

IMPROVED SIFT ALGORITHM TO MATCH POINTS IN THE TEXTURE REGION IMAGE

Cheng-Yi, Chen^a and Shih-Hong, Chio^b

^aPh.D. student, Dept. of Land Economics, National Chengchi University,
NO.64, Sec.2, ZhiNan Rd., Wenshan District, Taipei City 11605, Taiwan; Tel: + 886-2-29393091#50621;
E-mail: 101257505@nccu.edu.tw

^bAssociate Professor, Dept. of Land Economics, National Chengchi University,
NO.64, Sec.2, ZhiNan Rd., Wenshan District, Taipei City 11605, Taiwan; Tel: + 886-2-29393091#51657;
E-mail: chio0119@nccu.edu.tw

KEY WORDS: Feature Extraction, SIFT, Image Matching

Abstract: Feature point extraction automatically instead of manually is important and can improve the efficiency for photogrammetric tasks. Scale invariant feature transform (SIFT) is an important algorithm developed by Lowe (2004) to extract keypoints with invariance on scale and rotation for image matching and registration. SIFT uses descriptor to describe a keypoint and employs these descriptors as keypoint's fingerprint in matching. But when it comes to texture region images, SIFT will produce a large number of similar descriptors and these similar descriptors will generate wrong or failed matching results. In this study, a new algorithm based on SIFT is designed to deal with the matching problem in texture region images. The new algorithm uses the concept of Harris Corner Detector and the entropy in order to make SIFT descriptors more independence. Thus even keypoints are in texture region image, the descriptor of every keypoints is still unique enough to other keypoints. From the tests, this developed new algorithm can make the successful rate of matching up to about 80% in texture region images. The performance of developed algorithm is better than SIFT algorithm.

1. INTRODUCTION

Feature point extraction automatically instead of manually is important and can improve the efficiency for photogrammetric tasks. There are two types of keypoint: Point-like keypoints and Blob-like keypoints. Point-like keypoints stand for specific point, like a junction or a cross point. Blob-like keypoints refer to small area or regions (Förstner, 2009). Therefore keypoints have different types that can be extracted by suitable keypoint extraction methods. Some of them can only extract point-like keypoints, others can extract blob-like keypoints, and the others can extract both. In computer vision field, these two kinds of keypoints are both important, but in a photogrammetric task the point-like keypoints is more important. Because the photogrammetric task needs precise points extracted at an exact location on the object surface, the operators can acquire the actual surface coordinates of the points by photogrammetric principles.

Moravec corner detector (Moravec, 1980) is one of the earliest corner detector, and it also defined what the corner is. It makes the computer has the ability to automatically extract image features and lets photogrammetric task become much more efficient. Then, Harris and Stephens improved Moravec corner detector (Harris, 1988) that makes the computer more accurately identify the feature points. This method always finds the keypoints on the corner that is useful for photogrammetric task so it is one of the methods currently used. But it only provides the image coordinates of the point during the image matching; if the imaging condition of the images must be quite close to each other, the matching results will be good enough. In other words, the images' scale, brightness, rotation, etc. must be very similar, otherwise image matching will fail. This is not a big problem in aerial photography, but in close-range photogrammetry it's a serious problem. Because the orientation and scale of images in close-range photogrammetry are quite different, only the image coordinates of keypoints are not sufficient to get more correct matching results.

