P2Prectify and MoB video mosaic for UAV surveillance applications

by

Saengrawee Pratoomtong

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

(Electrical and Computer Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

2020

Date of final oral examination: 11/25/2019

The dissertation is approved by the following members of the Final Oral Committee:

Hu, Yu Hen Professor, Professor, Electrical and Computer Engineering

Sethares, Bill Professor, Professor, Electrical and Computer Engineering

Gubner, John A. Professor, Professor, Electrical and Computer Engineering

Lee, Kangwook Professor, Assistant Professor, Electrical and Computer Engineering

Li, Yin Professor, Assistant Professor, Biostatistics and Medical Informatics

To my parents.

Acknowledgments

It has been a long journey to complete my PhD degree. I would like to take this opportunity to acknowledge and extend my sincere gratitude to great number of people who have helped me along the way.

First and foremost, I would like to thank my advisor, Professor Yu Hen Hu, for his guidance. I appreciate his valuable feedback, advice, and encouragement that make my PhD journey possible. It has been an honor to be his PhD student. No words can express my sincere thanks and gratitude to him.

I would like to thank my committee members, Professor Bill Sethares, John Gubner, Kangwook Lee and Yin Li for giving me an opportunity to complete my PhD degree.

Special thanks to my sister and her husband for helping me prepare for my final defense exam and thesis. Thanks to Benny Vanhauer for supplying me with many source videos for my experiments. I would also like to thank Ms. Alexandra Walter for her constant support throughout the year of my graduate studies. I also thank my brother who inspired me to be resourceful and persevere.

No acknowledgment would be complete without mention my parents. I would never make this far without their love, support, and sacrifices. My mom has been through so much but she's always there for me. No one can ever replace her in my love. My father showed me how to be resilient and never give up on achieving your goal.

A very special thanks to my husband. I am fortunate to have your love and support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The use of video cameras onboard unmanned aerial vehicles (UAV) is a cost-effective alternative to traditional and aircraft platforms in surveillance and reconnaissance applications. Acquired video frames are often mosaicked into a panoramic image. Traditional image mosaic algorithms often yield distorted panoramic images and require excessive computational power, renders it unsuitable for time-critical applications. To address these challenges, we propose a Perspective-to-Parallel rectification (P2Prectify) and a Motion-Based video mosaic (MoB) algorithm. By recognizing that successive video frames acquired from UAVs are highly correlated and spatially dense, the execution time of a mosaic can be drastically reduced.

As parallel projection images can be mosaicked efficiently, P2Prectify detects and rectifies perspective projection images prior to creating a mosaic without any *a priori* knowledge of the scene geological data or the UAV's camera parameters. Therefore, the accumulated error, caused by the scaling and skewing components of the projective transformations used to align the images, are eliminated.

MoB video mosaic algorithm --- as its name may suggest --- uses motion information to facilitate the final panoramic image formation. MoB registration uses motion vectors of track points to estimate the camera motion. MoB composition consists of key frame and pixel selection mechanisms guided by the spatial distribution of the motion vectors.

Because the motion vectors and their statistics can be obtained trivially, both MoB registration and composition are computationally simple and therefore can be implemented efficiently for real-time applications. Compared to the traditional feature-based image registration and stitching algorithms, the MoB algorithm exhibits significant improvement in both performance and execution time.

CHAPTER 1: INTRODUCTION

1.1 Challenges and motivation

A rapid rise in the popularity and availability of unmanned aerial vehicles (UAVs) [1] has led to the use of video cameras onboard UAVs as a cost-effective alternative to traditional satellite and aircraft platforms [2]-[5]. It is often desirable to convert the acquired video frames to a panoramic image to provide a bird-eye view of the surveyed or monitored scene. In geological survey applications [3]-[5], prior knowledge of geological environment [6]-[12] as well as camera parameters are often known, and the image mosaic can be produced offline.

Contrary, in aerial surveillance applications [13]-[15], the UAV often flies in unfamiliar terrain without Geographic information system (GIS) landmarks. Therefore, several stitching algorithms that require GIS are inapplicable [16]-[23]. The UAV used in these applications also has a smaller payload [24] for portability, ease of deployment, and extended flight range. Consequently, it may not be equipped with guidance instruments such as the Inertial Measurement Unit (IMU). It may also be desirable to forego the use of a Global Positioning System (GPS) for security reasons [25]. Therefore, a real-time video mosaic algorithm capable of producing low-distortion panoramic images without relying on gyro or GPS sensors would provide tremendous benefits to these applications.

Accumulated error from misregistration is a common problem in image mosaic and in 3D reconstruction of a long image sequence. It is difficult and time-consuming to reduce the accumulated error. Techniques such as loop closing used in 3D reconstruction or global optimization used in image mosaic are based on a photogrammetry technique called simultaneous bundle block adjustment (BA) [26]. The accumulated error in image mosaic is caused by the scaling and skewing components in the projective transformations use to align the images. Two-dimensional BA [27] is used to reduce accumulated error in image mosaic by simultaneously minimizing the misregistration between all overlapping pairs of images and find a globally consistent set of transformations. The major drawbacks with this approach are the huge numbers of variables in the optimization and the non-linearity and non-convexity of the objective function that requires good initial estimates to ensure convergence. Due to its complexity, BA is usually performed offline after an initial panoramic is created and hence, not suitable for any time-critical video mosaic applications.

