Image Stitching

Joseph Johnston (Student ID: 201824080006)

Max Benjamin Hort (Student ID: 201825000101)

November 2018

1 Introduction

Recent advances in technology lead to an improvement in cameras. Additionally, it is very common that phones nowadays include a camera. Not only does this lead to an increasing amount of pictures that can be found online, but also to an increasing amount of pictures individuals take. If multiple pictures of the same object are being taken, one might desire a larger combination of the pictures, such as one may get by stitching them together into a panorama.

The use and relevance of panoramas can be seen when investigating modern cameras. Modern cameras often offer a feature for taking panoramas. However, not everyone has access to these methods, or one may decide to create panoramas at a later point of time, after the pictures are taken. This introduces the need for other methods to create panoramas from static photos.

As images are usually represented by matrices, image stitching provides an interesting framework for applying matrix operations and algorithms. We are therefore concerned with the questions of how to describe image features and how to perform image stitching with images when those images are represented as matrices.

2 Related Work

Using several images to create panoramas have existed for several decades. There is an enormous amount of researches which are being carried out regarding image stitching. Image stitching is built upon several algorithms. Generally, these algorithms require a lot of computation and memory. but recently there are some applications which could run on mobile phones thanks to efficient and optimized algorithms. [1] Mainly, there are two different methods of image stitching which are domain based registration and feature-based registration. [2]

Feature detection is a primary procedure in image processing such as image stitching, target tracking or motion estimation. [3] There are several methods of feature detection. some of them are Harris, Susan,SIFT and SURF feature point which is a faster variant of SIFT [4] The Harris corner detector is robust against the change in rotation, illumination, visual angle, and noise. and it achieves higher speeds due to its simpler calculation methods. However, this operator fails when disposing of images with larger scale changes [5] but still Harris corner detector is one of the mostly used corner detectors in the field.

SIFT is robust enough to detect features even in microscopic images. Moreover, its already employed in several commercial software. [6] In addition, Mikolajczyk performed a study which provides an extensive comparison of many key point descriptors and according to his research SIFT performs well in many scenarios. [7] However, SIFT has shortcomings of being sensitive to the changes in viewing angle, evolving slow detection and registration. It's impossible to state a specific feature detector which works for every case.

After the necessary features are being extracted from the images. the image transformation is being carried out using RANSAC algorithm. RANSAC is an algorithm which is capable of matching feature points with noise and has good

robustness. It performs well even in situations where more than 50 percent of the data points are outlier. [8]

3 Method

Image stitching in general runs through the following steps:

1. Feature Detection and Description
2. Feature Matching
3. Computation of Transformation
4. Stitching of Images

Section 4 illustrates feature detection and description using Harris corners. An explanation of feature matching and an outlook on a modern approach (SIFT) will be given. Afterwards, Section 5 describes different types of image transformations and how transformations can be computed, utilizing RANSAC to improve the quality. Lastly in Section 6, an experiment has been conducted to perform image stitching in practice.

4 Feature Detection and Description

If you were to stitch to images, it would be supportive to find unique features of the images that stand out, to match them. For example, if the images show a mountains, one might choose these features to be the peaks of the tallest mountains because they are easily identified in both images. This step can be denoted as *feature detection*. The second step is to describe the features such that identical features between images can be matched. One of the peaks might be described as having snow, or having a sharp tip. This step is called *feature*

description. Finally, feature descriptors will be compared to match features.If two peaks are described identically, they can be matched and stitched together as being the same peak. Our approach will follow this rational in three similar steps.

