Real Time Video Stitching by Exploring Temporal and Spatial Features

Shaoen Wu, Kelly Blair, Junhong Xu, Shangyue Zhu, Hanqing Guo
Department of Computer Science
Ball State University
Muncie, IN 47306
swu,kablair,jxu7,szhu,hguo@bsu.edu

Kai Wang
Department of Computer Science
Georgia Southern University
Statesboro, Georgia 30460
kwang@georgiasouthern.edu

Lei Chen
Department of Information Technology
Georgia Southern University
Statesboro, Georgia 30460
lchen@georgiasouthern.edu

ABSTRACT

Although image stitching has been investigated for years, realtime video stitching still lacks of efficient methods to meet the required frame rate for satisfactory human vision experience. This work proposes efficient video stitching solutions by exploiting both temporal and spatial features among video frames. As a result, the stitching speed is significantly improved with two techniques by exploiting: (1) the dimmension of distance (spatial) by focusing only on the region of frame overlap and (2) the dimmension of time (temporal) by reusing homography information across multiple frames. Based on these two techniques, this paper presents three solutions to determine submiages for rapid stitching the video frames from side-by-side cameras. This work implements these solutions into a video stitcher. The evaluation over video streams shows that the proposed solutions can stitch the video at 6.5 frames per second (fps) in contrast to 1.5 fps in conventional imaging stitching approaches, which is over 400% improvement on stitching speed performance, but at the cost of a marginal drop in accuracy.

KEYWORDS

Interest point and salient region detections, Matching, Appearance and texture representations

ACM Reference format:
Shaoen Wu, Kelly Blair, Junhong Xu, Shangyue Zhu, Hanqing Guo, Kai Wang, and Lei Chen. 2017. Real Time Video Stitching by Exploring Temporal and Spatial Features. In *Proceedings of 10th EAI/ACM International Conference on Mobile Multimedia Communications, Chongqing, China, July 2017 (MOBIMEDIA'17),* 6 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Image stitching is the process of combining multiple images to create a panorama, which is done in a series of procedures called the image stitching pipeline [1–3]. An image stitching algorithm first searches for distinct feature points in the images. Unusual points and their surrounding points are identified as a feature descriptor [4, 5]. For example, a white pixel surrounded by black pixels is highly unique, but a cluster of green pixels is not. Then, the stitching algorithm searches for identical features among different images. The images will be matched together upon these features. Feature descriptors are compared against each other with a forgiveness margin to allow small deviations. Feature points without a match within this margin are discarded. Flann based matching is reported to be the fastest currently available matching algorithm [6]. Based on the locations of the matched points, the program is then able to calculate a 3D model of the area. This 3D model is referred to as the image homography [7]. The images can then be warped and positioned with respect to the 3D model [8, 9]. This process involves modifying the image matrices to generate the stitched image. Normally the stitching will generate a visible seam where two images are combined.

To address the problem of a visible seam in a stitched image, a procedure of image blending is needed to smooth out this seam so it is not noticeable. Image feathering, or setting a gradient of transparencies, is one common approach [10–12], which "feathers" the transparencies of the images close to the seam. In particular, the small overlap region of two images will be processed with image *alpha*, which relates to the transparency, is set close to 0.5. When the *alpha* value approaches to 1, the image becomes closer to its original one. The alphas are usually set to one within a small area, so only the region surrounding the overlap is blended [13]. Today, OpenCV [14] can fully automate this process of stitching with APIs of matured stitching algorithms that implement the feature point identifying, matching, image combining and blending.

