Final Project Report

Computer Graphics

Image Matching System

Submission Date: 2017.1.6

Group15 Members:

Name	Student ID
Xiaoqing Geng	5140309383
Rulin Huang	5140379001
Chao Gao	5142029014

Prepared by: Anum Masood (Course TA)

Table of Contents

Group Members:

Name	Student ID	Tasks Assigned
Xiaoqing Geng	5140309383	Related work studying Coding of proposed idea Report documentation
Rulin Huang	5140379001	Coding of proposed idea Implementation Presentation
Chao Gao	5142029014	Understanding the given Research Paper Implementation Report documentation

1. Introduction

Here we use the technique for extracting local features from images of architecture scenes, based on detecting and representing local symmetries. We have the horizontal, vertical and rotational symmetry detections. For the feature points extracted in two images, the Euclidean distance indicates the possibility of match. And we also take the location of feature points into consideration. Besides, we add accuracy display and add the lines between feature points to make match intuitive. We demonstrate our method on the dataset containing image pairs exhibiting a range of dramatic variations in lighting, age, and rendering style, and show that we have good performance for this difficult task.

1.1. Related Work

Researchers in computer vision have made significant progress in representing and detecting symmetries in images and other types of data (Liu et al. [4] present an excellent survey). However, there has been relatively little work on using local symmetries as explicit features for matching tasks.

In the work of Daniel Cabrini Hauagge and Noah Snavely[1], they presented a method of using symmetry features as detector and descriptor to match images of urban constructions. Our new approach is based on the method of this paper, but takes a larger variaty of symmetry features into consideration and made other improvements. Schindler et al. [7] also detect and match repetitive patterns on building facades, and Wu et al. [8] detect large repeating elements of facade images for use as robust support regions for computing features. Loy and Zelinsky also propose a feature detector based on radial symmetries [5], using a fast radial symmetry transform that accumulates votes for symmetric regions from gradient information. However, this approach is applied to find symmetric objects in scenen, not to match images. In an earlier approach, Reisfeld et al. [6] used a similar voting scheme to detect interest points using radial symmetries via a generalized symmetry transform.

Our local symmetry detector is also related to other lowlevel measures of symmetry. Kovesi observed that local bilateral symmetry in image intensity relates to the response of filters of different phase [2], and later extended this concept of “phase-congruency” for detecting features such as edges and corners [3]. Similiar like that, our approach is to consider sparce local feature (like HOG) together with densly calculated symmetry features.

1.2. Proposed Work in Assigned Research Paper

In the work of Daniel Cabrini Hauagge and Noah Snavely[pro], they presented a method of using symmetry features as detector and descriptor(called SYMD) to match images of urban constructions.

They densly calculate the symmetry score of each pixel based on intensity(SYM-I) and HOG feature(SYM-G) separately. And for both of them, calculate each pixel’s vertical symmetry score, horizontal symmetry score, and rotational symmetry score. (That is to say, SYM-I score includes SYM-IV, SYM-IH and SYM-IR scores. It is the same with SYM-G.)

The method to calculate these scores:

- Given a symmetry type ***s***, and a point ***p*** at which we want to compute a symmetry score, we define a function $M_{s,p}$ that maps an image point $q = (x, y)$ onto its symmetrically corresponding point with respect to the point of interest ***p*** and the symmetry type ***s***.
- Calculate either intensity based or HOG features(denoted f) of each pixel. Define a function $d(x, y) = \| f(x) - f(y) \|$, to represent the difference of a feature value of two pixels ***x*** and ***y***.
- For each pixel p in the image, define a function called local symmetry distance: $SD(p) = \sum_{q} \omega(\| q - p \|) \cdot d(q, M_{s,p}(q))$. Here ω stands for weight, which is derived from Guassian Function.

After calculating these scores, they pick ***k*** pixels with the highest symmetry scores as keypoints, and use the symmetry scores of these keypoints to do a matching.

The matching function is based on arithmetic of vectors and metrics to calculate an overlap score. They claim a match as long as the overlap score is more than 0.6.

1.3. Limitations of Proposed Work

In the paper, of course there are some things we can improve.

- The author traverses every pixel of the image and comparing it with all other pixels. It is a little bit time-wasting and useless. We can traverse each pixel ***p*** of the image but only compare it with pixels nearby, of which ***p*** is the center(patch). In our work we add the function of patch matching with a phase of finding ROI.
- In the paper, characteristic points are the points whose symmetric scores are above maximum score times ***a***, where ***a*** is given by the author. We can get the properest value of ***a*** by our own training method.
- The author does not care about the location of feature points. We have an idea that the feature point in the top can not be a match with a feature point in the bottom.
- In the paper, the author only uses the rotational symmetry features(SYM-IR) without horizontal and vertical symmetry features, they don't build detectors that leverage more types of symmetry. This can be improved.

