Comparison and Application of Implementing Image Homographs in Computer Vision

Xingqi Qiu

School of International Education, GuangDong University of Technology, Guangzhou, 511495, China
3121010028@mail2.gdut.edu.cn

Abstract. In the field of computer vision, planar homography plays a pivotal role in our research process. The homography matrix is capable of performing a variety of functions such as image warping, stitching, and video stitching. Within the realm of epipolar-geometry, it enables the execution of numerous tasks, including 3D reconstruction. This paper primarily focuses on the creation of panoramic images through automatic stitching of photographs with using homographic matrix, comparing the efficacy and efficiency of different feature extraction algorithms in terms of feature point matching, like Speeded Up Robust Features (SURF), Features from Accelerated Segment Test (FAST), and KAZE Scale-Invariant Feature Transform (SIFT), Oriented FAST and Rotated BRIEF (ORB),and put forward some applications. Consequently, this leads to variations in the effectiveness and efficiency of images stitched using the homography matrix. This paper finished a feature matching experiment based on comparing the panoramic image with using different feature detect algorithms. For scenarios requiring high accuracy where processing time can be longer, SIFT, KAZE, or SURF might be better choices. On the other hand, for applications that need fast response, FAST or ORB would be more appropriate.

Keywords: Feature Detect Algorithms, Deep Learning, Panorama, Homograph.

1 Introduction

Panoramic image stitching is an active research area that involves multiple sub-tasks such as image detection, alignment, stitching, and image blending [1]. In the ever-evolving landscape of computer vision, the application of homography matrices alongside feature extraction and matching algorithms has increasingly become a focal point. The efficacy of these complex techniques, however, can vary substantially depending on a multitude of situational factors. Consequently, it is imperative in a range of differing contexts to ascertain the algorithm that is optimally aligned with the requirements of the given application. As it stands, a significant body of contemporary research endeavors to refine the methodologies underpinning feature extraction and matching. The pursuit of obtaining precise planar homography matrices, especially within the confines of more restrictive environments, represents one such area of intense scholarly concentration. The current state of the art in this field is underscored by a series of notable advancements and ongoing challenges. Foremost among these is the progression of sophisticated feature detection and matching algorithms, which form the cornerstone of many computer vision tasks. Innovations in homography computation and the development of robust transformation models continue to enhance our capacity to manipulate and align images with high precision.

However, different feature extraction and matching algorithms may demonstrate varied effectiveness under different circumstances. This means that in diverse scenarios, identifying the most suitable algorithm for application is requisite. Currently, the majority of research is dedicated to improving this feature extraction and matching techniques and obtaining planar homography matrices under more constrained conditions. The current state of the field can be characterized by several key developments and challenges: Advanced Feature Detection and Matching Algorithms, Homography and Transformation Models, Real-time and Panoramic Video Stitching.

Nonetheless, in most instances, when we need to employ mature algorithms to accomplish tasks, the challenge lies in choosing the most appropriate algorithm for the task at hand. This paper introduces several classic and practical feature detection algorithms and provides a detailed comparison of these algorithms' differences and applicability in static environments and dedicated to comparing the differences among these algorithms and explain the method of how to use auto-stitched skills to make a panoramic image and compare the different feature detection algorithms under the same setting and parameters.

2 Method

This part is about to introduce the typical methods the paper used. Besides, this part will show the basic steps of that algorithms and give the brief description.

2.1 Model construction

SIFT Scale-invariant feature transform (SIFT) features were proposed by David G. Lowe [2] and represent an algorithm for image feature detection and description. It extracts location-, scale-, and rotation-invariant keypoint descriptors and locates scale-space extrema in pictures. This is extensively employed in domains like object identification and picture matching. It operates by identifying precise locations and main orientations of extrema (feature points, keypoints) across different scale spaces, constructing keypoint descriptors for feature extraction. SIFT is renowned for its high stable and strong robustness, what's more the discrimination ability of SIFT is also admirable. However, inevitably, SITF need higher computation cost, and which is an enormous and fatal problem in currently. Therefore, it is hard to use SIFT to deal with the real-time process assignment.

