Comparing Stitched Panoramas using Variation of Feature Detection Techniques

Computer Vision CSCI0-4220

Shayne Taylor
100552047
Ontario Tech University
shayne.taylor@uoit.net

Matt McColeman
100525235
Ontario Tech University
matthew.mccoleman@uoit.net

ABSTRACT

Feature detection is a process which is critical to many computer vision applications. In particular, with image stitching it can make the difference between an image which has been stretched and is now distorted, or is shifted too far in one direction. To make a good quality panoramic image it is important to have a feature detector which can accurately identify features which can be mapped. It has been shown that Scale-Invariant Feature Transformation(SIFT) feature detection works best for most applications [1], it is worthwhile to see how other feature detectors work in different conditions. In this paper we will be exploring which pictures the following features detectors work well in. The following feature detectors will be explored: ORB, SURF, BRISK, AKAZE, and STAR feature detector in combination with BRIEF descriptor.

1. INTRODUCTION

Image stitching is a process in which two images which share content are combined together to make a larger, more complete image. In the current implementation, image stitching is accomplished by using Scale-Invariant Feature Transformation (SIFT) or another feature detection algorithm to detect local features. This is done on both photos which are going to be combined, once completed, the local features are then used in a Brute Force, K Nearest Neighbours algorithm. This will compute the closest points to all of the local features, which will help when trying to identify which features match from each image. Now that local features have been identified the ratio test based on distance will be applied to filter out some bad matches on the local feature set. An acceptable set of local feature points have been identified, a homographic matrix can be computed. Finally once the homographic matrix has been found the images can then be modified to align the features.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. RELATED WORKS

There are a multitude of different approaches to solving the problem of image stitching. There are many challenges and each technique is better optimized for solving certain issues regarding image stitching. One obvious challenge is how to determine the key features in the image such as the tip of a mountain. Another challenge is how to align the images before integrating them together. There are many ways to approach the challenge of unifying the two images to appear as one. Often images are viewed on regular shaped screens so images are often cropped. How to crop the image appropriately is a challenge. To further extend the issue of regularizing the image one could look into ways to preserve image information while achieving this. One such method to overcome this challenge is to use image synthesization to keep the contents of all the photos taken while filling in the area around the image to form a regular shape. Lastly a challenge could be to create a robust system that filters out bad sets of images that are not suitable to be stitched together thus maintaining the legitimacy of the programs results. Creating a smooth transition between stitched images is paramount in image stitching. Gao et al. [14] attempted to use two homographies to achieve nonlinear alignment. In this case the scene is modeled by dominant distant and ground planes. Unfortunately this method is limited to having no local perspective variations. Brown et al. [2] experimented using a camera that only rotates around its optical center where the images are shot from. This means the images are shot from the same point and are nearly planar. His experiments use a single homography to align multiple images creating a fully automatic panoramic image stitching technique.

To accomodate for less controlled cases where images are not taken under ideal circumstances, other techniques must be used to improve alignment. Zaragoza et al. [15] proposed as-projective-as-possible (APAP) warping. APAP uses warping by Moving Direct Linear Transformation (DLT) which can seamlessly align images that have differing projective models. This technique works for cases more common by images taken by the general public. It allows local non-projective deviations and handles global perspectives. Chang et al. [16] proposed a method for parametric warping which combines similarity and projective transformation. This in combination with APAP significantly improved results of their method which had reduced distortions. Chen et al. [18] further implemented natural image stitching with global similarity prior. Their selection scheme automatically deter-

mines the proper rotation and global scale for each image. Elimination of stitching artifacts which use local alignment adjustment methods exist. Zhang et al. [17] proposed a method to stitch images with large parallax using the observations that overlapping regions need not be perfectly aligned. Li et al. [19] proposed a robust elastic warping for parallax-tolerant image stitching. To remove incorrect local matches the method used a Bayesian model to ensure robust alignment.