Static image stitching is a process that has been investigated for decades. Today image stitching is available in a wide variety of commercial applications. Although there are many algorithms available for conventional image stitching [15–17], they do not work well for video stitching. This is because time is not a constraint when stitching a static images. However, video stitching, especially realtime, is very challenging because the stitching speed has to meet the basic human visual experience requirements. For example, to stitch a video feed in real time, the stitching must be done quickly enough to produce an attractive frame rate of 15+ fps. Currently, the time to stitch two static images using the openCV library is between 1 and 3 seconds, equivalent to 0.33 and1 fps. Therefore, *in order to achieve real time video stitching, the stitching speed must be dramatically improved.*

MOBIMEDIA'17, Chongqing, China

DOI: 10.1145/nnnnnnn.nnnnnnn

DOI 10.4108/eai.13-7-2017.2270086

In this work, to improve the stitching speed, we propose three solutions based on two approaches to exploit the temporal and spatial correlations among video frames. The first approach reduces the amount of pixels needed for the homography calculation that takes up the majority of the stitching time, therefore improving the speed of this calculation will substantially improve the overall speed of the stitch. The other approach reuses information calculated in previous frames. Because individual successive frames do not change very much, it is very likely that information taken from a previous frame can be reused for a short period of time.

In the rest of this paper, Section 2 talks of the related work in literature. We discuss the detail solutions in Section 3. The evaluated performance is shown in Section 4. This paper is concluded by Section 5.

2 RELATED WORK

Image stitching or mosaic has been investigated for years and has come out with a variety of solutions [18–23]. It normally consists of two phases: alignment (a.k.a registration) and composition [24]. Image stitching schemes have been used in video stitching by simply treating a video frame as an image [25–27]. Different from the requirement in image stitching, the challenge in video stitching, especially in real-time cases, is: *the stitching speed has to match the expected frame speed for pleasant user visual experience*. Pal, Steedly and Szeliski [20] proposed to exploit the temporal characteristics existing in the video frames of a camera to expedite the stitching. Most available video stitching solutions [27–30] target at the successive video frames of a *single* mobile camera moving across a scene, for example, a person holds a camera and rotates 360^o. Stitching video streams from *multiple* cameras has also been studied [25, 26, 31–33], but these efforts focused on *statically* placed and well-calibrated cameras. The only known endeavor of stitching video streams from *multiple mobile* cameras was attempted by El-Saban, *et al.* [34], in which the mobile smartphones were used as the video capturing devices, but the users normally move in random patterns and they can not be forced to cooperate. Some recent work on realtime image stitching include [35–37].

3 TEMPORAL AND SPATIAL VIDEO STITCHING (TSVS)

In this section, we first present special correlations among successive video frames that can be exploited by video stitching, and then describe the design of our proposed temporal and spatial video stitching (TSVS) solutions.

3.1 Temporal and Spatial Correlations

Although each video frame can be considered as an image, video stitching has more information to exploit than image stitching. Imagine a pair video cameras that overlap over a portion of FoV (Field of View) as shown in Figure 1. After identifying the feature points of their first video frames in the overlapped regions, some of these feature points can be reused for successive stitching. First, because of the very short interval in taking successive frames in a video stream, e.g. 1/16 second in a 16 fps camera, it is of a great probability that these *temporally successive* frames taken at t_0 and t_1 from the same camera have a large portion of overlap, which we define as *temporal correlation*. Second, the two frames from these two cameras at time t_1 is very likely to overlap on a portion of the last overlap between the two frames taken at time t_0, which we define as *spatial correlation*. As shown on Figure 1, the feature points inside the central shaded area of $A'_2B'_1C_1D_2$ identified on two video frames expect to be patricianly inherited by successive video frames due to temporal and spatial correlation.