The concept of the Gaussian pyramid was introduced in the context of the SIFT operator. The Gaussian pyramid is built using the following steps: First, the original image is doubled in size. This becomes the first layer of the first octave in the Gaussian pyramid. The second layer of the first octave pyramid is created by applying Gaussian convolution to the picture at the first layer of the first octave. The Gaussian convolution function is utilized in this process, in SIFT the σ =1.6.

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \quad (1)$$

Then, by multiplying σ (sigma) by a scaling factor k, a new smoothing factor σ = k*σ is obtained. This new factor is used to smooth the image at the second layer of the first octave, and the resulting image serves as the third layer, continuing in this manner, we eventually obtain L layers of images. Within the same set, each layer of images has the same dimensions but different smoothing factors. By taking the image from the third-to-last layer of the first set and down sampling it by a scaling factor of 2, we obtain the first layer of the second set (Table 1).

Table 1. SIFT algorithm

	SIFT algorithm
1.	Potential locations of interest are indicated using a Gaussian differential function that is invariant to scaling and rotation.
2.	To find the scale position, an enhanced model is applied to each possible site, choosing important points according to their stability.
3.	Each keypoint is allocated one or more directions based on the local image gradient directions. The direction, size, and position of important points are transformed throughout subsequent procedures to maintain their integrity.
4.	At the chosen scale, local image gradients are measured close to each feature point. These gradients are converted into a representation that guarantees a low translational similarity by permitting large distortion and illumination variations in local shape comparison.

SURF The Speeded Up Robust Features (SURF) algorithm [3] is an improvement and acceleration of SIFT (Table 2). SIFT is popular for its accuracy and robustness, but it may be slower due to its large computational load. On the other hand, while retaining some of the scale and rotation invariance features of SIFT, SURF has been optimized for speed, making it particularly useful in real-time applications or situations where computational resources are limited. It improves the computational efficiency of feature detection and description by using integral images to quickly approximate the Hessian matrix and box filters to accelerate the convolution process.
SURF employs box-type filters as an approximation to the Gaussian blur used in SIFT, which allows for faster computation. The filter's size fluctuates, while the image's size stays constant. The following is a representation of the filter:

$$S(x,y)=\sum_{i=0}^{x}\sum_{j=0}^{y}I(i,j) \quad (2)$$

In order to attain scale invariance, SURF further employs the Hessian matrix's σ-scale determinant for feature point detection. The Hessian matrix H(x, σ) at scale σ is defined for a given point x=(x, y) in the picture as follows:

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{3}$$

$L_{xx}(x, \sigma)$ is the second order Gaussian derivatives' convolution assessed at point x.

Table 2. SURF algorithm

	SURF algorithm
1.	SURF uses integral images to quickly calculate the sum of intensities in any image rectangle.
2.	The algorithm utilizes a Hessian matrix-based approach for keypoint detection.
3.	Similar to SIFT, SURF refines the position, scale, and ratio of the detected keypoints to find the most stable and accurate features.
4.	For each keypoint, an orientation is assigned based on the dominant direction of the local gradients around the keypoint.
5.	Constructs descriptors for the keypoints by extracting the sum of the wavelet responses around the keypoint location. The responses are organized into a vector, which represents the local gradient structures around the keypoint.
6.	Keypoint Matching.

