UNIVERSITAT DE GIRONA

VISUAL PERCEPTION

LABORATORY REPORT

Applications of Invariant Features

Author
Mohit VAISHNAV

Supervisor
Dr. Rafael GARCIA

Submission Last Date 12th, May

Contents

List of Figures

1 Introduction

Any object can be identified by extracting its most important feature. For any classifier it has to detect an object with a great accuracy, all the features are supposed to be detectable under various changes like scale, noise or illumination etc. Edges were good candidates for such a points. Harris Corner were one of the most used technique to detect feature for a long time which was invariant for rotation. But this has a problem when scaling comes into picture as can be seen in Fig. 1. Later **D. Lowe** from University of British Columbia came up with a paper *Distinctive Image Features from Scale Invariant Keypoints*. It has property to robustly identify under illumination change, partial occlusion, orientation, scaling and even to affine distortion.

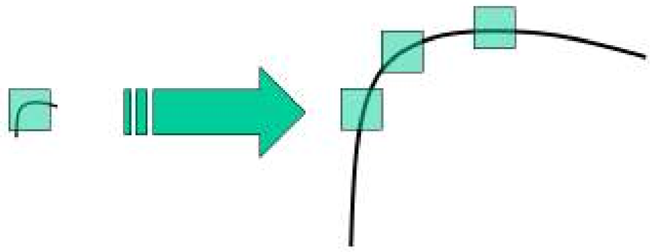


Figure 1: Corner Detection (source:OpenCV.org)

2 Algorithm

It could be in general outlined in the following steps Fig. 2 :

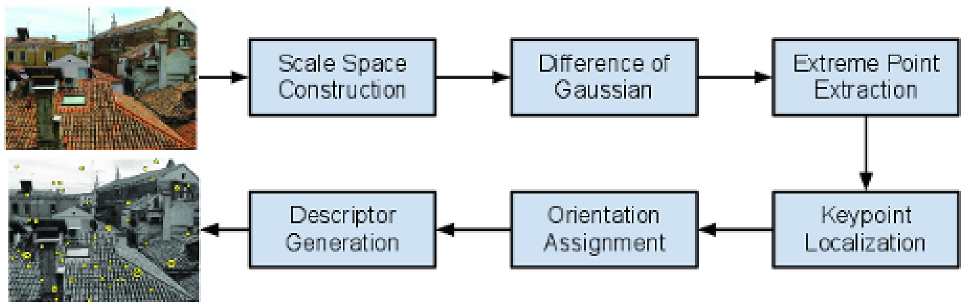


Figure 2: Pipeline (source:Google images)

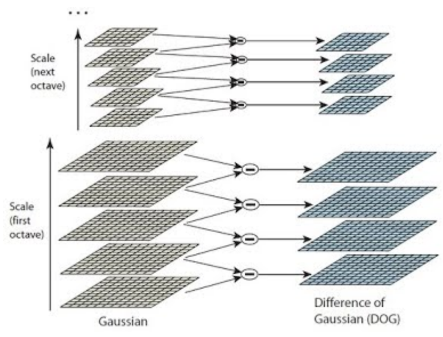
- *Constructing a scale space*: To ensure scale invariance, scale space is created representing original image.
- *LoG Approximation*: Laplace of Gaussian is used for finding keypoints in an image.
- *Finding Keypoints:* They are the maxima and minima in the Difference of Gaussian image calculated in above step.
- *Get rid of bad keypoints*: Usually edges with low contrast regions are called bad keypoints.
- *Assigning an orientation to the Keypoints*: For each keypoints, an orientation is calculated and all the calculation are done relative to it. This is useful in making it rotation invariant.
- *Generate SIFT Features*: Once rotational and scale invariance is tackled unique features are identified which are used in all SIFT applications.