4.1 Feature Detection

The goal of this section will be to *detect features* within a single image. This means identifying 'interesting' points (or pixels) within the image that can serve as anchors. A commonly used method to detect features is 'Harris corner detection' [9], from which other methods follow. For simplicity, and without loss of generality, consider a square image I characterized as a matrix. Also consider a square window W within the image of size much smaller than I. When W moves from one location of I to another location nearby, the contents of the window change. If the contents change only a little, this implies this particular area of the image contains little contrast, and therefore it would be a poor choice for a feature point. On the contrary, if the contents of the window changes significantly, this implies this particular area contains a lot of change (e.g. an edge or corner of an object), and therefore it would be a wise choice for a feature point. Figure 1 shows this relationship. In order to find those neighborhoods within I where W changes significantly, we must first describe a function to measure the contrast. The function below, denoted C for 'contrast', will measure the contrast in the neighborhood of W when W is at an arbitrary location.

$$C(\Delta x, \Delta y) = \sum_{(x,y)\in W} (I_{x+\Delta x, y+\Delta y} - I_{x,y})^2$$

Note that while x, y and $\Delta x, \Delta y$ are theoretically integers due to the discrete form of the matrix, from here forward we will treat them as real numbers in order to use regular matrix theory and get an optimal result. With first order

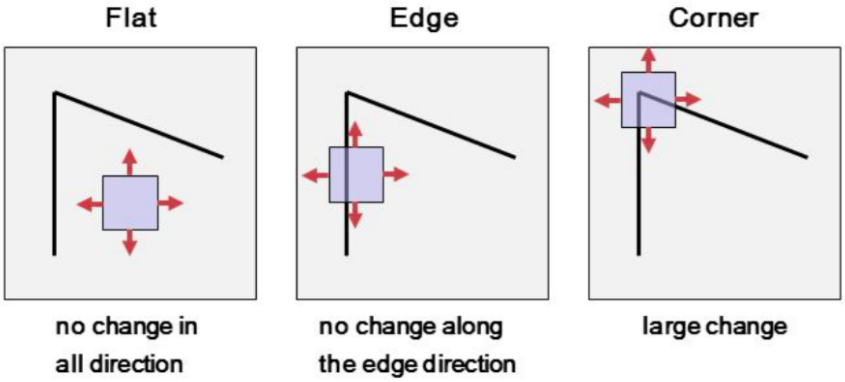

Figure 1: Contrast on window W, image from [9]

Taylor expansion,

$$I_{x+\Delta x,y+\Delta y} \approx I_{x,y} + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y$$

the function can be simplified to a more meaningful form

$$\begin{aligned}
&C(\Delta x, \Delta y) \\
&\qquad \approx \sum_{(x,y)\in W} \left(I_{x,y} + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y - I_{x,y}\right)^2 \\
&\qquad = \sum_{(x,y)\in W} \left(\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y\right)^2 \\
&\qquad = \sum_{(x,y)\in W} \left(\begin{pmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \right)^2 \\
&\qquad = \sum_{(x,y)\in W} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}^\top \begin{pmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{pmatrix}^\top \begin{pmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \\
&\qquad = \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} M \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}
\end{aligned}$$

Where M is the almost Hessian-looking matrix, known as a 'structure tensor'.

$$\begin{pmatrix} \sum_{(x,y)\in W} \frac{\partial I}{\partial x}^2 & \sum_{(x,y)\in W} \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_{(x,y)\in W} \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_{(x,y)\in W} \frac{\partial I}{\partial y}^2 \end{pmatrix}$$

The gradient matrices $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ can be computed discretely in a number of straightforward ways, and from those M may be computed. If $(\Delta x, \Delta y)^\top$ is a unit vector, then the value $C(\Delta x, \Delta y)$ is the contrast when moving the window unit distance in the vector's direction. We would like to know in what direction there will be the *maximum* contrast, because this implies a direction orthogonal to an edge within the image. We would also like to know in what direction there will be the *minimum* contrast, because this implies a direction parallel to an edge within the image. We find these directions by computing the maximum and minimum eigenvalues, λ_+ and λ_- respectively, of the matrix M, as well as their corresponding eigenvectors, v_+ and v_- respectively. The eigenvectors inform us of the *directions* of maximum and minimum contrast, while the eigenvalues inform us of the *magnitudes* of maximum and minimum contrast. This is shown by the equations below.