A problem with resulting images is that not all are regularly shaped. This becomes a problem and most devices have regular shaped screens. He et al. [20] proposed a warping method to produce rectangular images from the stitched panoramas that is content-aware. The warping method has significantly improved results that rectify irregular boundaries caused by casual camera movements and by projections. This is an effective two step method that results in a regular shaped resulting image. However, Zhang et al [17] proposed a unified framework that outputs a stitched regular shaped image. In order to account for information loss the method constructs a piecewise rectangular panoramic image that can be cropped by the end user to obtain a rectangular image.

3. BACKGROUND

3.1 Speeded Up Robust Features(SURF)

SURF is a scale and rotation invariant feature detector. There are 2 main steps for SURF to find features. The first step identifies key points, such as edges, and corners. Second it will identify the neighbour of every point of interest and load it into a descriptor vector. Finding descriptors for the key points is important, because if a key point's descriptors is not something that can be found repeatedly, then it is not likely that the key point will be able to be matched [9].

3.2 Accelerated-KAZE (AKAZE)

AKAZE is based on another feature detection algorithm, KAZE. Both KAZE and AKAZE use nonlinear diffusion filtering. AKAZE detection uses the determinant of the hessian blob like SURF, but also uses scharr filters to help improve the rotation invariance. AKAZE finds descriptors using modified local difference binary algorithm. With all of these features it makes AKAZE invariant to scale and rotations [7].

3.3 STAR Feature Detector and Binary Robust Independent Elementary Features (BRIEF) Descriptor (S-B)

S-B is a unique feature detection algorithm when compared to the other algorithms used in this paper. This is because all of the other algorithms used find both the keypoints and the descriptors, while S-B uses STAR to find key points, and then uses BRIEF to find descriptors. STAR is based on Centre Surround Extrema (CenSurE) feature detector[13].

3.4 Binary Robust Invariant Scalable Keypoints(BRISK)

BRISK uses 2 main steps for feature detection. First key points are detected, this is done by making an image pyramid and using Features from Accelerated Segment Test (FAST), to identify edges and corners. It uses a image pyramid so it can identify key points at any size, making this a scale invariant algorithm. Next BRISK will identify keypoint descriptions, to do this 2 main things will be detected, first is the direction of the key point. To find the direction of the key point, the neighbouring points will be examined. BRISK will also then look for brightness comparisons between neighbouring points which are very useful in matching key points [6].

3.5 Oriented Fast and Rotated BRIEF(ORB)

ORB is a combination of Features from Accelerated Segment Test (FAST), and BRIEF. It uses the FAST corner detection methods, and then the Harris Corner score to find quality points. It also uses the BRIEF description methods. Due to BRIEF's description methods being unstable a modified descriptor for BRIEF is used. All of these features help make ORB invariant to both scale and rotations [8].

4. METHODOLOGY

4.1 Scale-Invariant Feature Transformation(SIFT) Feature Detection

SIFT feature detection is an algorithm used to detect and describe features in images. SIFT works by using a histogram of local gradient orientations. The gradients are then copied into a different orientation plane and blurred. The newly oriented and blurred image is then used to make another histogram of local gradients. Due to this method of feature detection the features typically are more stable for matching, because there is no variance due to scale [2].

4.2 K-Nearest Neighbor (KNN)

KNN was used in combination with Brute Force to identify the similarities between the images. KNN is an algorithm which is used for regression and classification. It will attempt to determine which local features that were found by SIFT match with the corresponding features from the other image. It does this by analyzing the local feature and finding the K (in the example k=2) nearest local features to see if we can match those in the other image. Once it has a list of features, whos k nearest neighbours match up on both images it will give us a list of points which we can be used for homography.

4.3 Ratio Test

With the list generated by the KNN algorithm we could make a homography matrix, but it is likely that the list is not perfect, and there will be some poorly matched features. For this reason before the homography matrix is computed, the KNN list will be refined. To achieve this the ratio test will be applied to the KNN list, this is done by taking a local feature and determining the distance from its first closest neighbour and then the second closest neighbour, whichever is closest will then be appended to a new list which contains only the good matches [2].