2.1 Scale Space Extrema Detection

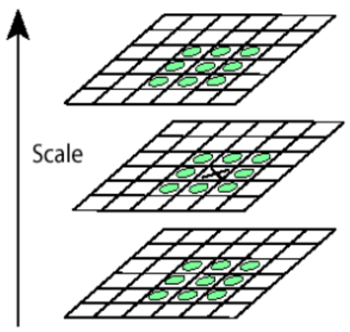
It is the first step towards detecting obtained by convolution of the image with Gaussian filters at different scales (Fig. 3a) leading to formation of blurred adjacent images. Interest points are identified as local maxima or minima of DoG wrt to 26 neighbours as seen in Fig. 10b across all the scales. For each keypoint, interpolation is done to estimate exact position, low contrast values and those which are along the edges are omitted and finally it is assigned an orientation using gradient orientation Histogram is used where peak in it corresponds to the dominant orientation. If any other values comes out to be in range of 80% of the maximum value, a new orientation is created and hence the keypoint.

2.2 SIFT feature representation

Once orientation is known, feature descriptor is estimated on 4×4 neighbourhood as set of orientation histogram. It contains 8 bins each having an array of 4 histogram around the keypoint. Hence we have the SIFT vector of size $4 \times 4 \times 8 = 128$ elements. To nullify the illumination this vector is normalized.



(a) Difference of Gaussian



(b) Nearest 26 neighbours

Figure 3: Comparison of DoG and Neighbours

2.3 SIFT feature matching

From the training images, nearest matching is searched wrt database of SIFT features database. Euclidean distance is the chosen method for comparison purpose which is also fast.

2.4 Recognition using SIFT features

For any input image, SIFT features are computed which are to be matched from the database of features. Each keypoint is composed of 4 parameters like location, scale and orientation. Hough transform is used to increase the robustness for identifying clusters matches. Each keypoint votes for each object pose consistent with the location, scale and orientation. Below mentioned Fig. 4 is the summary of this.

3 Analysis

This lab requires to explore various aspects of SIFT properties starting from the implementation part. First the task was to get used to the software that David Lowe created and was available on the website for all the references and testing purpose. But because of the 32 bit version availability, it was impossible for anyone to run/use it on the current system which are commonly of 64bit architecture. Now the task became to avail the next possible option for implementation of SIFT which was suggested to be *vl_feat* [6].

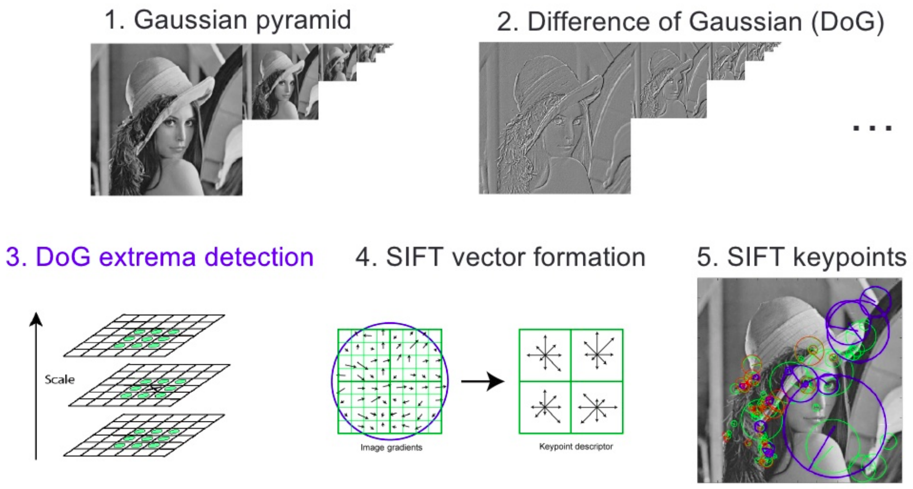


Figure 4: Summary (source:Google images)

VLFeat is the opensource to implement various vision based algorithms with special focus on feature extraction and matching. Some of the most important algorithm includes VLAD, MSER, Heirarchical K- Means, SLIC Superpixels, large scale SVM training, SIFT etc. Though written in C, it has interface in MATLab with support in Windows, Linux and MAC OS X. Version used for the research purpose is *0.9.21*.

3.1 vl_shift

Input for this command is a single precision gray scale image normalized in the range of [0,255].