$$C(v_+) = v_+^\top M v_+ = v_+^\top(\lambda_+ v_+) = \lambda_+ v_+^\top v_+ = \lambda_+$$
$$C(v_-) = v_-^\top M v_- = v_-^\top(\lambda_+ v_-) = \lambda_- v_-^\top v_- = \lambda_-$$

The eigenvalues can be computed by the formula:

$$\lambda_\pm = \frac{1}{2}\left(M_{1,1} + M_{2,2} \pm \sqrt{4M_{1,2}M_{2,1} + (M_{1,1} - M_{2,2})^2}\right)$$

and then the eigenvectors can be solved for with the equation:

$$(M - \lambda_{\pm})v_{\pm} = 0$$

The best feature points in an image are corners of objects because they are the most identifiable, and unlike edges which correspond to lines, corners correspond to single points. According to the image of Figure 2, W contains corners when both the maximum and minimum eigenvalues are large. Therefore, in order to

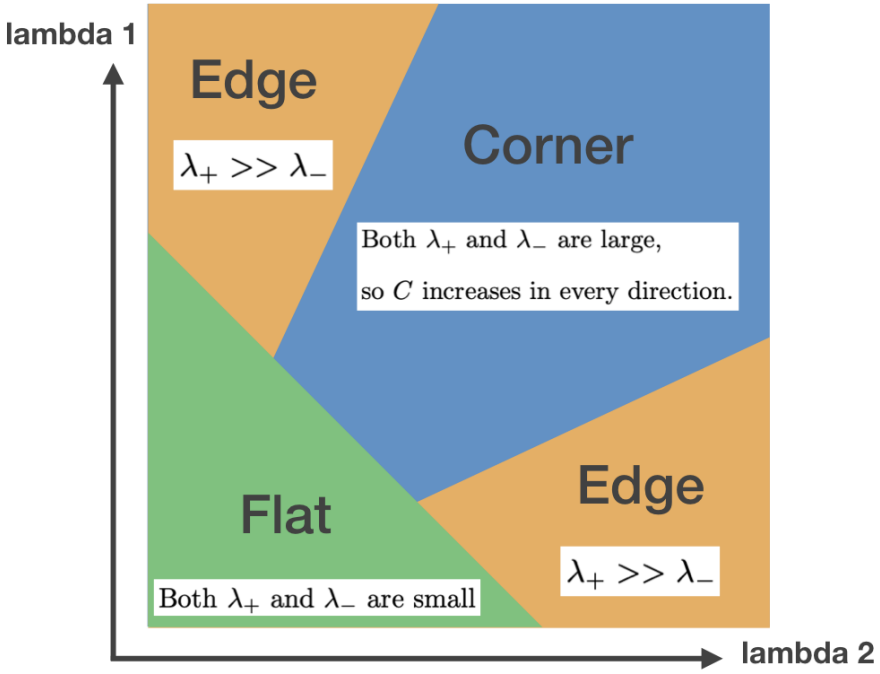

Figure 2: Eigenvalues of M

select the best features, C will be computed for all W. More specifically, we compute $C(x, y)$ with W centered at (x, y) for all $(x, y) \in I$. Secondly, we select point (x', y') if both eigenvalues for $C(x', y)$ exceed some chosen threshold. The lower the threshold, the more feature points, but the higher the threshold the better the feature points. It is worth mentioning that since computing eigenval-

ues can be expensive, an popular alternative is to compute the 'Harris operator' which is quick to compute and offers the same insight as λ_-. Depending on an empirical constant K, the operator is

$$\begin{aligned}\text{Harris}(M) &= \lambda_+\lambda_- - K(\lambda_+ + \lambda_-)^2 \\ &= (M_{1,1}M_{2,2} - M_{1,2}M_{2,1} - K(M_{1,1} + M_{2,2})^2 \\ &= \text{Det}(M) - K\text{Trace}(M)^2\end{aligned}$$