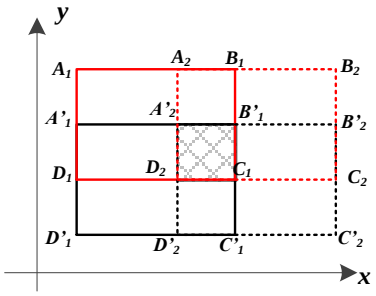

Figure 1: Temporal and Spatial Correlation

Homography Reduction: A key to the stitching speed is to reduce the number of pixels in the frames to be processed. There are two approaches to the pixel reduction. One option is to lower the image resolution, but at the potential cost of losing feature points. The more favored approach, however, is to focus on the overlapping region of the frames. Because the image stitching calculation focuses on feature points that fall into the overlap region, removing extraneous edges is unlikely to hurt the stitch integrity. Namely, rather than using the entire frame images to identify the feature points, we propose to focus on partial of a frame image, called subimage. Once if the image homography is completely calculated from subimages, it will be applied to the full images, and then an appropriate translation on the coordinates of x and y is performed.Therefore, it expects to significantly expedite the stitching process. We name this solution "homography reduction" and the challenge now is to determine the submiages of stitching frames. If a large subimage is used, we will unnecessarily waste time on identifying feature points in a region that does not really have. On the other hand, if a small subimage will lose some feature points and leads to the loss of accuracy of stitching results.

Coordinates Translation: Before we discuss the determination of the subimages, we present how the coordinate translation is performed with the assumptions that submiage homography is already known. Currently several algorithms can automatically calculate the x and y translation, as well as the overlapping region. With a series of tests to determine the ideal size of the subimage, we consistently found that values of $0.35x$ and $1.0y$ were the minimum needed to get reliable results. To handle the x and y coordinate translation, we tentatively stored a pair of x and y values based on the offset of the subimage from its initial position. For example, if a subimage used in homography calculation starts at an x coordinate of 100 px, the translation is to compensate the warped image by 100 px. Although it is an imperfect method, it is more accurate than the default offset of "0".

In the follow, we present three homography reduction solutions to the subimage determination: *fixed-point subimage determination, feature-point subimage determination,* and *seam subimage determination.*

3.2 Fixed-Point Subimage Determination

Suppose that we have two (left and right) frames to stitch as shown on Figure 2. Our initial approach is to simply take the rightmost half ($0.5x$, x) of the left image, and the leftmost half (0, $0.5x$) of the right image. This solution is simple, but every fast. It works efficiently on a pair of side-by-side cameras. Because it is sure that these images overlapped after the very first stitching with feature points already identified on previous frames, it is also certain that the portion of image surrounding the seam will appear in both images.

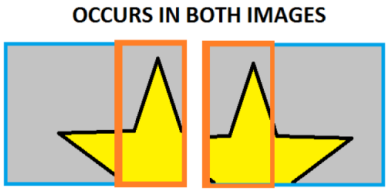

Figure 2: Fixed Point Subimage Determination

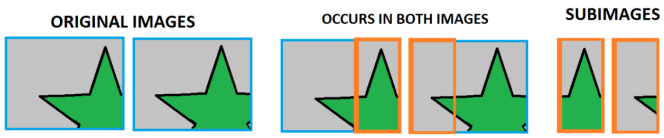

Figure 3: Problem of Fixed Point Subimage

Although the fixed-point subimage solution works for those images overlapped in the middle, which is the case where a pair of cameras set side-by-side, this simple solution does not work for other cases that the overlap is not in the middle. For example, as illustrated in Figure 3, the right side of the left image overlaps the middle part of the right image. With the fixed-point subimage solution, the results are completely unacceptable. More of such problems are demonstrated in Figure 4 with images from real life.

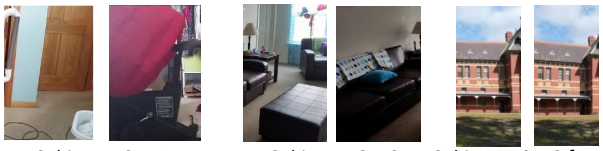

Figure 4: Problem of Fixed Point Subimage on Real Images

3.2.1 Feature Point Subimages Determination. Regardless of the settings of cameras, their feature points needed to be identified anyway as part of the stitching pipeline for the very first frames. Based on this observation, we propose the second subimage determination solution that finds a overlap region, i.e. subimage, based on the feature points of the two images. Whenever two matching feature points are identified within two frame images, it implies that those points represent part of the overlapping region, as illustrated in Figure 5. By stitching the current frames, the subimages can continuously be predicted for the next frames. However, for this solution to work effectively, three problems need to be addressed as discussed below.