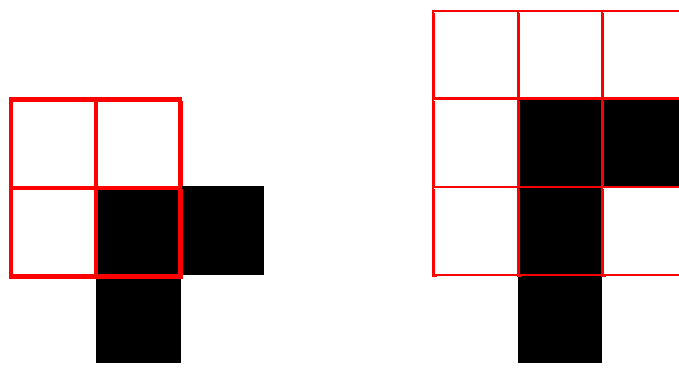
Scale invariant feature transform (SIFT) is an important algorithm developed by Lowe (2004) to extract keypoints with invariance on scale and rotation for image matching and registration. It can overcome the matching difficulties caused by the different orientation and scale in close-range images, and it provides every keypoint a unique descriptor that will make the image matching or registration more reliable. These descriptors just like the fingerprint of keypoint, every single keypoint in different images should have the same descriptor in theory. So it is useful in image matching, these descriptors can be provided for efficient image matching. Even though SIFT can find large numbers of keypoints in every single image, these keypoints are not only point-like keypoints but also blob-like keypoints. It can say that SIFT finds numerical keypoints instead of geometric keypoints. As mentioned earlier, blob-like keypoints are less suitable for photogrammetric task, therefore highly accurate photogrammetric

operations must exclude those blob-like keypoints. In addition, because keypoints with descriptors is more suitable to match correct corresponding points for close-range photogrammetric task, there are some algorithms which used descriptor to improve the common feature point extraction algorithms. For example, contrast context histogram (CCH) descriptor (Huang, 2008) uses Harris corner detector to extract keypoints and gives every keypoints a descriptor. This approach ensures that all extracted keypoints are point-like keypoints and these keypoints with descriptor are more suitable for image matching.

In some texture region images, e.g. forest area or buildings with glass curtain, it is hard to acquire a good keypoint for matching, because the descriptors in these texture region images will be analogous. These analogous descriptors will make the result of matching incorrect. So in photogrammetric tasks, it will cost a lot of time to correct the wrong matching.

2. HARRIS CORNER DETECTOR (Harris, 1988)

When it comes to Harris corner detector, it should explain the Moravec detector firstly. Moravec detector uses a moving window to calculate the sum of the gradient. If the difference is larger than a threshold, the point is called keypoint, like Figure 1(a). When the window size increases, it will fail at the area with alternating light and dark because of the gradient of the positive and negative offset, like Figure 1(b).



(a) (b)
Figure 1: (a) The red window is moving on the image, when it moves to the center of the image it will have the largest sum of the gradient. So there should have a corner point. (b) The red window cross the bright area and dark area, and it will fail to find a corner because it makes the sum of the gradient's difference small.

Harris used numerical methods to improve Moravec detector by utilizing a matrix to calculate the eigenvalues and find the corner or the edge point according to the equation (1):

$$A = \sum_u \sum_v w(u, v) \begin{bmatrix} G_x^2 & G_x G_y \\ G_x G_y & G_y^2 \end{bmatrix} = \begin{bmatrix} \langle G_x^2 \rangle & \langle G_x G_y \rangle \\ \langle G_x G_y \rangle & \langle G_y^2 \rangle \end{bmatrix} \quad (1)$$

Where A is the structure tensor, $w(u, v)$ is a window over the pixel (u, v) , G_x is the gradient of x-direction, G_y is the gradient of y-direction, and angle brackets denote averaging. This matrix is called as Harris Matrix. By finding two eigenvalues of Harris Matrix, the pixel can be determined as a corner pixel, an edge pixel, or nothing. Let λ_1 as the bigger eigenvalue, and λ_2 as the smaller eigenvalue, and the rules are shown below:

1. If $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$ then this pixel (x, y) is not a corner pixel or an edge pixel.
2. If λ_1 has some large positive value and $\lambda_2 \approx 0$ then this pixel (x, y) is an edge pixel.
3. If λ_1 and λ_2 have some large positive values then this pixel (x, y) is a corner pixel.