4.4 Homography

Once the list of matches made by the KNN algorithm has been created from the local features, then the list can be used to create a homography matrix. To create a homography matrix, it requires 4 points. When there are N sets of corresponding coordinates (x_i, y_i) and (x'_i, y'_i) between two pictures which are approximately related by an unknown homography H, as seen below.

$$w \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

These are the points determined by the feature detection. We want to solve for H by rewriting the equation as Ah=0.

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' & -y_1' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i & y_i & 1 & 0 & 0 & 0 & -x_ix_i' & -y_ix_i' & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_iy_i' & -y_iy_i' & -y_i' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx_n' & -y_nx_n' & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_ny_n' & -y_ny_n' & -y_n' \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}}_{\mathbf{h}} = \mathbf{0}$$

Using 4 corresponding points (x_i, y_i, x_i', y_i'), i =1,....,4 we obtain a system of 8 homogenous equations with 9 unknowns. Finding the null space of the system of equations gives a homography H that puts the points into exact correspondence. So we have 9 vectors each with 8 elements. These vectors are not linearly independent and so there is a linear combination of them that sums to the zero vector. With this we find H by solving for the null space of the matrix A.

4.5 Image Stitching

Now that the homography matrix has been found and the list of good features has been made, the images can now be combined together. As simple as that seems it requires more work than would be expected. This part of the process is one of the most important, because if the image is improperly placed then it can create a bad quality image, with tearing, and missing content. This problem was not handled in an optimal solution, but it does produce acceptable images with some minor tearing, but the content is not missing in most cases. This was accomplished by calculating a new location for each pixel, this works by taking both the x and y coordinate from the image being warped, it then will use the following equation to determine the new x and y coordinate for the pixel.

$$\mathtt{dst}(\mathtt{x},\mathtt{y}) = \mathtt{src}\left(\frac{\mathtt{M}_{11}\mathtt{x}+\mathtt{M}_{12}\mathtt{y}+\mathtt{M}_{13}}{\mathtt{M}_{31}\mathtt{x}+\mathtt{M}_{32}\mathtt{y}+\mathtt{M}_{33}}, \frac{\mathtt{M}_{21}\mathtt{x}+\mathtt{M}_{22}\mathtt{y}+\mathtt{M}_{23}}{\mathtt{M}_{31}\mathtt{x}+\mathtt{M}_{32}\mathtt{y}+\mathtt{M}_{33}}\right)$$

Figure 1: Pixel Transformation Formula

In this equation src is the source image, M is the homography matrix, and the x and y which are being multiplied by the homography matrix are the pixels location from the original image. Once the equation has been completed it will return the new location for the pixel. In order to avoid stitching artifacts the function iterates through the pixels in the destination image returning information from the source image. This avoids problems that might occur if we were to map based on source pixels to destination pixels which could leave some pixels as void in the destination image.

5. RESULTS

5.1 Compare

To compare the images and decide which feature detector has done the best job, the panoramic images which have been created can be compared using the tool in linux, compare. Using the following code:
compare -compose src <img1> <img2> <outputImage>
This will allow us to generate an image that will print out all of the similarities in white and all of the differences in red. Using these new images we can compare SIFT, which can be considered the best feature detection available, to all of the other feature detection methods to decide which one is the most accurate. Along with this we can also analyze the feature matching by connecting the features which are the same from both images with lines. For some cases this will not be useful because there are so many features detected, but in some cases it will be clear which features it is mapping. Finally a visual inspection of the panoramic pictures can be compared and manually decided which is the best.

Figure 2: SIFT panoramic image

5.2 SIFT vs SURF

Figure 3: SURF panoramic image

SURF vs SIFT. Comparing SIFT to SURF is an interesting case, because SURF is based on SIFT. When looking at the feature mapping, it is very similar but looks like SURF may have more features matched. This does not necessarily mean that it is a better feature detector, it could have more false positives. When doing a manual inspection of the panoramas created by SIFT and SURF they appear almost the exact same. Upon a close inspection the main difference seemed to be where the land connects to the water, SURF was slightly off when matching the land from both images. Finally comparing the difference image shows that the panoramic image is almost the exact same. This was definitely the closest to being the same, but there were some differences. Mainly around where the water meets the land as pointed out before.