1.3.1 Critical Review:

Here we present the analysis and evaluation of the whole idea of the research paper.

- The idea of the paper:

The main contribution of the paper is a simple method for densely characterizing local symmetries across image and scale space and defining a corresponding local symmetry feature detector and descriptor which are used to find correspondence between difficult image pairs rich in symmetries.

By demonstrating their method on the dataset containing image pairs exhibiting a range of dramatic variations in lighting, age, and rendering style, and show that the features can improve matching performance for this difficult task.

- Analysis and evaluation:

In our opinion, the critical point of the paper is extracting local features based on detecting and representing local symmetries. This idea is the heart of the paper. These new features, which are motivated by the fact that local features are the fundamental characteristic of many urban

images, are potentially more invariant to large appearance changes than lower-level features such as SIFT.

For local symmetries, what we can improve is to add horizontal and vertical symmetry detection instead of only using rotational symmetries.

What's more, a direction detection function is needed, the points located at the top are not matched with the points located at the bottom to ensure that the relative position of the matching is fixed, thus avoiding the nonsense matching of the feature points which causing the wrong matching.

Characteristic points should be stricter here not only above the threshold. We also consider the second highest feature point, if their ratio is larger, then the match is better. This method helps us to decrease the noise.

2. Proposed Research Work

We propose a novel approach to match two images of urban constructions. Motivated by the work done by Daniel Cabrini Hauagge and Noah Snavely[1], we find that symmetry feature is accurate and novel in the field of image matching. Based on their work, our approach is to use a wider range of symmetry features, and implement a more powerful matching function, which is completely different with the previous work. We also added in patch matching in our work.

In our approach, the process of matching two images is divided into 3 phases: find region of interest(ROI), extract local symmetry features and get the charateristic points, and do matching with the characteristic points.

The phase of finding ROI isn't mentioned in the work of the provided paper at all. However, we add this phase to enable our work to do patch matching. So that even if the aim construction is at a corner of the whole image, we can do the perfect matching, too.

As for the phase of extracting local symmetry features, different from the previous works, we first use HOG to calculate the value of each pixel, and then calculate symmtry scores based on the HOG values. The symmetry features are extracted based on these features. Symmetry features are extracted in the same way with the proposed work in the provided paper, but Then we pick top $\boldsymbol{k}$ pixels with the highest symmetry score as characteristic points.

As for the phase of matching function. Our matching function is based on BF(brutal force) matching but made improvement to it. We do ratio test on the characteristic points and symmetry test on the original matchings. Then we claim that if point $\boldsymbol{A}$ of an image is matched to point $\boldsymbol{B}$ of another, but $\boldsymbol{A}$ is on the top(or left) of the ROI which it is in, while $\boldsymbol{B}$ is on the bottom(or right), then the match between $\boldsymbol{A}$ and $\boldsymbol{B}$ is invalid. Because the datasets are all constructions of urban areas, this screening functions contributes to a better result.

Finally, our work can achieve a better acuracy than the paper provided.

3. Methodology

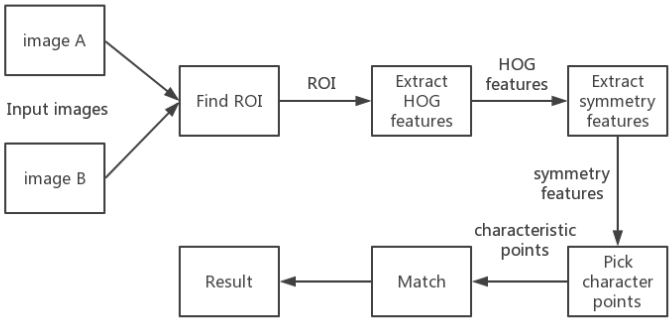

Figure 1: Flow chart of the implementation steps.

As is shown in figure 1, our implmentaion is divided into 6 steps.

At the beginning we introduce two sets of images which are needed to be matched. We can automatically do the batch processing through the program. Any two of them or determine one of the image can be matched.

Then we use the algorithm to find the ROI in the image. Since the image may require batch processing, it is easy to find the desired results in other images by determining the ROI after determining which primitive image needs to be found. Because of the ROI, we can even perform local matching in other images.

Then we obtain the corresponding value of each pixel by HOG feature, and then calculate the symmetry of each pixel on this basis, in order to obtain the eigenvalues of each pixel.