One trouble remains. That is, how can one be certain to select the *same* feature points from two similar images? What if the images are of different angle, brightness, or scale? If the images are presented at different angles, the eigenvalues will be the same, because eigenvalues are invariant under rotation, so the same feature points will be selected. If the images are of different brightness, the eigenvalues for the brighter image will generally be larger, but relatively the eigenvalues will maintain the same relationship, and thus the same feature points will be selected. If the two images are of different scale, however, what appears as a corner on the small image will appear as a curved line on the large image, which is difficult to solve with Harris corners. The general solution involves duplicating the images for a number of different sizes, and for each size again duplicating them for a number of different degrees of blurring. Blurring is achieved using the Gaussian filter with blurring parameter σ

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

The state of the art algorithm for this is SIFT (scale invariant feature transform) [10]. The algorithm is a re-formulation of the method above, and puts emphasis on scale invariance, while remaining robust against angle and brightness differences. In addition, the algorithm provides descriptions of the feature

points.

4.2 Feature Description

After detection features, they have to be described in order to allow for a matching in a later step. Difficulties arise when identical feature points are suspect to differences in angle, brightness or scale. Therefore, descriptors have to be *invariant* to such differences.

A simple approach is to use the pixels of the window surrounding the feature point as a descriptor of a keypoint. However, this is susceptible to brightness, rotation, and scale. Invariance under brightness can be accomplished by normalizing the magnitude of the pixels, and invariance under rotation can be accomplished by normalizing the orientation of the window of pixels. Normalizing orientation of the window W can be done by computing the maximum eigenvector v_+ of the corresponding matrix M and fixing that vector to always point in the same direction. Invariance under scale can roughly be accomplished by repeating the procedure for a number of different sizes and blurrings of the image.

The previous was a simplified method for accomplishing invariant descriptors. The algorithm SIFT is a particularly refined and extended version of this method that is used in production. For a feature point, SIFT takes the 16x16 window of pixels around it, and divides it into 16, 4x4 groups of pixels. For each group of 4x4 pixels, the orientation v_+ of each pixel is computed as an angle between 0 and 2π, and the angle for the 8 pixels with the maximum corresponding λ_+ are kept. For each of the 16 groups, the 8 angles are put together to form a descriptor represented as a 4*16 = 128 dimensional vector.

4.3 Matching Descriptors

Finally, identical features are to be matched by comparing the similarity of their descriptors. Therefore, a definition of 'similarity' is needed. A simple method is called SSD, which given two descriptors represented as vectors, computes the sum of squares of difference between the two vectors. This method is susceptible to false positives because a descriptor in one image may be similar to many more than just one descriptor in the other image. A solution is to compute the ratio of the similarity of the descriptor with its best match to the similarity of the descriptor with its second best match, effectively normalizing the measure of similarity. Other more sophisticated methods exist but they are far beyond the scope of matrices and thus beyond the scope of this paper.

5 Transformation

After obtaining feature matches from two images, one is able to further investigate the transformation of one image onto the other one. Therefore, different transformation types will be outlined and an approach to construct the transformation matrix will be given.

5.1 Transformation Types

Images can be transformed in different ways, which are illustrated in Figure 3. Transformation types can be seen as building upon each other, where a later one adds another type of adjustment to the image:

1. Translation - Translation
2. Euclidean - Rotation
3. Similarity - Scale

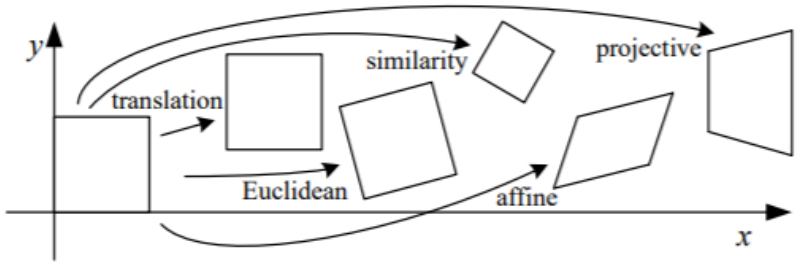

Figure 3: 2D Transformation of Images [11]