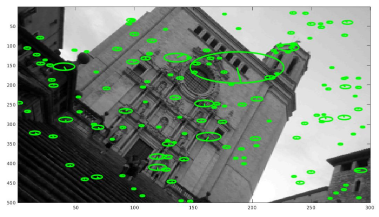
$$I = single(rgb2gray(I))$$

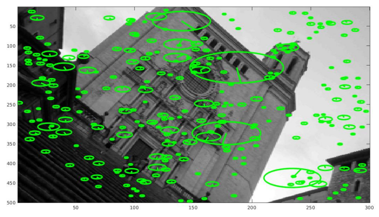
Then the keypoints and descriptors are obtained using the command:

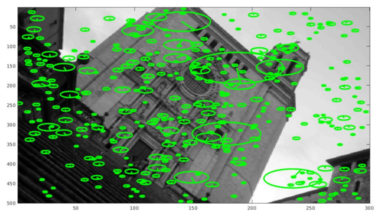
$$[keypoints, descriptors = vl\_feat(I)]$$

Keypoints are are dimension (1×4) in a column format where the first two values represent center of the disk, next is for the scale and the last relates to the orientation. At any particular point and orientation too a descriptor can be seen (Fig. 7).

Now the beginner step is to take randomly 50 keypoints and then plot them. *vl_plotframe* is used for that purpose where the keypoints acts as input. Along with the keypoints, descriptors can also be visualized using


(a) Threshold 3


(b) Threshold 5


(c) Threshold 7

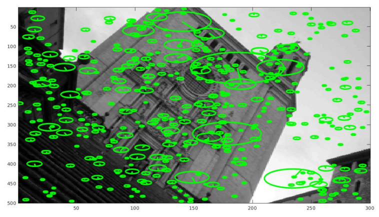

(d) Threshold 10

Figure 5: Keypoints Variation with Edge Threshold in vl_sift

the function *vl_plotsiftdescriptor* as seen in Fig. 8. Two parameters which could be twisted in *vl_sift* function is *peak threshold & edge threshold*. While *peak threshold* filters peaks of the DoG scale which are too small, *edge threshold* eliminates small curvature peaks in a DoG scale. Their variation can be seen in Fig. 5, 6. It is seen that as the *Edge Threshold* increases number of keypoints detected also increases while in case of *Peak Threshold* as it is increased number of keypoints decreases for the obvious reasons.

Once keypoints and descriptors are in place, one would like to find the best matches. To obtain the best match between any two set of images, it is required to first find their keypoints and descriptors. They are to be compared using another inbuilt function from the VLFeat library function, *vl_ubcmatch* where both the descriptors outputs acts as input and results in form of matches and scores. L2 norm is used to find the scores while comparing these descriptors. Another parameter which could be altered is *threshold* which if increased changes the number of matches. It represents the ratio of distance between best matching keypoint and the second best one. After matching them to know the best matches, scores are arranged in decreasing order and their corresponding matches are viewed on the image using the above mentioned command (Fig. 9).

$$[matches, scores] = vl\_ubcmatch(descriptor1, descriptor2)$$

3.2 Part II

In this section, it has been investigated how SIFT descriptors match with the variation in various parameters like brightness offset, adding noise or by blurring of an image. Ideally they are not suppose to vary much with these variations. Lets take an example image and plot their corresponding keypoints as seen in Fig. 10.

Fig. 11a hows how repeatability varies as the noise standard deviation is increased. There is a slow but steady drop in the repeatability for larger noise levels. Since the image is low-pass filtered with a Gaussian kernel in generating the scale-space, the noise is reduced before the image gradients and feature descriptors are computed.

The following plot in Fig. 11b shows how repeatability varies as the g value in the contrast adjustment is changed. Like for small brightness offsets, the SIFT descriptor only changes very slightly when the contrast is adjusted. The SIFT descriptor is normalized to have unit L2 norm, so most effects of contrast changes are removed during the normalization.

In plot 11c shows how repeatability varies as the brightness offset changes. For small brightness offsets, the SIFT descriptor does not change, because the descriptor is formed from image gradients which are unaffected by a brightness offset. The reason why some of the descriptors change for larger brightness offsets is that the gray values are saturated when they reach 0 or 255 in an 8-bit image representation.