After obtaining the eigenvalues of each pixel, we pick the first ***K*** pixels as the feature points according to the requirement.

In the process of matching, we create a matrix, through the establishing the relationship between image ***A*** and image ***B***, then we extract the characteristics of the two points according to their eigenvalues to obtain a corresponding weight. The smaller Euclidean distance is, the higher the matching is, so we determine the degree of correlation between the two images by means of violent matching. At the same time, we also increase the relevance of the feature points in the position to reduce the possibility of error matching.

Finally, based on their specific gravity in all the feature points and the number of these feature points associated with, we confirmed the associated feature points. The corresponding precision results are output as numerical values.

4. Implementation

The main processes and sudo codes are as follows.

- Find ROI.

```
Func find_ROI
{
   Do binaryzation on source image.
   Smooth the binary image with cvSmooth.
   Extract connected components by tracking contours.
   Return bounding boxes of ROIs.
}
```

- Extract HOG features.

  Func HOG
  {
  Divide the grayscale image into 2×2 cells.
  Calculate the gradient of each pixel in each cell.
  }

- Extract symmetry features.

  Func SYM
  {
  for each kind of feature, including horizontal, vertical, and rotational
  {
  for each pixel ***p*** in the image
  {
  for each pixel ***q*** in the image
  {
  Find pixel M_p which is symmetrically corresponding point of ***q*** with respect to the position of ***p***.
  Use Gaussion Function to calculate the weight of ***q***, which is determined by distance of ***q*** from ***p***.
  Calculate the absolute value of HOG(***q***)-HOG(M_q), and multiply it by its weight.
  }
  The mean value of is regarded as the symmetry score of **p** of a certain symmetry type.
  }
  }
  }

- Pick characteristic points.

  Func get_points
  {
  Return the first ***k*** points with the highest 0.7×(SYM-H × SYM-V) + 0.3×SYM-R score.
  }

- Matching function.

  Func matching
  {
  Construct two vectors of matches called vectA and vectB.
  Construct a vector of matches called VECT.
  Call ratio_test.
  Call symmetry_test.
  Call pos_test.
  Return matches.
  }
  Func ratio_test
  {
  for each characteristic point x in image A
  {
  for each characteristic point y in image B
  {
  Calculate the Euclid Distance between x and y.
  If the distance is smaller than a set ratio, then add to vectA.
  }

```
    }
    for each characteristic point x in image B
    {
        for each characteristic point y in image A
        {
            Calculate the Euclid Distance between x and y.
            If the distance is smaller than a set ratio, then add to vectB.
        }
    }
}
Func symmetry_test
{
    for each element in vectA
    {
        if the element is also in vectB
        then put it into the vector VECT.
    }
}
Func pos_test
{
    for each element in VECT
    {
        if the two point which are matched are in quite different scope of the images, for
example, one is on the very left part of the ROI it is in, while the other is on the very right
position of the ROI it is in, then delete it from VECT.
    }
}
```

- Test.

```
Funct test
{
    for each image in folder A
    {
        for each image in folder B
        {
            Calculate the number of matches between the two images
        }
    }
    If the number of matches between 2 images is larger than 1/30 of the numbers above, then
claim a match.
}
```

5. Results

In our exprimental evaluation of our method, we compare the result of the returned value and the correct value from the dataset. The evaluation is done on Ubuntu16.04LTS. In this section we will show that our experimental result exceeds the result of the work in the provided paper.

5.1. Detector evaluation

We do tests on the database which is provided by the author of the paper provided. In their dataset, the images are arranged in the form of pairs, which are of the same construction, either photographs or paintings.

For each image pair ***(I1, I2)*** in the dataset, and each detector, we detect sets of keypoints ***K1*** and ***K2***. A common repeatibility metric is the fraction of keys in ***K1*** that have a similar detection in ***K2***. However, this measure is biased towards detectors that produce large numbers of keypoints. To alleviate this problem, we compare subsets of detections of varying size ***k***, selected according to some "goodness" criterion (in our case, detector response or feature size), rather than the full sets of detections. In particular, for each ***k*** we compute the fraction of the top-***k*** detections in ***K1*** with a similar detection in the top-***k*** detections in ***K2*** (1.0 is a perfect score). By varying ***k*** we can produce a curve showing repeatability for all sets of top ***k*** features(figure 2).