ORB Oriented Features from Accelerated Segment Test (FAST) and **R**otated **B**inary Robust Independent Elementary Features (BRIEF) (the full name of ORB) [4], is a highly efficient feature point detection and description algorithm designed to be an effective alternative to SIFT and SURF (Table 3). Building on the strengths of FAST keypoint detection and BRIEF descriptors, ORB introduces keypoint orientation and employs a swift binary descriptor to facilitate efficient matching, which is a new concept compared to the SURF and SIFT. By utilizing rapid binary strings for feature description and matching, ORB is faster than SIFT and SURF, making it more suitable for real-time applications. ORB has been specially designed with a rotation-invariant weighting mechanism, and it employs an efficient learning algorithm to ensure quick feature description computation. Although ORB adopts certain scale invariance measures through its multi-scale detection algorithm, its performance in handling extensive scale changes is not as robust as that of SIFT and SURF As this paper mentioned before, orb is usually used in real-time process tasks. The Rotated BRIEF algorithm generates a binary string descriptor by selecting n pairs of pixels, denoted as pi and qi (i=1, 2…, n), around a feature point within its neighborhood. It then compares the grayscale values of each pixel pair. If the intensity at pixel pi exceeds that at pixel qi, a '1' is placed in the binary string; otherwise, a '0' is assigned. After comparing all designated pixel pairs, a binary string of length n is produced.

If P is a feature point and Q is the gray centroid within the neighborhood, the vector $\overrightarrow{PQ}$ represents the direction of the feature point. The calculation method for the centroid is as follows:

$$m_{ij} = \sum_{x=-r}^{r} \sum_{y=-r}^{r} x^i y^j I(x,y) \tag{4}$$

In this case, r is the neighborhood's radius, i and j are either 0 or 1, and I(x, y) is the gray value of the pixel at (x, y). The gray centroid may thus be written as follows:

$$Q = \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \tag{5}$$

The angle of the feature point is represented as:

$$\theta = arctan(m_{01}, m_{10}) \tag{6}$$

Table 3. ORB algorithm

ORB algorithm
1. ORB begins by swiftly detecting potential feature points in the image as corners using the FAST algorithm.
2. These FAST corners are then scored and filtered using the Harris corner detector to determine the most prominent keypoints.
3. For each keypoint, ORB calculates its orientation using a method based on the intensity centroid of the surrounding pixels, enhancing rotational invariance of the feature.
4. Subsequently, ORB applies a rotated version of the BRIEF descriptor to the pixels around each keypoint to maintain the descriptor's orientation consistency.
5. these orientation-corrected BRIEF descriptors can be used for efficient feature point matching, especially using Hamming distance for fast matching.

KAZE A key attribute of the KAZE algorithm is its use of non-linear scale-space technology, which marks a significant departure from methods based on linear scale spaces like SIFT and SURF. In non-linear scale areas, KAZE provides more accurate feature recognition and description in comparison to traditional techniques like SIFT and SURF. Designed with the dual aims of preserving essential image features while eliminating noise, KAZE may necessitate more computational resources than SIFT and SURF. However, the improvements in performance and effectiveness it brings to certain applications offer clear advantages. The KAZE algorithm's developers build a non-linear scale space by using Additive Operator Splitting (AOS) and non-linear diffusion filtering techniques. Variations in picture luminance (L) over various scales are treated by non-linear diffusion filtering as a divergence of a flow function, which is characterized by a non-linear partial differential equation:

$$\frac{\partial L}{\partial t} = \mathrm{div}(c(x,y,t)\nabla L) \tag{7}$$

The c(x,y,t) means the $g(|\nabla L_\sigma(x,y,t)|)$. The gradient ∇L_σ is of the Gaussian-smoothed image L_σ and several expressions for the g() function are provided in the article “KAZE Features” [5] (Table 4). This equation significantly enhances the algorithm’s ability to manage luminance changes, making it particularly effective for high-detail image analysis. The non-linear approach ensures that features are preserved with higher fidelity, supporting advanced image processing tasks.

Table 4. KAZE algorithm

KAZE algorithm
1. Constructs a non-linear scale space to efficiently handle image details at different scales.
2. Proceeds to detect key points using a multi-scale approach.
3. Applies a descriptor to each key points to capture its unique attributes, enabling the comparison of features across different images.
4. the algorithm finalizes the feature set, readying it for applications such as image matching, recognition, or tracking.

