University of Science and Technology at Zewail City

Communications and Information Engineering Program

CIE 552: Computer Vision

Mini Project II - Local Feature Matching

Team Information

Team Name: Regina Phalange

Elsayed Mohammed Elsayed Mostafa ID: 201700316

Mohammed Magdy Alasmar ID: 201700038

Contents

Introduction

Feature detection and matching are core procedures in many computer vision applications like image alignment , object detection and recognition , 3D reconstruction and robot navigation. The goal of these procedures is to find unambiguous matches in other images. In this project, many algorithms that developed over years have been introduced to help find effective features between corresponding images to be matched. These algorithms include Harris corner detector technique and nearest neighbor test ratio (NNTR). To achieve the main goal of matching images, many tasks should be fulfilled from detecting , describing and matching features. Any defects or gaps that are left unaddressed by an algorithm, there is another more robust technique that handles this gap. Our implementation adapts a simplified version of SIFT, the difference is that it does not handle scale invariance as it is implemented on one octave for simplicity.

Algorithm

Local feature matching algorithm is mainly divided into 3 main parts: interest points extraction, SIFT-like features creation and matching technique. Interest point extraction is done using Harris corner detection algorithm, the algorithm seeks to obtain the points (pixels) representing the corners in the image. These corner points are used as unique features in the image. The next stage is to have a unique representation of these points which is done through SIFT-like features creation. The method applies SFIT algorithm for local features representation in a simpler way. It constructs a patch of pixels (usually 16 x 16 patch) around each point of interest. Every patch is further splitted into several pixel boxes (usually 4 x 4), for each box the gradient is computed for each pixel. From the gradient, the magnitude and the orientation of the gradient is extracted and then the orientations are categorized into 8 bins (0 : 360 degrees). The previous two steps are done for both images to be matched. The last step is to find the matched pairs by calculating the distance between a specific feature vector in the first image with all the feature vectors in the second image. The ratio between the least two distances is calculated and then compared to a specific threshold as specified by the nearest neighbors ratio test algorithm. If the ratio is less than that threshold -which is tuned- then the feature with least distance is considered a good match with the original feature form the first image. Finally, we end up having the matched features in the two images with specific confidence calculated using the threshold in the matching step which is greater for the feature that achieves the matching condition easier.

Implementation

Our implementation is straightforward for the three main functions: **get_interest_points**, **get_features** and **match_features**. The following is a brief description of each function and its parameters.

- *get_interest_points:* This function uses our function (apply_harris) to get the corners in the image passed. The corners after certain tunable threshold are the interest points needed and then their corresponding coordinates are returned.
    - *The tunable parameters:*
        - ksize: the Gaussian kernel size of the filter used
        - k: the weighting parameter between the determinant of Harris matix and its trace value.
        - w_size: the size of the square window used in applying Harris corner detection for each pixel.
        - The threshold value to consider each point after computing its Harris matrix a corner (interest point) or not, it is usually set to 0.05*r_max where r_max is the maximum value of the computed Harris matrix for the whole image.
        - min_distance, which is the distance between the neighbors taken into account at performing *non max suppression* on the values achieving the previous threshold.
- *get_features:* This function computes the feature vector for each point of interest got from *get_interest_point*

- *match_features:* This function checks the match between each two points in the two images to be matched. The matching is done using Nearest Neighbour Distance Ratio Test (NNDR). Also, the function assigns confidence value for each matching.
    - *The tunable parameters:*
        - matching_threshold: the threshold to compare the ratio between the two nearest neighbours for a specific interest point. If the ratio is less that this threshold then the matching is good and acceptable with (1 - ratio) as the confidence value.

Procedures and Results

The main evaluation criteria of the implementation is the accuracy of the most 100 confident matchings in ***notre dame*** image. The matching in the two other images (***rushmore mountain - Episcopal Gaudi***) are also considered but with less importance.

The first step was to test ***match_features*** function with random feature vectors for pre-defined (cheated) interest points. The accuracy of matching - which makes no sense - was around 10% in ***notre dame*** image. Secondly, the ***get_features*** function was implemented and tested where the features were only the normalized pixel values of a cell inside each patch i.e. 16 values for each cell. The accuracy on all matches after this step was 55%. Next, the ***get_features*** function was updated with the SIFT-like algorithm mentioned earlier. The accuracy on all matches increased to around 90%. Note that these values of accuracy were for the all matches not specifically for the most 100 confident matchings. The last step was to implement the ***get_interest_points*** function. After the function implementation, the stage of parameters tuning began. All the previously mentioned parameters were tuned to get the most possible accuracy on the most 100 confident matches, The tuning procedure is summarized in the following table (Table 1). The approach was to change one parameter at a time. When the number of interest points is little, the threshold (R) was decreased or the minimum distance was decreased and vice versa. For the same interest points number, the value of matching threshold was tuned to accept more/less matchings; the higher the value, the more matches are approved. The other values such as kernel size and window size were mostly kept constants as they have no severe effect on the accuracy as tested in the initial tests.