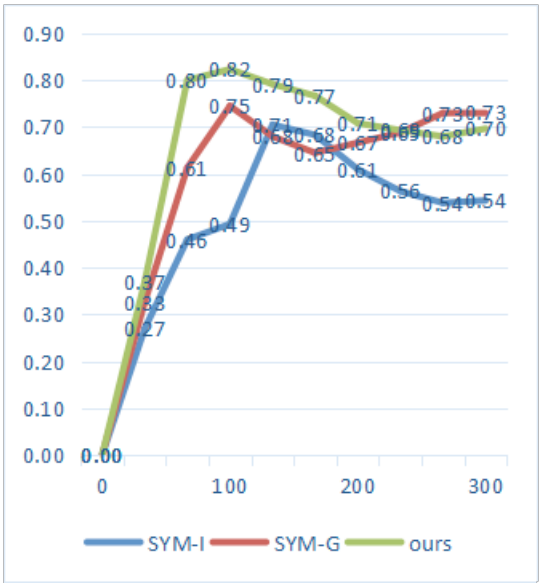

Figure 2: Example results for the detector repeatability experiment for two pairs of images (shown as insets), for each detector, SYM-I, SYM-G(from the provided paper) and our detector. The horizontal axis shows the number of features ***k*** in the selected subsets, and the vertical axis shows the fraction of keypoints that have a repeated detection (higher means better repeatability).

5.2. Descriptor evaluation

In the evaluation of descriptor, for each pair of images we extract keypoints in, we set different ratio values and get the corresponding accuracy and recall. Then we can produce a curve showing the relationship between accuracy and recall(figure 3).

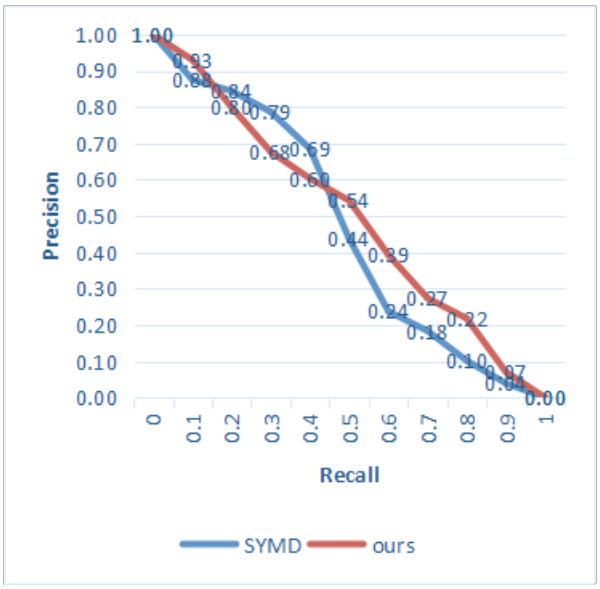

Figure 3: Precision-recall curves for a feature matching task on selected image pairs. Comparing SYMD(from the provided paper) and our descriptor.

5.3. Running result

The running screeshots are shown in the follows. Here we placed screenshots of two images that are of the same instruction(figure 4), two images that not are not of the same instruction(figure 5) , and an example of patch matching(figure 6).

We define a testing function, which combines both precision and recall together. The function is that if and only if two images of a construction are matched by our system then we claim that our result is accurate, otherwise it is not. Under this function, our method can achieve an accuracy of 95.7%, while the function in the provided paper only reaches an accuracy of 88.5%.

Figure 4: An example of matching between two images which are of the same construction. (The number of matches is quite large.)

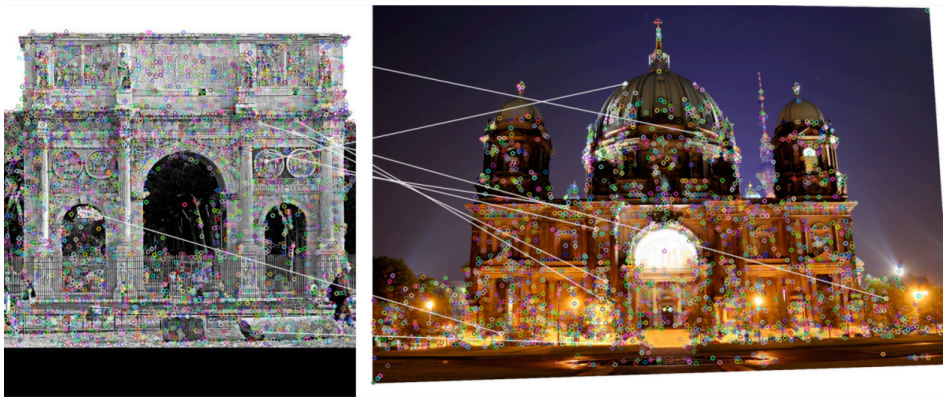

Figure 5: An example of matching between two images which are not of the same construction. (The number of matches is quite small.)