5.3 ORB vs SIFT

ORB vs SIFT. ORB found one of the least amounts of matches out of all of the feature detectors. From this fact alone it can be assumed that the panoramic image will be

Figure 4: SIFT vs SURF Difference

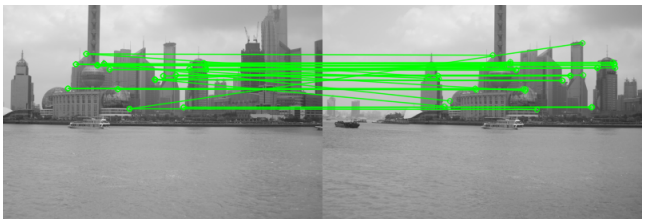

Figure 5: ORB key point matches

of lower quality than SIFT. But more inspection should be done because SIFT could just have more false positives. The panoramic picture was actually very close, but it appeared to be shifted up too high. ORB when compared to SIFT using the compare function, was quite different, especially the bottom half of the picture which when visually inspected was shifted up too high. ORB was consistently one of the worst feature detectors, from all of the images we tested on.

5.4 BRISK vs SIFT

Figure 6: SIFT vs BRISK Difference

BRISK vs SIFT. BRISK gave consistently impressive results, when testing with any images. When comparing the matches found by BRISK to SIFT, it was apparent that BRISK had many less matches. From this it was assumed that the quality of the panorama was going to be lower. Surprisingly when comparing the panoramas BRISK is very similar to SIFT, and is one of the best panoramas that was generated. When comparing the image difference BRISK seemed to be the best performing other than AKAZE, but AKAZE only out preformed BRISK in very few images.

5.5 B-S vs SIFT

B-S vs SIFT. B-S was consistently the worst performing feature detector which was used. This could be because STAR and BRIEF dont work together as well as the feature detectors which had both the features and descriptors found at the same time. When comparing the features found by B-S and SIFT, B-S had very few matches, which as described with BRISK may not mean the quality will suffer.

Figure 7: SIFT vs STAR feature detector and BRIEF descriptor Difference

When comparing the panoramic image to SIFT's panoramic it was very clear that the quality was not very high. B-S's panoramic image was shifted much too high and had a improper rotation applied to it. The image compared to SIFT returned a result that was almost completely different, as shown in figure

5.6 AKAZE vs SIFT

Figure 8: AKAZE panoramic image

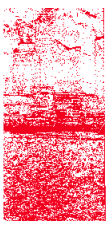

Figure 9: SIFT vs AKAZE Difference

AKAZE vs SIFT. AKAZE was an interesting case to try, as with some images it could not compute enough matches when using the KNN algorithm with k=2. To fix this the k value has been set to 1, and the ratio test could not be used. This makes AKAZE likely to be less accurate than the other algorithms which could use k=2 and could use the ratio test. AKAZE recognized a lot more matches than SIFT, which again could mean its higher quality, or it could mean it has recognized more false matches. Looking at the panoramic image produces by AKAZE it was shockingly accurate, especially because it was run only using KNN with k=1 and could not use the ratio test at all. With images that are well lined up without much rotation needed, AKAZE performed very well, and in some cases would be the best preformed feature detector. To continue with the example Shanghai picture, AKAZE found a less matches than SIFT but more matches than B-S and ORB. When comparing the panoramic image, AKAZE's panoramic was almost the exact same as SIFT, and was hard to pick out any differences.

Finally comparing the image difference showed that the majority of the image was the same as SIFT.