4. Affine - Affine (preserve parallelity)

5. Projective - Perspective (parallelity not ensured)

5.2 Transformation Estimation

Given a set of matching coordinates in two images, one is able to estimate the desired transformation matrix. Therefore, elements of the transformation matrix can be seen as unknowns, and they can be solved for with known pairs of matching coordinates. The following relation holds for two images in an affine transformation, where coordinates of the first image are represented as (u, v), and coordinates of the second image as (x, y).

$$\begin{bmatrix} u_1 & u_2 & \dots & u_n \\ v_1 & v_2 & \dots & v_n \\ 1 & 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_3 & \dots & y_n \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

As six unknowns exist, at least three keypoint matches are required to solve for the elements by least squares optimization:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ & & & \vdots & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_n \\ v_n \end{bmatrix}$$

[12]

5.3 Ransac

Even though thresholds are used to determine matching keypoints, there is a possibility for incorrect matches. In order to mitigate the impact of incorrect matches (outliers) and create a transform utilizing as many correct matches (inliers) as possible, RANSAC [13] will be used. RANSAC fits transformations with a random, small subset of matches. The first image will be transformed accordingly and the position of matching keypoints is being compared to find inliers. This procedure is repeated multiple times, until a sufficient number of inliers have been found.

6 Experimental Result

We implemented a simple image stitching approach in Python, in order to stitch an image of the UESTC Qingshuihe Campus [1]. This implementation relies to a

[1] http://en.uestc.edu.cn/index.php?m=content&c=index&a=lists&catid=169

large extent on external libraries including Numpy [2], Cyvlfeat [3] and OpenCV [4]. Numpy allows an easy processing of matrices, cyvlfeat provides us with a SIFT algorithm, and OpenCV is used for image processing.

We proceed by reading image files and extracting SIFT features and descriptors. Features will be matched in regard to a manually chosen threshold, such that we have a reasonable amount of features for the next step. For simplicity, similarity of features will be measured with a euclidean similarity, whereas other similarity measures exist that are more suitable for vector computations, such as Cosine similarity. Figure 4 illustrates 50 keypoint matches of a divided image of the Quingshuihe Campus. After matching features, RANSAC is performed, to find

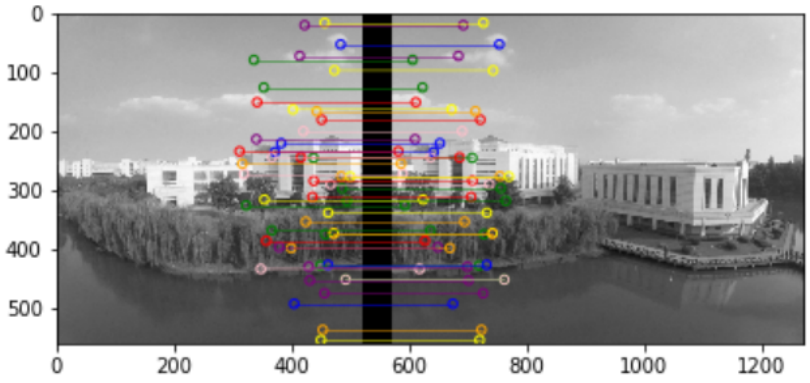

Figure 4: Feature Matching of a splitted Image of the Qingshuihe Campus

the best matches and fit an affine transformation. For our purposes, affine transformations are sufficient.

Lastly, we use the obtained transformation matrix to project the first image onto the second image to create a stitched image. The result can be seen in Figure 5.