FAST The FAST [6] corner detection algorithm focuses on high-speed corner detection using a technique known as "Accelerated Segment Test" to swiftly identify corner points within an image (Table 5).

Table 5. FAST algorithm

FAST algorithm
1.Choose a candidate pixel in the image that might be a corner. This pixel is referred to as the test pixel.
2.Surround the test pixel with a circle of 16 pixels at a certain radius. This circle is used to determine if the test pixel is a corner.
3. Examine the test pixel's intensity in relation to the 16 other pixels' intensities within the circle. If a pixel on the circle has an intensity that differs noticeably from the test pixel's intensity, it is deemed to be a component of the corner.
4.If there is a collection of n consecutive pixels in the circle (where n may generally be 12 out of 16) that are all darker than the test pixel by the same threshold (let's say t) or all brighter than the test pixel by the same threshold, then the pixel is considered a corner.
5.In order to increase the algorithm's efficiency, a faster test looks just at the first, fifth, ninth, and thirteenth pixels before looking at all 16 pixels. To cut down on computation time, the test pixel is instantly excluded as a corner if these pixels are not all darker or all brighter than the test pixel (plus or minus the threshold).
6.Once all potential corners have been detected, a non-maximum suppression (NMS) step is applied. This step eliminates corners that are not local maxima to refine the corner detection and reduce the number of features.
7.Optionally, the intensity threshold (t) can be varied to control the number of corners detected, allowing the algorithm to adapt to different image conditions and requirements.

The key advantage of the algorithm is its velocity, making it suitable for real-time visual systems. the FAST algorithm excels in speed, making it well-suited for real-time applications. Its main trade-off is the potential reduction in the robustness of detected features, especially in comparison to more comprehensive (but slower) algorithms like SIFT and SURF. However, its efficiency makes it a popular choice, especially in applications where processing time is critical, and it serves as a fundamental building block for more complex feature detection and matching algorithms like ORB.

In fact, when comparing pixel grayscale values, a threshold (t) needs to be added.

$$S_{p->x} = \begin{cases} d, I_{p\to x} \leq I_p - t \ (dark) \\ s, I_p - t < I_{p\to x} < I_p + t \ (simialr) \\ b, I_p - t \leq I_{p\to x} \ (brighter) \end{cases} \tag{8}$$

In this expression, p->x means the pixel point x on the ring which surround pixel point p, (x=1,2,3...,16), and the $\mathrm{S_{p->x}}$ means the interval category corresponding to the grayscale of point p. This means, if the neighborhood of point p has a sequence of n points whose comparison results are $\mathrm{S_{p->x}}$ =d or b. then p is considered a corner point. Generally, n is set to 12, known as FAST-12; in practice, n=9 often yields better results.

2.2 Feature Detection

Feature detection refers to the process of identifying points of interest within an image. These points, or features, are distinctive and can be easily recognized in different images of the same scene or object. For instance, the Harris corner detector [7], which focus on the corner point. It slid a fixed window in any direction on an image, one compares the situation before and after the slide. If a significant change in the grayscale within the window area occurs between the pre-slide (before the sliding) and post-slide situations, it is considered that a corner has been encountered within the window. If there is no change in the grayscale within the window area before and after the slide, then there are no corners present in that window area. The goal is to detect features that are invariant to transformations such as scaling, rotation, and illumination changes.

SIFT, SURF, ORB, and KAZE are a few of the frequently utilized feature identification techniques. These algorithms look for regions in the image that have unique patterns or textures that stand out from their surroundings, which can then be used as markers for identifying and tracking objects across multiple images.