R Threshold	min_distance	matching_threshold	*50 - Accuracy*	*100- Accuracy*	*All- Accuracy*	*# matches*	*Time consumed (Minutes)*
0.05	20	0.9	***90***	***71***	***63***	***213***	***3***
0.05	20	0.8	***-***	***-***	***89***	***47***	***3***
0.05	20	0.85	***90***	***71***	***68***	***109***	***3***
0.03	20	0.9	***96***	***88***	***74***	***421***	***5***
0.01	20	0.9	***-***	***-***	***-***	***-***	***TIME EXCEEDED***
0.05	40	0.9	***92***	***71***	***65***	***206***	***3***
0.03	20	0.8	***96***	***-***	***90***	***86***	***7***
0.03	10	0.8	***96***	***-***	***90***	***86***	***6***
0.02	20	0.8	***100***	***95***	***93***	***140***	***10***
0.02	20	0.9	***100***	***95***	***79***	***670***	***5***

Table 1. Parameters Tuning for notre dame image matching.

As a major parameter to take into account, the time consumed was estimated for each trail to avoid having a good accuracy with high time consumption. After these trials, the last two in Table 1 were the best results obtained for the accuracy on the most 100 confident matchings; however, the last parameters are preferred for its lower time consumption. The matching result is shown in figure (1).

For the Mount Rushmore image, a similar approach was followed to gain best results after hyperparameters tuning to achieve most effective time-resources tradeoff as shown in Table 2.

R Threshold	min_distance	matching_threshold	*50 - Accuracy*	*100- Accuracy*	*All- Accuracy*	*# matches*	*Time consumed (Minutes)*
0.05	30	0.9	***90***	***83***	***74***	***158***	***3***

Table 2. Parameters Tuning for Mount Rushmore matching (bestt trial)

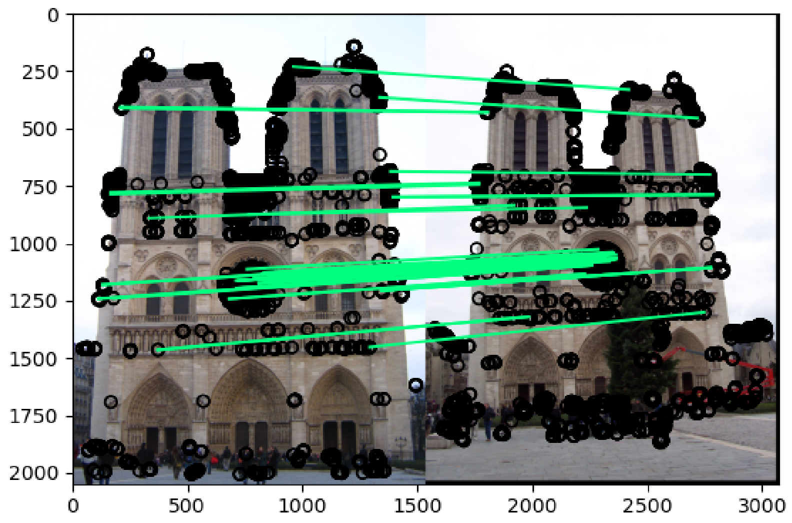

Figure (1). The matching result for notre dame image.

Finally, matching ***Episcopal Gaudi*** image did not achieve good results at all as shown in the best trial parameters in Table 3. This seems to be logical as our implementation is not scale invariant and invariance in this example was too apparent to be handled by parameter tuning.

R Threshold	**min_distance**	**matching_threshold**	***50 - Accuracy***	***100- Accuracy***	***All- Accuracy***	***# matches***	***Time consumed (Minutes)***
0.08	20	0.9	***-***	***-***	***10***	***30***	***3***

Table 3. Parameters Tuning for Episcopal Gaudi matching (last trial)

Note **

Most of the tests (not all of them) were run on a laptop (HP Pavilion x360 Convertible 14-cd0xxx) with the following specifications:

- Processor : Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
- RAM : 8 GB Samsung 2400MHz
- Dedicated GPU : NVIDIA GeForce MX130