It can be said that the two eigenvectors of Harris matrix depict the two directions of the pixel, and the two eigenvalues is the strength of the direction, the three rules can be represented by Figure 2:

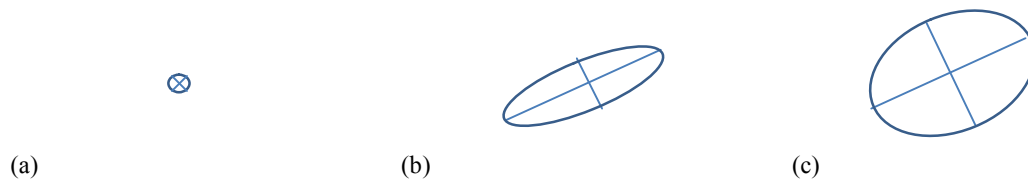


Figure 2: (a) Two eigenvalues are both very small, so it is not an obvious point. (b) Only one eigenvalue is very big, so it is on the edge. (c) Both two eigenvalues are very big, so it is a corner.

SCALE INVARIANT FEATURE TRANSFORM (SIFT) (Lowe, 2004)

SIFT is an algorithm to extract and describe the keypoints from images. It can be applied in object recognition, video tracking, etc. It can be used to match image features on different images even though these images have different scales and rotations. Because of the features mentioned above, SIFT is useful in close-range photogrammetry.

In original paper, SIFT contains four main steps (Lowe, 2004):

1. Scale-space extreme detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

But for illustration, the following six steps will be used to explain the SIFT algorithm in this study:

1. Create images with different scales:

Images with different octaves from original image are created. The number of octaves depends on the size of original image. In detail, images in the first octave are employed to generate progressively blurred out images, then the original image is resized to half size and this resized image is transformed to be blurred images by Gaussian Smoothing again. Original paper had suggested that 4 octaves and 5 blur levels are ideal.

2. Generate DOG images:

In theorem, Laplacian of Gaussian (LOG) can be used to detect and extract edge and corner pixels based on the second order derivatives of gray level in an image. Yet LOG is sensitive to noise, DOG (Difference of Gaussian) is used to approximate Laplacian of Gaussian.

The images created in step 1 are utilized to calculate the difference between two images with consecutive scales, like Figure 3. After DOG images are generated, edge and corner pixels can stand out.

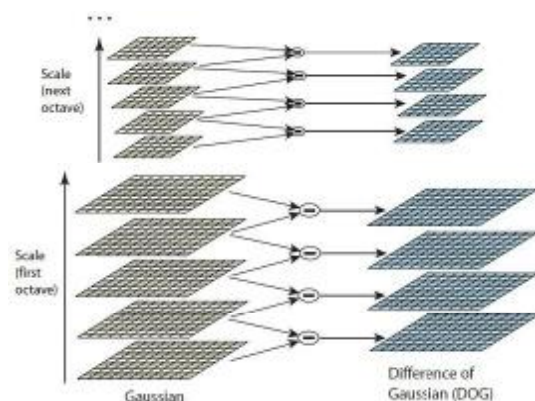


Figure 3: Difference of Gaussian (DOG) (Lowe, 2004).

3. Find Keypoints:

A lot of edge and corner pixels will be found after processing in step 2. This step will find candidate keypoints. For each point, eight points around it and nine points at the same location of its upper and lower image, total 26 neighbor points (shown in Figure 4), will be taken into account to determine the

candidate points. If the DOG value of this point is the maxima by comparing to its 26 neighbor points, then the point is viewed as a keypoint.

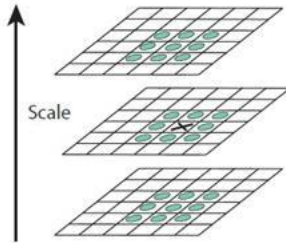


Figure 4: Use the 26 neighbor points to find the X mark whether is maxima (Lowe, 2004).

4. Delete edge and low contrast points:

SIFT needs to find specific keypoints, but the points found in step 3 may be on the edge or with low contrast. For eliminating low contrast points, a threshold value can be used to filter low-contrast points. Additionally, two eigenvalues of Harris Matrix of each point can be used to decide whether the point is on the edge or not. If one eigenvalue has some large positive value and the other eigenvalue is a very small value, then this pixel (x, y) is an edge pixel.