In Fig. 11d repeatability varies as the standard deviation of the Gaussian kernel used for blurring is increased. Since large amounts of blurring significantly alter the gradients and consequently the histograms of gradients, the SIFT descriptor changes noticeably as the amount of blurring increases. Of the four types of distortions tested, blurring seems to have the most serious effect on the SIFT descriptor.

3.3 Part III

While looking at the various applications of SIFT in terms of assignments given in different universities across the globe, this came to be interesting ones worth mentioning and implementing. A dataset of few images in a scenario of poster presentation. Snaps are collected from each place having posters and authors along with the individual posters. Now the task is to match individual poster to the one from the database having the same along with their corresponding author. To execute the task, following steps are to be followed:

Step 1 : Extract SIFT feature from the input image using vl_sift.

Step 2 : Match the query image's SIFT features to every database image's SIFT features using nearest-neighbor search with a distance ratio test as implemented in vl_ubcmatch.

Step 3 : From the feature correspondences that pass the distance ratio test, find the inliers using RANSAC with a homography as the geometric mapping.

Step 4 : Report the database image with the largest number of inliers after RANSAC as the best matching database image.

Below shown Fig. 12a is the query image and the best matching database image, Fig. 12b represents the SIFT features extracted from both images, Fig. 12c is the feature matches after nearest neighbor search with a distance ratio test, and Fig. 12d is the feature matches after RANSAC with a homography.

4 Part IV

Now, in-depth understanding on Fundamental Matrix is focused in this section. First lets discuss about the *Essential Matrix*. It relates corresponding image points between both cameras, given the rotation and translation. These points are in Camera Coordinate System. Moving ahead to the *Fundamental Matrix* which relates pixel coordinate in two views. It is more general form of the essential matrix as there is no need to know intrinsic parameters. If we estimate fundamental matrix from correspondences in pixel coordinates, can reconstruct epipolar geometry without intrinsic or extrinsic parameters. Mainly it is used for rectifying the stereo image calibration. It is used to solve the correspondence problem in a uncalibrated system of cameras.

estimateFundamentalMatrix estimates the fundamental matrix from corresponding points in stereo images. This function can be configured to use all corresponding points or to exclude outliers which is done by using a robust estimation technique such as random-sample consensus (RANSAC).

When robust estimation is used, results may not be identical between runs because of the randomized nature of the algorithm. Matches for providing input to any Fundamental Matrix is computed using two kind of techniques, Harris Feature Detection (*detectHarrisFeatures*) as shown in Fig. 13 and SURF Feature Detection (*detectSURFFeatures*) represented in Fig. 14. Best matches are found out using MATLab inbuilt function *points.selectStrongest(n)*. They are given input to the *estimateFundamentalMatrix* and they are matched to the relevant image as seen in Fig. 15.

4.1 Part V

As a last part of the experiment, focus is on obtaining a fully functional *Mosaic* of any number of images. This is one of the widely used application in every smart device. The following steps are to be taken to get the desired application:

- Load any two images to be stitched, convert them into single precision gray image.
- Detect features in both the images using VLFeat
- Compute the distance between every descriptor in one image to every descriptor in another image.
- Select all the pairs whose descriptor distance are below specified threshold and choose the least 100.
- Run RANSAC to estimate homography mapping mapping one image onto another. Figure also displays the average inliers and the location of matches in both the images.
- Using the estimated transformation map one image onto another.
- Create a big image to hold both these images in a single frame with the region of overlap.
- The above steps work on gray-scale image but could work the same for the RGB too.

Resulting image can be viewed in Fig. 16 where the input images can be seen in Fig. 17 with output in Fig. 18. As a part of aligning, images can be seen in Fig. 9 where keypoints are matched and plotted. Once found, the process followed is applying RANSAC and obtaining inliers (Fig. 19) and the final output in seen in Fig. 20.