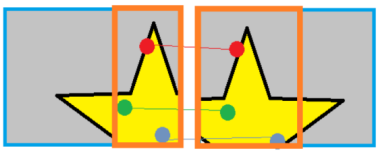

Figure 5: Features in Images and Their Subimages

Accumulative Errors: Because the feature point subimage determination process is self-contained, any error from the current stitching will accumulatively impact future frame stitching. To solve this issue, we occasionally re-run a full image stitch to discard the accumulative errors, or reset the subimage determination procedure.

Different Subimage Sizes: In order to stitch images, they need to be identical in size. The subimages returned by the feature point determination algorithm are almost never in the same size. We therefore choose the largest horizontal and vertical size (i.e. $\max(x_1, x_2)$ and $\max(y_1, y_2)$) among subimages, then transform the subimages into the size of $(\max(x_1, x_2), \max(y_1, y_2))$. Then, the stitching can work correctly.

Outliers: Since no algorithm can identify all the feature points with 100% confidence, the incorrectly identified false feature points will significantly hurt the accuracy of the stitching. To address the outlier problem of false feature points, we design the fuzzy algorithm to filter out the outlier feature points. Basically, the fuzzy algorithm uses the standard deviation to measure the location of the feature points to its surrounding neighbor feature points. If the deviation is larger than a threshold, the feature points will be classified as false points and will be discarded.

With a combination of the above schemes, a block diagram of the feature point subimage determination algorithm is shown in Figure 6.

3.3 Seam Subimage Determination

The third subimage determination is completely based on a stitching on full images. After a single stitching of both full frame images, the information on the region of overlap between these two images can be available. Specifically, based on the size of the stitched image, we can determine the region of overlap by eliminating those pixels that are not part of the stitched overlap. For example, if two images have an identical size of $1.0x$ and their stitched image has a size of $1.4x$, it can be estimated that the region of overlap in each image is about $0.6x$. This is because if the images did not overlap at all, the size of the stitched image would be $2.0x$ and any smaller size implies a region of overlap, namely subimage, that is equal in size in both images.

The subimage is calculated as shown in Figure 7. First, we label the left-bottom of the right image with the coordinates of ($X1$, $Y1$). We define the original image width IMAGE_WIDTH and define the

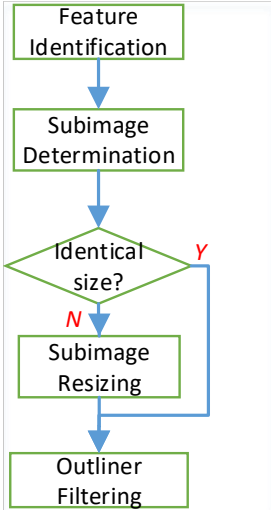

Figure 6: Feature Point Subimage Determination Block Diagram

width of subimage ROI_WIDTH. Because the frames from the cameras have the same length, this length will remain in the stitching. We only need to calculate ROI_WIDTH to determine the subimages, which is the "overlap", namely the difference in the sizes of X between the stitched and before-stitched images.

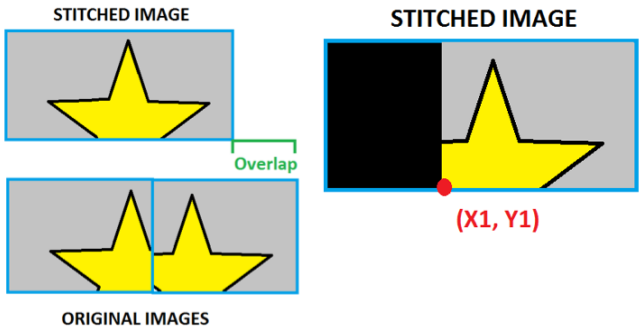

Figure 7: Seam Subimage Determination

The calculation is performed as follows. First, we transform all the stitched images to always be the same size. They are just superimposed on a large file with a black background. Then, we trace a line around the image on this background, and remember each point on that line. Next, the corners of each of the images are calculated. Finally, the corners of the image will be compared to the boundaries of the unwarped images. The subimage calculation is illustrated by a procedure below:

(1) For the right subimage, because it is always the leftmost portion of the entire right image, its X coordinate starts from X_r=0 and ends at X_r=ROI_WIDTH on the right image.
(2) For the left subimage, its X coordinate starts from the point of ROI_WIDTH away from its rightmost edge. Therefore, its X coordinate spans from X_l=IMAGE_WIDTH - ROI_WIDTH to X_l=IMAGE_WIDTH.
(3) For both subimages, we set their $Y = Y1 \times 0.5$ with the assumption that half of their pixels will be carried into the successive images to be stitched.

With the approach of seam subimage determination, we tested its performance against a few images and the results are shown in Figure 8. On the figure, each set shows the subimages of a pair of frames. We can observe that, although the subimages are not exactly identical, they do show a large region of overlaps.

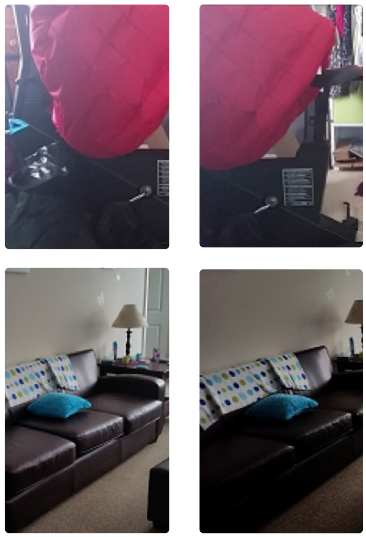

Figure 8: Seam Subimage Results

4 PERFORMANCE EVALUATION

We evaluate the performance of the subimage based stitching solutions by comparing to the literature solution implemented in OpenCV that we name it "Full Stitching". We first evaluate their stitching accuracy in the results and then compare their stitching speed that is the main focus of this work.

4.1 Visual Comparison

We first compare the visual stitched results between our solution of seam subimage determination and the full stitching with existing OpenCV algorithms. The comparison is shown in Figure 9. As we can observe from the figure, in image sets 1, 2, and 4, the stitching is nearly identical. Although the stitching based on seam subimage on the set 4 has alignment issues, those alignment issues are also present in the full stitch, which means the problem are not caused by the seam subimage determination.

4.2 Stitching Time

The time on stitching is critical in video stitching to satisfy the human visual experience. With the translation values and region of overlap calculated properly, stitching using subimages can significantly improve the stitching time. We evaluate the stitching time performance with subimages of a size of $0.35x$ and $1.0y$ of the original image. Our solution offers a 20.83% increase in speed over traditional approaches. However, this performance improvement

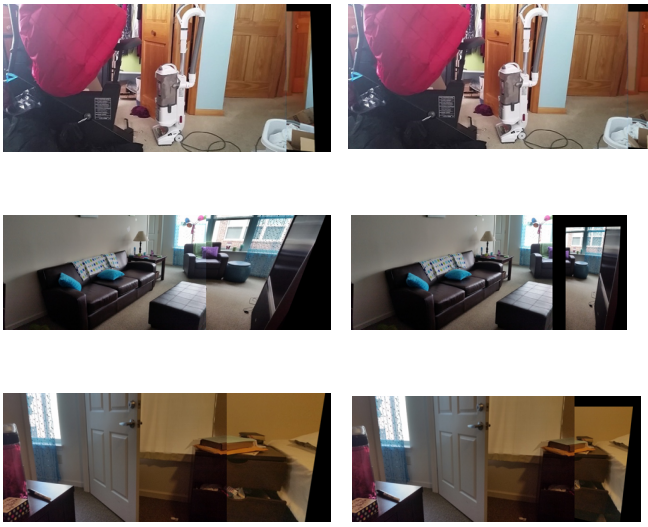

Figure 9: Visual Comparison

is at a minor drop in accuracy. The performance is summarized in Table 1.