2.3 Feature Matching

Feature matching takes the process a step further by finding correspondences between features detected in different images. This involves comparing features detected in one image with those in another to find matches based on similarity measures. The matching process is critical in applications such as stereo vision, where matching features across left and right images allow for depth estimation, and in panorama stitching, where overlapping images are seamlessly merged based on matched features. Algorithms like the Brute-Force matcher and the Fast Library for Approximate Nearest Neighbors (FLANN)-based matcher [8] are often used for this purpose, and this paper focus on the Brute-Force matcher. Feature matching relies on descriptors, which are unique signatures generated for each detected feature during the detection phase, to compare and identify similar features across different images.

Feature detection and feature matching enable a wide range of computer vision applications, from augmenting reality to creating 3D models from 2D images. Their effectiveness lies in their ability to identify and leverage the unique characteristics of images to recognize, track, and analyze objects across different views and scales.

In the process of feature matching, we employ the K-nearest neighbors (KNN) algorithm to sift through initial matches, further refining these matches by implementing Lowe's ratio test. This involves taking a keypoint from one image, then identifying its two closest matches in the other image based on the Euclidean distance. We estimate the distance between the closest and second-closest matches; if the quotient of these distances is lower than a predetermined threshold (T), the match is considered acceptable. Obviously, lowering the ratio threshold (T) will reduce the number of matching points but make them more stable. In our experiments, this paper actually recommends a ratio threshold of 0.8 because some algorithms, like ORB, require more matching points. Experimental observations suggest that an optimal threshold ranges from 0.4 to 0.6. Values under 0.4 lead to a scarcity of matches, while those over 0.6 tend to include numerous inaccurate matches. This approach is crucial for mitigating issues related to keypoints that do not match due to factors like image occlusion and complex backgrounds, ensuring that only the most plausible matches are retained.

Planar Homography transformation refers to the mapping relationship between two different two-dimensional plane matrices on a plane. This mapping is a linear transformation, with specific correspondences, such that for every point on one matrix, there is a unique corresponding point on the other matrix [9]

3 Result

3.1 Settings

In order to find the homography matrix to implement applications, this paper used SIFT, SURF, ORB, FAST algorithms to do the feature detection and used Random Sample Consensus (RANSAC) [10] to deal with the noisy point and estimate the parameter in modeling. In feature matching, this paper only uses the Brute force matching. The “reprojThresh” (reprojective threshold) is the maximum allowable reprojection error threshold used in computing the homography (perspective transformation) between images. This parameter is crucial in image stitching, particularly in feature matching and finding the homography matrix. This paper has the fix “reprojThresh”, which equal to 50. The match rate means the rate of all the detected points from the two original images divided by the number of matched points.

To compare the application effects of homography matrices generated under different feature detection algorithms, this paper utilizes plane homography to realize panoramic images, hereby comparing the differences among various detection algorithms under static conditions.

3.2 Evaluation Metrics

Incorrect detection and matching can impact the intuitive stitching effect. The robustness of the method and use requirements are reflected in the number of feature points created from pictures and the needed number of feature points. The difficulty of the approach is demonstrated by the runtime for producing panoramic photos. The quality of the homogeneous matrices produced by the algorithm may be seen in the size of the teeth in the gaps.

3.3 Experiment

This experiment used different feature detected algorithm to obtain the homography matrix to implement the panoramic image and feature matching, which aim to compare the efficacy and effect, hereby comparing the differences among various detection algorithms under static conditions (Figure 1).

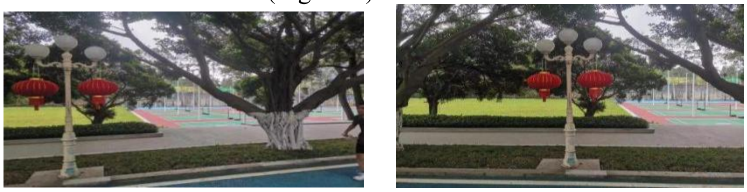


Fig.1. Original figures

SIFT KAZE ORB FAST Feature Match Here, this paper set three different ratio,0.6, 0.8, and "reprojective" threshold=50.0 (Table 6, figure 2, figure 3, figure 4, figure 5).