5 Conclusion

Most interesting part of this is to learn and play with the SIFT features. SIFT has been a revolutionary yet simple implementation with numerous applications in vision. This practical experimentation is useful in learning the variation in features wrt various parameters like blurring, noise, contrast and see if the claim which is made in paper by the author. It is found that there is a change in number of matches between keypoints with those parameter variation. Although various implementations were experimented in this lab over various set of images but it came to the light that SIFT works best amongst all. In Part IV, output from Harris and SURF was computed on a image but the matches did not work out that properly like for the case of SIFT. While implementing the Fundamental Matrix, it can be said that for cases of uncalibrated system it could be used to get the correct correspondences and help in removing the outliers further. Other options tried is Image mosaicing which is a direct visible application to be related from this lab as a learning.

References

[1] Wiki,
`https://en.wikipedia.org/wiki/Scale-invariant_feature_transform`

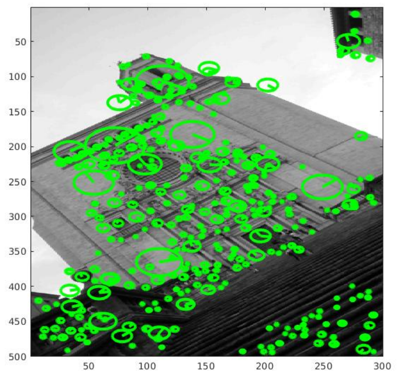
[2] OpenCV,
`https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html`

[3] Course Website,
`http://syllabus.cs.manchester.ac.uk/pgt/COMP61342/resources/SIFTtutorial.pdf`

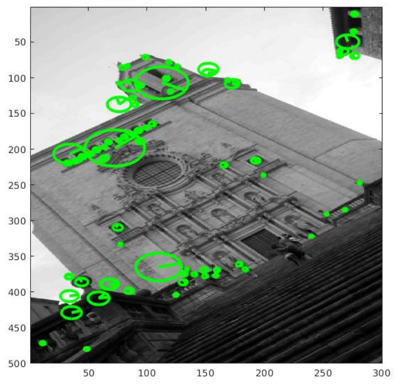
[4] Oxford VGG,
`http://www.robots.ox.ac.uk/ṽgg/hzbook/code/`

[5] VLFeat,
`http://www.vlfeat.org/install-matlab.html`

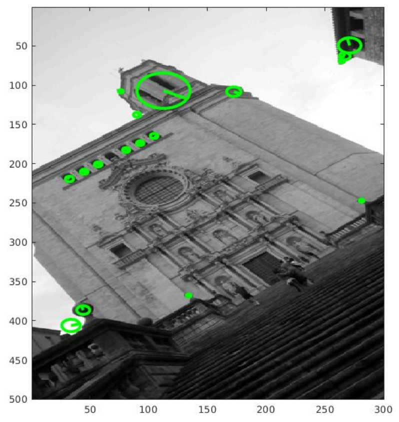
[6] Explanation,
`https://courses.engr.illinois.edu/cs498dwh/fa2010/lectures/Lecture%2017%20-%20Photo%20Stitching.pdf`