5. Determine the orientation of keypoint:

It will give every keypoint an orientation in this step. For each keypoint, its magnitude and orientation for all pixels around the keypoint will be calculated by the following equations (2) and (3):

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \quad (2)$$

$$\theta(x, y) = \tan^{-1} \left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)} \right) \quad (3)$$

$m(x, y)$ is magnitude of a pixel (x, y) and $\theta(x, y)$ is orientation of a pixel (x, y) , $L(x, y)$ is the gray value of a pixel (x, y) . A histogram is created for determining the keypoint orientation. When creating a histogram, it is recommended to divide 360 degrees into 8 regions. That means $0^\circ \pm 22.5^\circ$, $45^\circ \pm 22.5^\circ$, and so on. Then a 16×16 window (see Figure 5) with a keypoint at center is generated, a histogram is made every 4×4 region, after all 4×4 regions' histogram done, the orientation of the keypoint can be selected from the highest histogram.

6. Create SIFT features:

When the histogram of this keypoint is done, the descriptor will be formed like Figure 5. The histograms will be saved into a 128 dimension's vector, for example: [8, 31, 16, 21, 40, 32, 13, 5, 28, 62, 35, 16, 15, 31, 42, 27, 33, 68, 42, 0, 0, 1, 40, 68, 55, 77, 32, 1, 3, 5, 28, 35, 63, 73, 41, 12, 8, 17, 15, 10, 52, 34, 32, 77, 37, 36, 47, 31, 114, 76, 16, 23, 13, 7, 61, 114, 23, 24, 36, 79, 79, 27, 55, 37, 25, 53, 56, 17, 19, 35, 39, 44, 55, 33, 32, 32, 7, 6, 82, 114, 114, 44, 37, 25, 9, 2, 36, 114, 7, 9, 43, 53, 40, 18, 38, 5, 19, 114, 35, 21, 10, 12, 31, 37, 53, 92, 28, 0, 0, 7, 27, 63, 10, 17, 73, 34, 17, 49, 25, 12]. This vector is called SIFT descriptor.

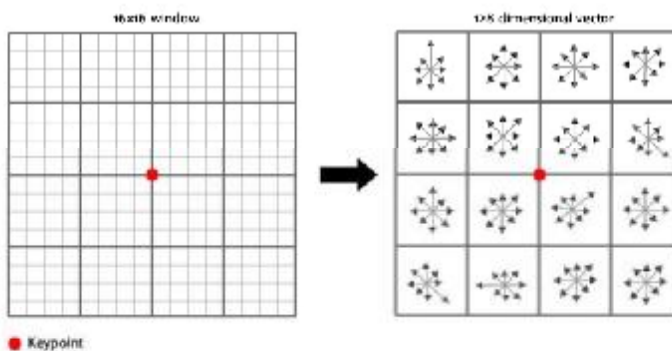


Figure 5: The illustration of keypoint descriptor (Lowe, 2004).

IMPROVED SIFT ALGORITHM

Because SIFT descriptor is based on a histogram, when there are large numbers of duplicate objects appear, it will produce many similar gradient histograms and cause the matching error. For general building, it usually has duplicate objects like windows, therefore the method of entropy is used to make descriptor more unique (Wu, 2007).

Entropy is a region's randomness. An entropy value e in a rough area will be higher than in other regions, because the region has a higher randomness. The equation to calculate the entropy is as follows:

$$e = -\sum_{i=0}^{L-1} p(z_i) \log_2 p(z_i) \quad (4)$$

Where z is the gray level values within the region, $p(z_i)$ is the number of pixels of the grayscale value z . By combining the SIFT descriptor and the entropy value e , a new descriptor is obtained as follows:

$$D = \begin{bmatrix} \omega \times S \\ (1 - \omega) \times E \end{bmatrix} \quad (5)$$

Where D is new descriptor, S is SIFT descriptor, E is an entropy value, ω is the weight, and usually ω is 0.5.