Table 1: Full Image vs Subimage

Solutions	20 Stitchings	fps	hps
Full Image	13.79s	1.450	1.450
Subimage	11.41s	1.752	1.752

4.3 Multithreading Stitching

Because subimage-based solutions reuse the history stitching information and a video stream often have correlated successive frames, it is ideal to use multithreading to stitch multiple frames in a parallel style. We design a two-thread solution in which one thread is dedicated to calculating the homography based on the subimage solutions and the other thread focuses on stitching images based on the last available homography.

Table 2: Full Image vs Multithreading in Stitching Speed

Solutions	100 frames	28 homographies	fps	hps
Single Thread	68.95s	19.31s	1.450	1.450
Multithreading	15.42s	14.70s	6.485	1.904

We evaluate the performance on 100 pairs of image frames and the calculation of 28 homographies. The results are shown in Table 2. From the results, multithreading combined with subimage information reuse offers a substantial speed improvement. It boasts 6.485 fps and 1.904 Homography Per Second (hps). This is equivalent to 447% of the fps performance of traditional methods. We also summarize the comparison of the performance of traditional stitching, single-thread stitching and multithread stitching in Table tab:sum that shows the performance gain with subimage based either single-thread or multithread solutions. However, the improvement of homography calculation is not significant because only it is still performed by only one thread.

Table 3: Full Image vs Single Thread vs Multithreading

Metric	Full Image	Single Thread	Multithreading
fps	1.450	1.752	6.485
fps gain	0	20.83%	347.24%
hps	1.450	1.752	1.904
hps gain	0	20.83%	33.31%

We also observe a downside that, while the the image homography can properly warp the images, it is more difficult to keep them aligned. A noticeable offset between the frames is thus observed. If the cameras shake, two video feeds appear independent, but ff the videos were taken on a tripod, this method will work effectively.

5 CONCLUSION

This work proposes a category of video stitching solutions to improve the stitching speed with less time to meet human visual experience. These solutions are based on calculating and reusing the subimages carried in successive video frames temporally and spatially. Overall, all approaches show improvements in speed, but with a varying degree of accuracy loss. When these solutions are implemented with a carefully designed multithreading framework, they can improve the stitching time by about 4.5 times. Our future work is to design a hybrid solution to improve the stitching accuracy by implementing blending algorithms, fully automating seam detection, and implementing feature point tracking to dynamically adjust the image translations, while attaining the high performance on stitching time.

ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation under Grant No.#1408165.

REFERENCES

[1] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1):59–73, 2007.
[2] Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.
[3] Chia-Yen Chen and Reinhard Klette. Image stitching—comparisons and new techniques. In *International Conference on Computer Analysis of Images and Patterns*, pages 615–622. Springer, 1999.
[4] Mark S Nixon and Alberto S Aguado. *Feature extraction & image processing for computer vision.* Academic Press, 2012.
[5] Linda Shapiro and George C Stockman. Computer vision. 2001. *ed: Prentice Hall*, 2001.
[6] Marius Muja and David G Lowe. Flann, fast library for approximate nearest neighbors. In *International Conference on Computer Vision Theory and Applications (VISAPP'09)*, volume 3. INSTICC Press, 2009.
[7] David Kriegman. Homography estimation. *Lecture Computer Vision I, CSE A*, 252, 2007.
[8] CorrMap. http://www.corrmap.com/features/homography_transformation.php.
[9] Junbin Gong, Chenlin Zheng, Jinwen Tian, and Dingxue Wu. An image-sequence compressing algorithm based on homography transformation for unmanned aerial vehicle. In *Intelligence Information Processing and Trusted Computing (IPTC), 2010 International Symposium on*, pages 37–40. IEEE, 2010.
[10] Heung-Yeung Shum and Richard Szeliski. Panoramic image mosaics. Technical report, Citeseer, 1997.
[11] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 251–258. ACM Press/Addison-Wesley Publishing Co., 1997.
[12] Matthew Brown, David G Lowe, et al. Recognising panoramas. In *ICCV*, volume 3, page 1218, 2003.
[13] Richard Szeliski. *Computer vision: algorithms and applications.* Springer Science & Business Media, 2010.