- Ratio=0.6

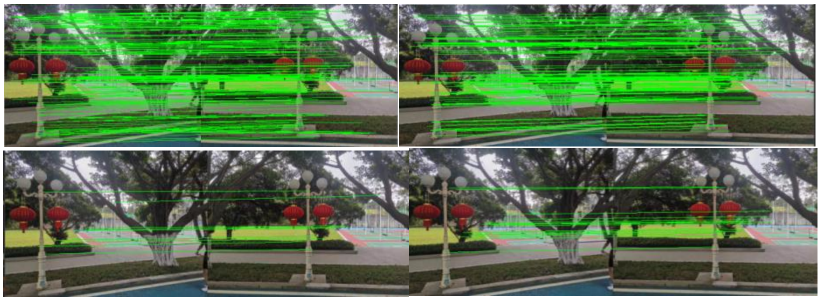
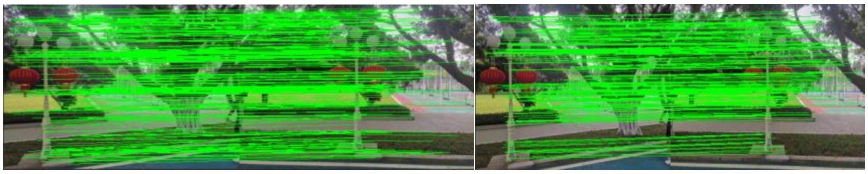


Fig. 2. Showing matching points with SIFT KAZE ORB FAST from top to bottom, left to right. (Ratio=0.6)

- Ratio=0.8



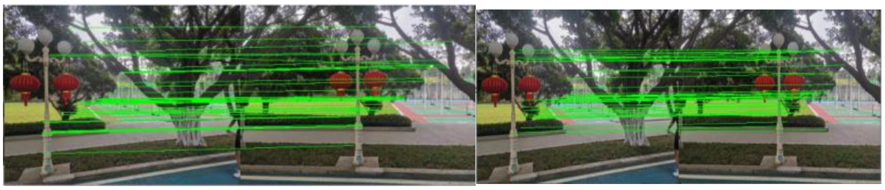


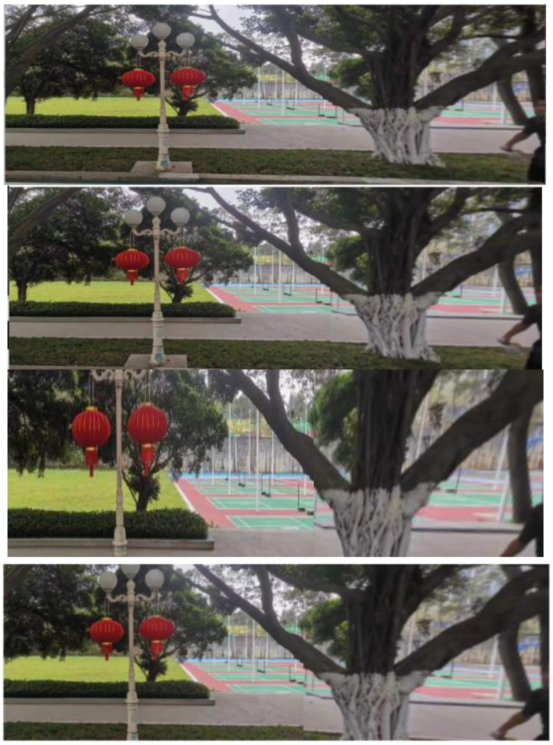
Fig.3. Showing matching points with SIFT KAZE ORB FAST from top to bottom, left to right. (Ratio=0.8)