The difference between SIFT and improved SIFT processing is shown in Figure 6. As shown in Figure 6, the improved SIFT algorithm uses Harris Matrix to ensure that every keypoints are all Point-like keypoint. Additionally, the entropy value is used to make the descriptor more independent. This can make every keypoint significantly different from each other, thus it can make the point matching much more correct.

SIFT algorithm

1. Create images with different scales
2. Generate DOG images
3. Find Keypoints
4. Delete inappropriate points
 - a. Delete edge points
 - b. Delete low contrast points
5. Determine the orientation of keypoint
6. Create SIFT features

Improved SIFT algorithm

- a. Delete edge points
- b. Delete low contrast points
- c. Delete the points that its two eigenvalues of Harris Matrix are both small value*

- a. Give every descriptor entropy value*

Figure 6: The difference between SIFT and improved SIFT

3. EXPERIMENTS AND RESULTS

The following test will prove that our developed improved SIFT algorithm can work well in the images with texture regions. Test stereo images are taken by Fujifilm FinePix REAL 3D W3 3D camera in the campus of National

Chengchi University(see Figure 7).There are 10buildings in these 14 stereo image pairs and each image size is 1280x1010.The improved SIFT and SIFT algorithm will extract keypoints from these test images and match the corresponding keypoints.From the test images, texture regions in these buildings can be found, e.g. the windows, the tiles,etc. so these images are good target to test the algorithmdeveloped in this study.





Figure 7:Test images

Table 1 is the results of the matching by employing SIFT and improved SIFT algorithms.The matching correctness is calculated by the following equation (6).


In this study,the evaluation of four parameters transformation between two images will be used to determine whether the result is correct or not. It uses least square method to calculate the four transform parameters. When the residual is greater than three times the standard deviation, the result is considered as a wrong matching. The number of correct matching pairs is the total number of matching pairs minus the wrong matching pairs.

$$\text{Matching Correctness} = \frac{\text{Number of correct matching pairs}}{\text{Number of total matching pairs}} \times 100\% \quad (6)$$

Table 1: Results of the matching by employing SIFT and improved SIFT algorithms, respectively

Image pair	Improved SIFT algorithm			SIFT algorithm		
	All match point	Wrong matching point	Correctness %	All match point	Wrong matching point	Correctness %
	805	48	94.04	1891	380	79.90
	326	1	99.69	714	78	89.08

	1353	127	90.61	3684	648	82.41
	938	79	91.58	1636	190	88.39
	584	8	98.63	1000	78	92.20
	758	37	95.12	1378	153	88.90
	780	5	99.36	2614	304	88.37
	461	49	89.37	2051	499	75.67
	608	9	98.52	1259	163	87.05
	910	178	80.44	3743	1310	65.00

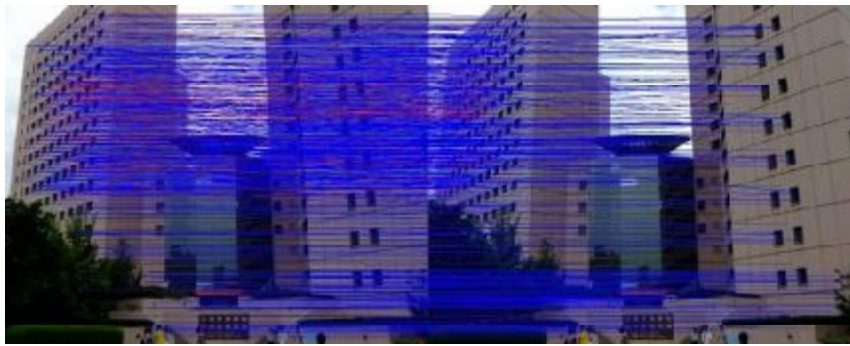
	282	3	98.94	1268	184	85.49
	542	23	95.76	1070	162	84.86
	360	13	96.39	2462	560	77.25
	597	15	97.49	3118	925	70.33
Average correctness			94.71			82.49

Subsequently, Z-test is used to compare correctness, as shown in Table 2.