6. CONCLUSION

When looking at the results obtained from the comparisons, and panoramic pictures, it is clear that the best consistent performance came from SIFT. This was assumed from the beginning because SIFT has been shown to perform the best in most cases which is why SIFT [1] was used for all of the comparisons. The other feature detectors which were tested can be ranked in the following order:
1.SURF
2.BRISK
3.AKAZE
4.ORB
5.B-S
These rankings do not always hold true, particularly in the case with AKAZE. It was found that AKAZE seemed to outperform SURF and BRISK when images were lined up well and needed very little rotation and warping. Without having prior knowledge of how to line the picture up SURF as well as BRISK would be a better feature detector to use, because they consistently performed well on all images. ORB and B-S also can be swapped depending on the rotations needed. Both feature detectors usually resulted in a much more warped image, and in general low quality panoramas, as such they are not very suitable for use. In conclusion SIFT is the most well rounded feature detector for image stitching.

7. REFERENCES

[1] Karami, Ebrahim and Prasad, Siva and Shehata, Mohamed *Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images arXiv preprint arXiv:1710.02726*,2017
Lowe, David G *Distinctive image features from scale-invariant keypoints International journal of computer vision*,2004

[2] Brown, Matthew and Lowe, David G *Automatic panoramic image stitching using invariant features International journal of computer vision*,2007

[3] Szeliski, Richard and others *Image alignment and stitching: A tutorial Foundations and Trends in Computer Graphics and Vision*,2007

[4] Calonder, Michael and Lepetit, Vincent and Strecha, Christoph and Fua, Pascal *Brief: Binary robust independent elementary features European conference on computer vision*,2010

[5] Leutenegger, Stefan and Chli, Margarita and Siegwart, Roland *BRISK: Binary robust invariant scalable keypoints 2011 IEEE international conference on computer vision (ICCV)*,2011

[6] Tareen, Shaharyar Ahmed Khan and Saleem, Zahra *A comparative analysis of sift, surf, kaze, akaze, orb, and brisk 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*,2018

[7] Rublee, Ethan and Rabaud, Vincent and Konolige, Kurt and Bradski, Gary R *ORB: An efficient alternative to SIFT or SURF ICCV*,2011

[8] Bay, Herbert and Ess, Andreas and Tuytelaars, Tinne and Van Gool, Luc *Speeded-up robust features (SURF) Computer vision and image understanding*,2008

[9] Brandt, J. *Transform coding for fast approximate nearest neighbor search in high dimensions IEEE Conf. on Computer Vision and Pattern Recognition*,2010

[10] Mistry, Darshana and Banerjee, Asim *Comparison of Feature Detection and Matching Approaches: SIFT and SURF GRD Journal*,2017

[11] Per Rosengren *Calculating homography http://www.csc.kth.se/ perrose/files/pose-init-model/node17.html*,2007

[12] mcgill *Homography http://www.cim.mcgill.ca/ langer/558/2009/lecture19.pdf*

[13] ButterCookies *Star Feature Detector http://experienceopencv.blogspot.com/2011/01/star-feature-detector.html*

[14] Junhong Gao and Seon Joo Kim and Brown, M. S. *Constructing Image Panoramas Using Dual-homography Warping Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*,2011

[15] Zaragoza, Julio and Chin, Tat-Jun and Brown, Michael S and Suter, David *As-projective-as-possible image stitching with moving DLT Proceedings of the IEEE conference on computer vision and pattern recognition*,2013

[16] Chang, Che-Han and Sato, Yoichi and Chuang, Yung-Yu *Shape-preserving half-projective warps for image stitching Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*,2014

[17] Zhang, Fan and Liu, Feng *Parallax-tolerant image stitching Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*,2014

[18] Chen, Yu-Sheng and Chuang, Yung-Yu *Natural image stitching with the global similarity prior European Conference on Computer Vision*,2016

[19] Li, Shiwei and Yuan, Lu and Sun, Jian and Quan, Long *Dual-feature warping-based motion model estimation Proceedings of the IEEE International Conference on Computer Vision*,2013

[20] He, Kaiming and Chang, Huiwen and Sun, Jian *Rectangling panoramic images via warping ACM Transactions on Graphics (TOG)*,2015