) Once the affine parameters were defined they are used in our project to transform the video frame $i+1$ in relation to the frame i, the transformation on these frames can cause them to extrapolate the borders of the video window, which create black spots in the image. Solutions as zoom-in and zoom-out could be implemented, however we could not get these implementations done.

In the video (**p2-3-4**) we can also see the presence of black strips, which is caused by the affine model. A bilinear interpolation would solve the problem, but we did not have time to implement such resource. The transformation process and the video stabilization can be seen in the video on the output folder.