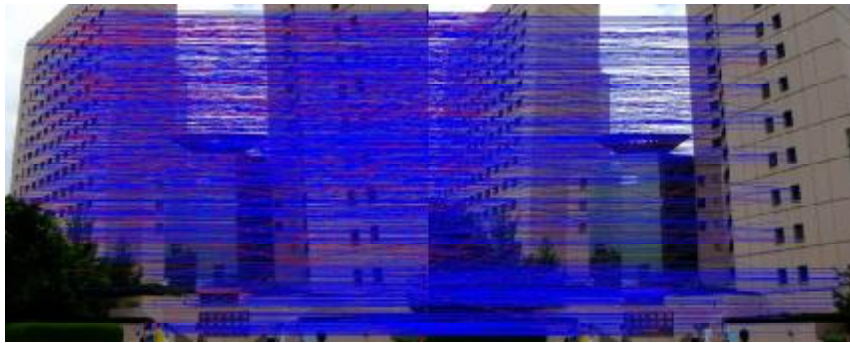
Table 2: Z-test at confidence level of 95% of the two algorithms

H0	H1	Z-value	Critical value (one-tail)	Result
Improved SIFT is the same with SIFT	Improved SIFT is better than SIFT	5.147089	1.644853	Reject H0, means improved SIFT is better than SIFT

By Z-test it shown that improved SIFT algorithm is better than the SIFT algorithm in texture region. One of the matching results of test images are shown in Figure 8:



(a)



(b)

Figure 8: Matching samples. (a) Improved SIFT algorithm. (b) SIFT algorithm. Blue line is correct matching, and red line is wrong matching.

4. CONCLUSIONS & RECOMMENDATIONS

According to the experimental results, the improved SIFT algorithm performs better than the SIFT algorithm in image matching. Especially in the texture region, the improved SIFT algorithm finds almost no blob-like keypoints. It can be said that the performance of improved SIFT algorithm is better than the SIFT algorithm for the applications in close-range photogrammetry. The points that the improved SIFT algorithm extracted can correspond to the definite position in real world, so the operators can verify these points for subsequent application, such as 3D building model reconstruction automatically. It can enhance operational efficiency of close-range photogrammetry and make the 3D model more accurate.

In the future study, because Harris's algorithm can find edge points (Harris, 1988), it will try to match edge points as 3D line features by improving SIFT for providing more information to build 3D object model automatically.

In this study, the four-parameter transform between two images is used to determine whether the result is correct or not. But in close range photogrammetric, the objects in image usually have different distance and angle to camera, four-parameter transform may not have good ability to detect error result in matching. So in the future relative orientation between two images should be used to detect the wrong matching result.

REFERENCES:

Förstner, W; Dickscheid, T; Schindler, F, 2009. Detecting Interpretable and Accurate Scale-Invariant Keypoints In: 12th IEEE International Conference on Computer Vision (ICCV'09). Kyoto, Japan 2009, pp. 2256-2263.

Hans P. Moravec, 1980, Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover, tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University.

Harris, C. G. and Stephens, M. J., 1988. A combined corner and edge detector, Proceedings of the Fourth Alvey Vision Conference, Manchester, United Kingdom, pp. 147-151.

Huang, C.R., Chen, C.S., Chung, P.C., 2008, Contrast context histogram—an efficient discriminating local descriptor for object recognition and image matching. Pattern Recognition (PR 2008) 41(10), 3071–3077.

Jiunn-Lin Wu, Yen-liang Chen., 2007. A Robust Feature-Based Registration Method for Differently Exposed Image Sequences. Journal of Computer Science and Application, 3(1), 1-22.

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision, Vol. 60, No. 2, pp. 91-110.