[2] http://www.numpy.org/
[3] https://github.com/menpo/cyvlfeat
[4] https://opencv.org/

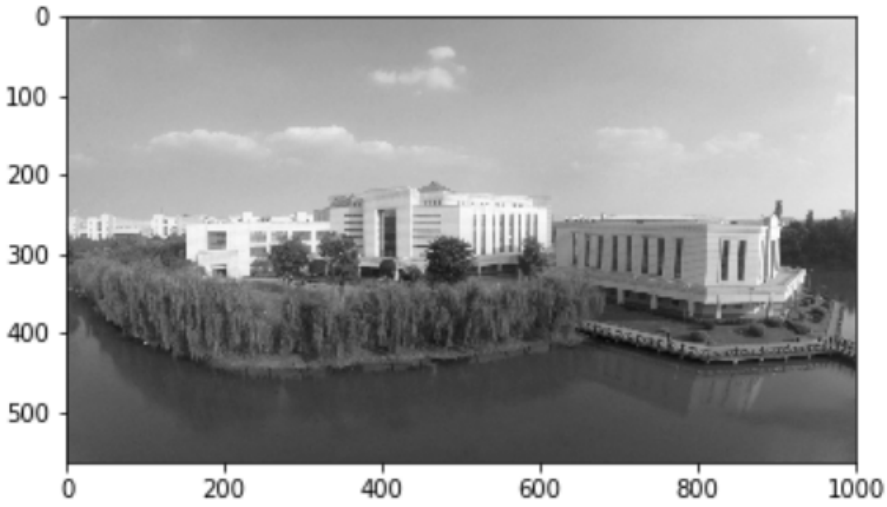

Figure 5: Stitched Image of the Qingshuihe Campus

7 Conclusion

Finalizing, we can see that image stitching offers an interesting practical application of matrices. Images are represented as matrices, features can be described as vectors and calculations are being done with matrix multiplications and solving for linear least squares in matrix form.

With our experiment, we were able to apply a selection of methods by ourselves, which gave us insights in the application of matrices not only in theory, but also in practical environments.

References

[1] H. Chau and R. Karol, "Robust panoramic image stitching."

[2] Q. Fu and H. Wang, "A fast image stitching algorithm based on surf," *2017 IEEE/CIC International Conference on Communications in China*

(ICCC), pp. 1–4, 2017.

[3] L. Yi-bo and L. Jun-jun, "Harris corner detection algorithm based on improved contourlet transform," *Procedia Engineering*, vol. 15, pp. 2239 – 2243, 2011, cEIS 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877705811019205

[4] P. Balasubramanian, V. K. Verma, and A. Mittal, "A performance evaluation of feature descriptors for image stitching in architectural images," in *Computer Vision - ACCV 2014 Workshops - Singapore, Singapore, November 1-2, 2014, Revised Selected Papers, Part II*, 2014, pp. 517–528. [Online]. Available: https://doi.org/10.1007/978-3-319-16631-5_38

[5] F. Yang, L. Wei, Z. Zhang, and H. Tang, "Image mosaic based on phase correlation and harris operator," 2012.

[6] X. Fan and S.-r. Xia, "Feature based automatic stitching of microscopic images," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques*, D.-S. Huang, L. Heutte, and M. Loog, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 791–800.

[7] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A comparison of affine region detectors," *Int. J. Comput. Vision*, vol. 65, no. 1-2, pp. 43–72, Nov. 2005. [Online]. Available: http://dx.doi.org/10.1007/s11263-005-3848-x

[8] A. Fathima, K. Ramamurthy, and V. Vaidehi, "Image stitching with combined moment invariants and sift features," *Procedia Computer Science*, vol. 19, p. 420–427, 12 2013.

[9] C. Harris and M. Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[11] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. Berlin, Heidelberg: Springer-Verlag, 2010.

[12] J. Brumberg, "Image mosaics," Feb 2006. [Online]. Available: http://cns.bu.edu/ brumberg/CS580/mosaic/MAIN/mosaic.html

[13] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.