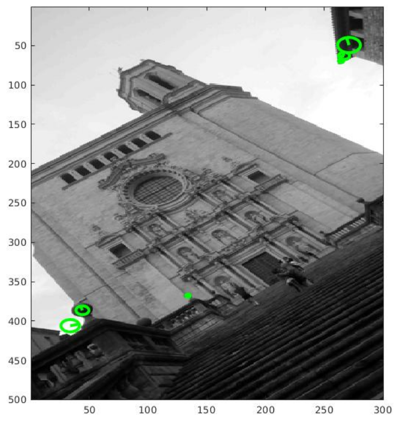
(a) Threshold .01



(b) Threshold .03

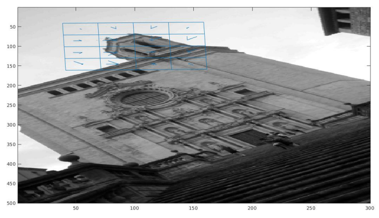


(c) Threshold .05

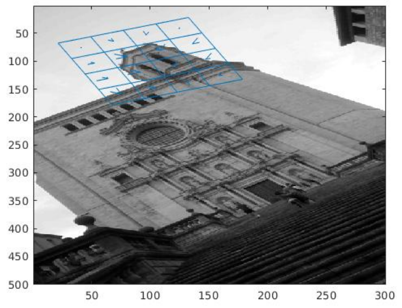


(d) Threshold .06

Figure 6: Keypoints Variation with Peak Threshold in vl_sift



(a) Location (100,100), angle 0°



(b) Location (100,100), angle $\pi/8$

Figure 7: Descriptor at a location and orientation

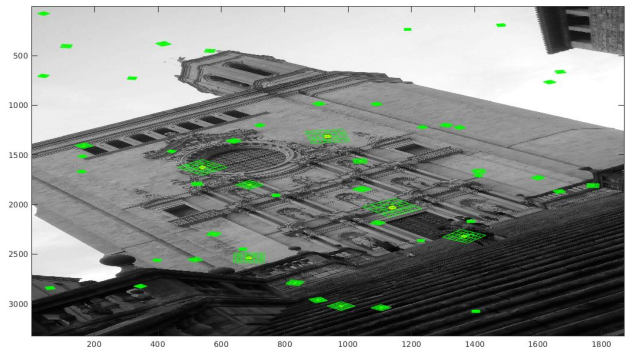


Figure 8: SIFT Randomly selected 50 Descriptors

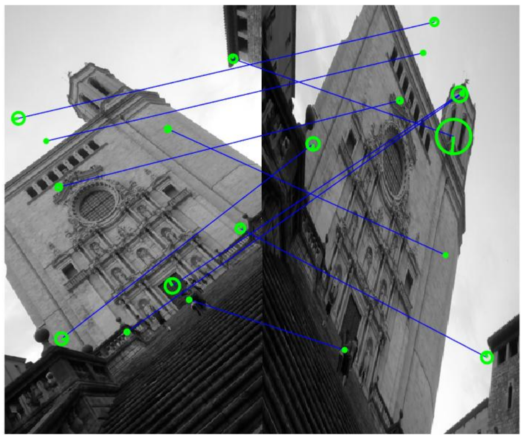
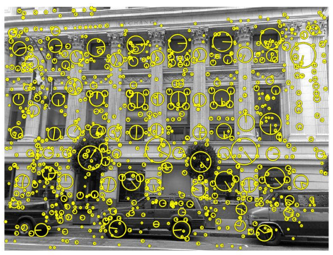


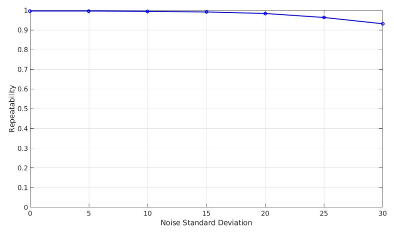
Figure 9: Best 8 matched between two images

(a) Image

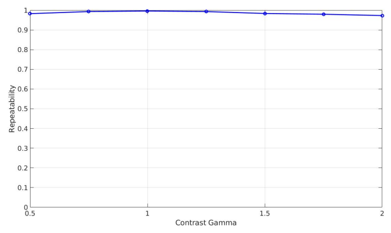


(b) Keypoints

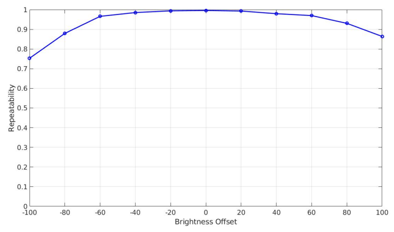
Figure 10: Describing the image and their keypoints



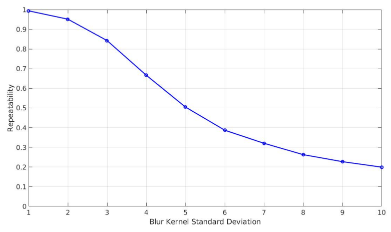
(a) Noise



(b) Contrast

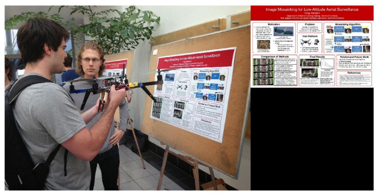


(c) Brightness

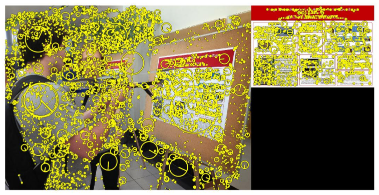


(d) Blurring

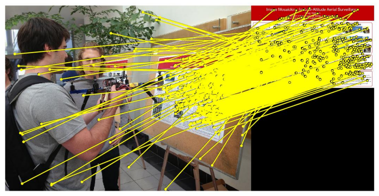
Figure 11: Variation in fraction of keypoints with various factors