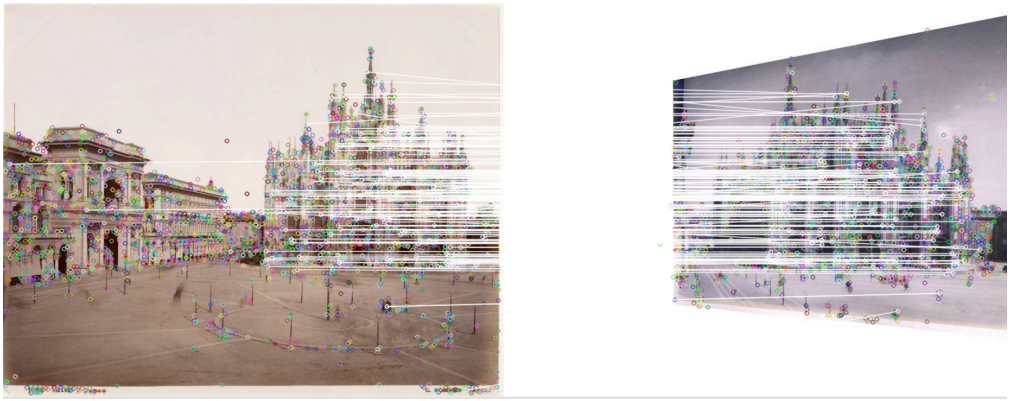

Figure 6: An example of patch matching. (The two images are of the same construction and the number of matches is quite large.)

6. Conclusion & Future directions

6.1. Conclusion

In this paper, we propose a novel method of image matching system based on local symmetry features. We not only improved the method in the paper provided but also implemented a new matching function. And the result of our work supports our objective to surpass the existing research work.

Our findings are similiar with the paper provided. Symmetry features can be widely used and are of high efficiency when it comes to matching two items which are of highly symmetric, such as urban buildings. This feature may not apply to many areas, but in the specific area, it work well. However, we find that not only the rotational symmetry features are meaningful, but the horizontal and vertical features can be exploited, too. This may because most of the buildings are highly horizontal or vertical symmetric. In our work, we also applied horizontal and vertical symmetry features to gain better results.

The limitations of our work is that we don't have a very perfect dataset. The dataset provided by the author is quite small, and the photographs in it are all clear although some may be photoed tens of years ago. This may restrict us from improving our method to make our system handdle more conditions.

All in all, we have improved the method of extracting symmetry features and implemented our matching function which is more reasonable. We can now draw a conlucsion that image matching based

on symmetry features works efficiently and accurately in the specific field of matching symmetric items, and horizontal as well as vertical symmetry features are worth exploring in future.

6.2. Future directions

Although we have achieved a good result, we can still improve our work in some aspects. Firstly, we can reduce the time latency of calculation, especially the time of symmetry feature extraction. Secondly, we can further explore the usage of horizontal and vertical symmetry features, and add other efficient algorithms in. Thirdly, we can build up a friendly interface to better present our work.

Our work can not only help with image matching, but also work in other extended areas. These areas may include object detection, photo restoration, and even 3D modeling(with patch matching). We can draw a conlusion that our work can be applied widely in many aspects.

7. References

[1] Daniel Cabrini Hauagge and Noah Snavely. Image Matching using Local Symmetry Features. Cornel l University, 2012

[2] P.Kovesi. Symmetry and asymmetry from local phase. Australian Joint Conf. on Artificial Intelligence, 1997.2.

[3] P.Kovesi. Image features from phase congruency. Videre: Journal of Computer Vision Research, 1992.2

[4] Y.Liu, H. Hel-Or, C. Kaplan, and L. V. Gool. Computational Symmetry in Computer Vision and Computer Graphics. Foundations and Trends in Computer Graphics and Vision, 2009.2

[5] G. Loy and A. Zelinsky. Fast radial symmetry for detecting points of interest. PAMI, 2003.2

[6] D. Reisfeld, H. Wolfson, and Y. Yeshurun. Context-free attentional operators: the generalized symmetry transform. IJCV, 1995. 2

[7] G. Schindler, P. K. R. Lubinearman, Y. Liu, and F. Dellaert. Detecting and matching repeated patterns for automatic geotagging in urban environments. In CVPR, 2008.2

[8] C. Wu, J-M. Frahm, and M. Pollefeys. Detecting large repetitive structures with salient boundaries. In ECCV, 2010.2