Table 6 The experiment data collection

	Number of matched points ratio=0.6	Number of matched points ratio=0.8	Running Time ratio=0.6	Running Time ratio=0.8	Match Rate Ratio=0.6	Match Rate Ratio=0.8
SIFT	162	272	0.146	0.147	13.84%	24.38%
KAZE	137	224	0.2506	0.256	19.27%	32.35%
ORB	10	52	0.354	0.310	2.02%	12.08%
FAST	16	43	0.0897	0.0977	11.94%	33.58%

SIFT KAZE ORB FAST SURF comparison of panorama

- Ratio=0.6



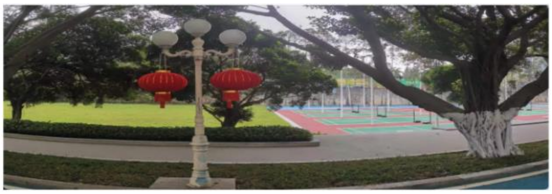


Fig.4 Panorama image with SIFT KAZE ORB FAST SURF from top to bottom, left to right.

- Ratio=0.8

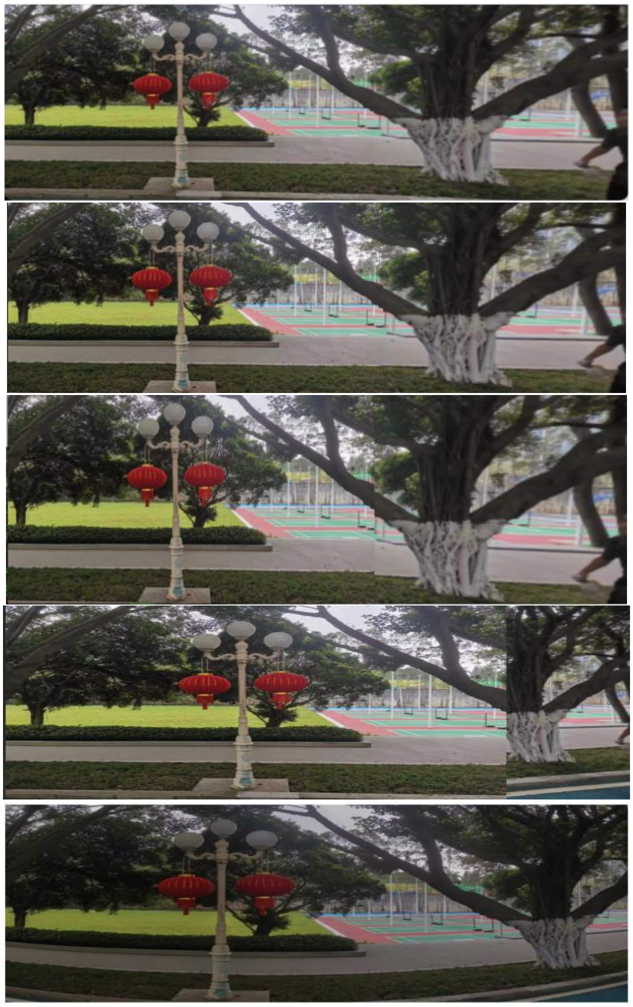


Fig.5 Panorama image with SIFT KAZE ORB FAST SURF from top to bottom, left to right

To compare the differences in the effects of using homographic matrices obtained through various feature detection and matching algorithms, this paper employs different feature detection algorithms combined with brute-force feature matching techniques and RANSAC to create homography matrices for implementing panoramic stitching technology. The results indicate that, under certain feature point matching constraints, there are noticeable differences in outcomes.

The generic rank of feature detected algorithm for their ability to detect low quality and low quantity of features is:

SIFT>KAZE>FAST>ORB

But, if we enhance the restriction of the matching:

SIFT>KAZE>ORB>FAST

That meanings that in the static feature matching SIFT shows the best performance among the others however, at the meanwhile, SIFT and KAZE probably shown more costly in large-scaled task. The ORB and FAST doesn't have good enough effect to support them to be used in static and low quantity image stitching task because of the lack of the matching points.