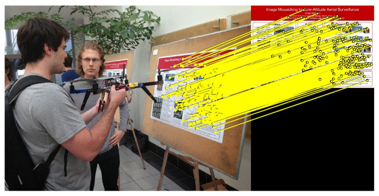
(a) Image



(b) De points



(c) Matching Keypoints



(d) Final Result

Figure 12: Finding the poster from a set of images

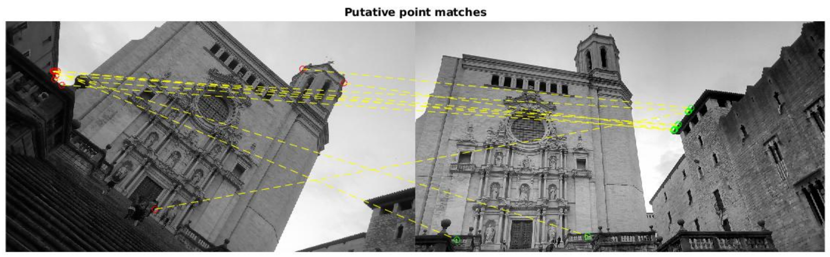


Figure 13: Harris Feature Detection

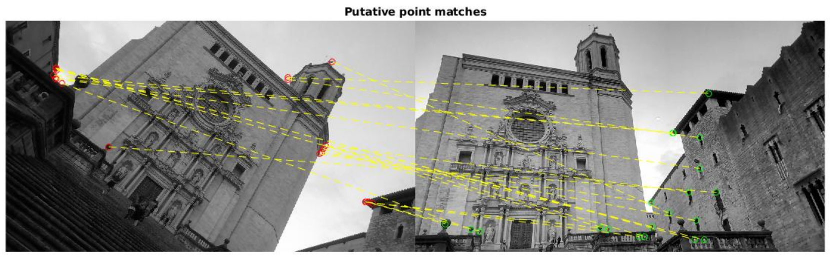


Figure 14: SURF Feature Detection

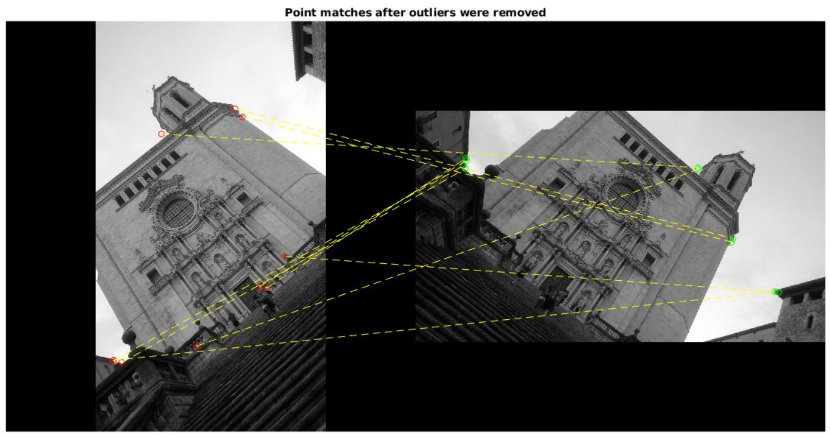


Figure 15: After applying Fundamental Matrix Calibration