[14] OpenCV. http://opencv.org.

[15] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[16] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[17] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[18] M. Brown and D.G. Lowe. Recognising panoramas. In *Proceedings. Ninth IEEE International Conference on Computer Vision*, volume 2, pages 1218 –1225, oct. 2003.

[19] Matthew Brown and David Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

[20] D. Steedly, C. Pal, and R. Szeliski. Efficiently registering video into panoramic mosaics. In *Tenth IEEE International Conference on Computer Vision*, volume 2, pages 1300 –1307, oct. 2005.

[21] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM Comput. Surv.*, 24(4):325–376, December 1992.

[22] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 510 – 517, june 2005.

[23] Richard Szeliski. Image alignment and stitching: a tutorial. *Journal of Foundations and Trends in Computer Graphics and Vision*, 2(1):1572–2740, 2006.

[24] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2011.

[25] Lin Zeng, Dexiang Deng, Xi Chen, and Yunlu Zhang. A self-adaptive and real-time panoramic video mosaicing system. *Journal of Computers*, 7(1):218–225, 2012.

[26] Bin He, Gang Zhao, and Qifang Liu. Panoramic video stitching in multi-camera surveillance system. In *25th International Conference of Image and Vision Computing New Zealand (IVCNZ)*, pages 1 –6, nov. 2010.

[27] Wei Zeng and Hongming Zhang. Depth adaptive video stitching. In *Eighth IEEE/ACIS International Conference on Computer and Information Science2009.*, pages 1100 –1105, june 2009.

[28] Aseem Agarwala, Ke Colin Zheng, Chris Pal, Maneesh Agrawala, Michael Cohen, Brian Curless, David Salesin, and Richard Szeliski. Panoramic video textures. *ACM Transaction on Graphics*, 24(3):821–827, July 2005.

[29] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transaction on Graphics*, 22(3):277–286, July 2003.

[30] Chaokui Li, Gang Yang, Jun Wu, and Fang Wang. The uav video image stitching based on improved moravec corner matching method. *International Journal of Remote Sensing Applications*, 2(1):41–44, 2012.

[31] Ashish K. Banerji, Kannan Panchapakesan, and Kumar Swaminathan. Stitching of h.264 video streams for continuous presence multipoint videoconferencing. *Journal of Visual Communication and Image Representation*, 17(2):490–508, 2006.

[32] A. Majumder, W.B. Seales, M. Gopi, and H. Fuchs. Immersive teleconferencing: a new algorithm to generate seamless panoramic video imagery. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 169–178. ACM, 1999.

[33] U. Neumann, T. Pintaric, and A. Rizzo. Immersive panoramic video. In *Proceedings of the eighth ACM international conference on Multimedia*, volume 2000, pages 493–494, 2000.

[34] M. El-Saban, M. Izz, and A. Kaheel. Fast stitching of videos captured from freely moving devices by exploiting temporal redundancy. In *17th IEEE International Conference on Image Processing (ICIP)*, pages 1193 –1196, sept. 2010.

[35] JD Westwood et al. Real-time image mosaicing for medical applications. *Medicine Meets Virtual Reality 15: In Vivo, in Vitro, in Silico: Designing the Next in Medicine*, 125:304, 2007.

[36] Markus Quaritsch, Robert Kuschnig, Hermann Hellwagner, Bernhard Rinner, A Adria, and U Klagenfurt. Fast aerial image acquisition and mosaicking for emergency response operations by collaborative uavs. In *Proceedings for the International ISCRAM Conference*, pages 1–5, 2011.

[37] Marcus Arthur, Raid AL-TAHIR Barbados, and Dexter DAVIS. Rapid processing of unmanned aerial vehicles imagery for disaster management. In *FIG Working Week*, 2012.