The generic order of running time (from high to low) for their ability to finish panoramic image (in low quantity match points) is:

ORB>KAZE>SIFT>FAST

Specially, when using ORB to do the panoramic image, if the number of the matched points increase, the running time decrease at the meanwhile. Although in this experiment ORB need the most time, but if the restriction of the match decreasing, the order probably will like: KAZE>SIFT>ORB>FAST

This paper also found that despite the preference of the creator of the Lowe's test algorithm to increase the ratio size to adapt different algorithms, this article finds that moderately reducing the ratio size is beneficial for improving the stitching results in panorama assembly tasks based on static and small quantities of feature detection. Too high a ratio can lead to many incorrect matches, not only increasing the computational cost but also not yielding good outcomes. For example, with ORB and FAST, this could result in significant gaps, while the stability of algorithms like SIFT, SURF, and KAZE might be impacted.

So, the quality of panoramic image (in low quality matched points):

SIFT > SURF > KAZE > ORB > FAST

The effect of the two ratios:0.6>0.8

4 Conclusion

This paper primarily concentrates on generating panoramic images through automatic stitching of photographs using a homographic matrix. It compares the effectiveness and efficiency of various feature extraction algorithms in terms of feature point matching, such as SURF, SIFT, ORB, FAST, KAZE, and proposes potential applications. SIFT/SURF is the best suited for environments demanding high stability and precise feature detection, which need high computation resource. ORB is ideal for real-time or resource-constrained applications, including visual applications on mobile devices. KAZE is optimal for scenarios with rich image details requiring high-quality feature descriptions. FAST might be the most appropriate for instances where speed is crucial, such as in tracking and surveillance. Normally, as for stitching the static panoramic image, in term of the quality, this paper recommends the SIFT, SURF, KAZE and if the responsible time is a requirement and, FAST is the best resort. However, if it is possible to get enormous, detected matches and concern about the running time, ORB is a nice algorithm for that.

Feature detection technologies will progressively evolve, where more accurate detection and matching will yield more precise homography matrices. This advancement will enhance scientific and technological research in the field of computer vision. However, the panorama stitching currently available in market applications remains a static function, and there is still significant room for improvement in the efficiency of obtaining homography matrices.

Reference

1. M. Brown, R. Szeliski and S. Winder, "Multi-image matching using multi-scale-oriented patches," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, **1**, 510-517. (2005)
2. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision **60**, 91–110. (2004)
3. Bay, H., Tuytelaars, T., Van Gool, L. SURF: Speeded Up Robust Features. In: Leonardis, A., Bischof, H., Pinz, A. (eds) Computer Vision – European Conference On Computer Vision. 2006. Lecture Notes in Computer Science, **110**, 346-359. (2006)
4. E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, Barcelona, Spain, **3**, 2564-2571, (2011)
5. Alcantarilla, P.F., Bartoli, A., Davison, A.J. KAZE Features. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds) Computer Vision – European Conference On Computer Vision, 2012. Lecture Notes in Computer Science, **7577**, 214-227, (2012).
6. Rosten, E., Drummond, T. Machine Learning for High-Speed Corner Detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds) Computer Vision – European Conference On Computer Vision, 2006. Lecture Notes in Computer Science, **3951**, 430-443, (2006).
7. Harris, C. G., & Stephens, M. J. A combined corner and edge detector. Alvey vision conference, **19**, 189-192, (1998).
8. V. Vijayan and P. Kp, "FLANN Based Matching with SIFT Descriptors for Drowsy Features Extraction," 2019 Fifth International Conference on Image Information Processing (ICIIP), Shimla, India, 600-605 (2019).
9. Multiple View Geometry in Computer Vision Second Edition, Richard Hartley and Andrew Zisserman, Cambridge University Press, March, 23-24 (2004).
10. Martin A. Fischler, Robert C. Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, **24**, 381-395.