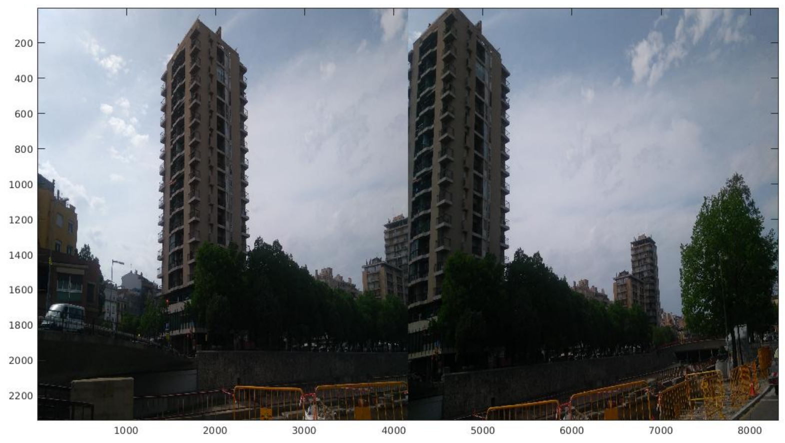


Figure 16: Two images to be Stitched

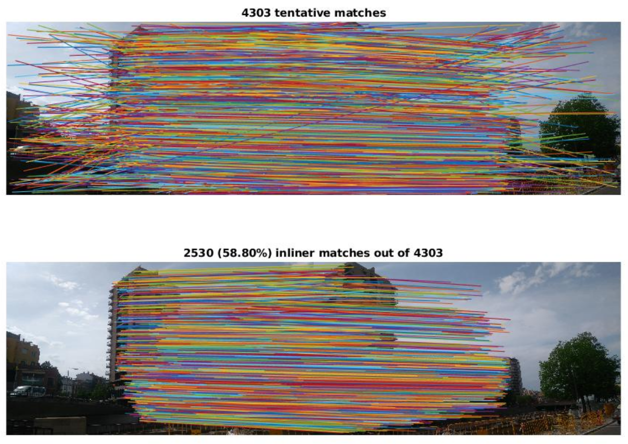


Figure 17: Matching for Mosaic

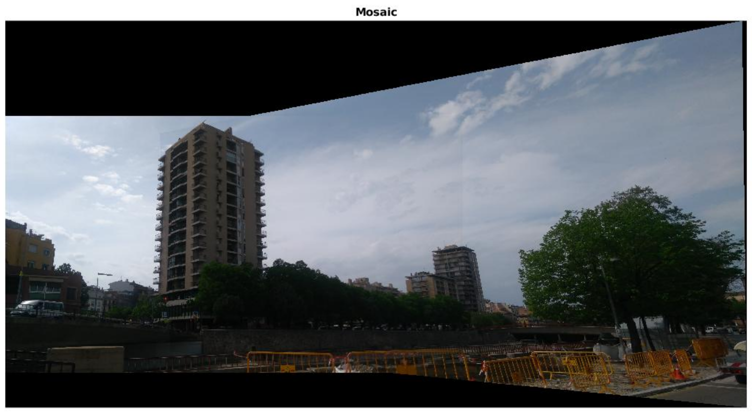


Figure 18: Resulting Image for Fig. 17

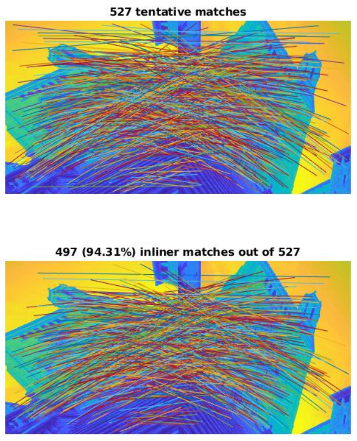


Figure 19: Matching for Mosaic

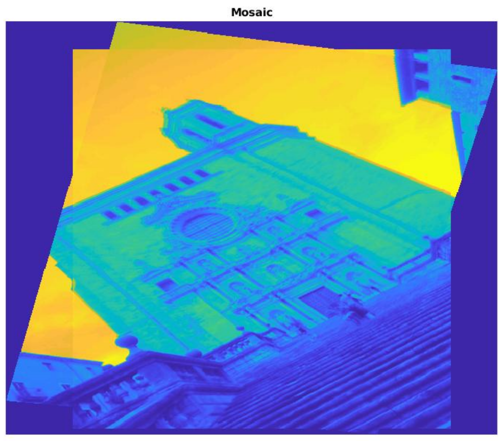


Figure 20: Aligning images for matching in Fig. 9