We develop a method to reduce accumulated error that are computationally simplified and can be implemented in real-time called Perspective to Parallel rectification (P2Prectify). P2Prectify algorithm is based on special motion characteristics of the video generated from a UAV camera that is pointing straight down. P2Prectify algorithm detects perspective images and rectifies them to parallel images prior to performing video mosaic. While other existing rectification algorithm requires knowledge of scene geological data [2],[28], or resulting in distort images [29]-[35]. P2Prectify algorithm produces low-distortion parallel images without any *a priori* knowledge of scene geological data or camera parameters.

We also develop a low-overhead Motion-Based video mosaic (MoB) algorithm that tracks the motion of sparse feature points to facilitate registration and composition. MoB registration uses motion vectors of track feature points to align video frames. Since motion vectors can be computed trivially from track points locations, MoB registration can be implemented efficiently. MoB composition uses spatial distributions of motion vectors to facilitate frame and pixel selection to reduce compositing time.

Lastly, we propose an incremental orthographic stitching (IO) algorithm which is capable of producing low-distortion panoramic images from weak perspective images.

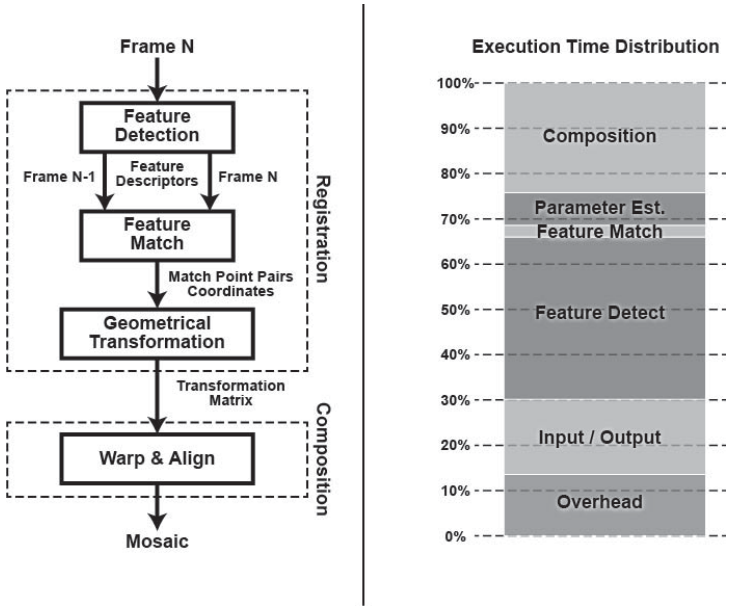


Figure 1.1: Overview of Feature-based (FB) image mosaic algorithm and execution time distribution.

1.2 Background: image mosaic

Image mosaic is a well-studied field in computer vision with a variety of commercial products [36]-[38]. Existing works involve video mosaicking in the past decade [39]-[45] generally use image mosaic algorithm consisting of four core steps as shown in figure 1.1: (1) feature detection, (2) feature matching, (3) geometric transformation model estimation, and (4) panorama composition. While this method can produce quality mosaics, it also requires a significant amount of time and computational power. For every successive frame, features must be detected and matched among hundreds (or thousands) of features from the previous frame. Then the parameters of the two-dimension planar motion model use to relate successive frames are estimated from those matching features. The 8 parameters projective transformation known as homography [18] is generally used because of its ability to relate images taken from different viewpoint. Since some matched features are not correct (i.e., miss match or false match), homography parameters are estimated with a robust iterative algorithm such as RANSAC [46] or MSAC [47] capable of handling a large number of outliers. The scaling and skew component of projective transformation will distort the mosaic over time. Therefore, to further refine the homography parameters, a bundle adjustment algorithm [19] may be used to minimize the miss-registration between all pairs of images and find a globally consistent set of parameters. Finally, all the pixels in the source image are mapped onto the final composite surface using the parameters obtained from the previous step (i.e., warp). To further enhance the panorama quality, pixel selection and weighting algorithm [21], optimum seam selection algorithm [22], or blending algorithm [23] may be used.

1.2.1 Feature detector

Many of the well-known feature detectors such as Harris [48], SIFT [49], SURF [50] are based on auto-correlation function. Features are detected by comparing the image patch against its neighboring block specify by direction vector (u, v) using their weighted summed square difference as shown in (1.1).

$$E(u, v) = \sum_{(x,y)} \mathbf{W}(x, y)(\mathbf{I}(x + u, y + v) - \mathbf{I}(x, y))^2 \quad (1.1)$$

A good feature should be a block that is highly distinctive so that it can be found in a different image of the same scene and provide a unique match. Therefore, good feature is a block that yields a high value of E when moving away from it in any u, v direction. We can use the Taylor series to

approximate the change in (u, v). The 2x2 matrix of a sum of gradient inside the window of interest in (1.2) is known as Harris Matrix [48]. The eigenvalues (λ_1,λ_2) and eigenvectors (e_1,e_2) of the Harris Matrix can be used to find good features.

$$\sum_{(x,y)} \mathbf{W}(x,y)\left(u\frac{\partial I}{\partial x}(x,y)+v\frac{\partial I}{\partial y}(x,y)\right)^2$$

$$=\sum_{(x,y)} \mathbf{W}(x,y)\left(u^2\left(\frac{\partial I}{\partial x}(x,y)\right)^2+2uv\left(\frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y)\right)+v^2\left(\frac{\partial I}{\partial y}(x,y)\right)^2\right)$$

$$=\begin{bmatrix}u\\v\end{bmatrix}^T\underbrace{\begin{bmatrix}\sum_{(x,y)}\mathbf{W}(x,y)\left(\frac{\partial I}{\partial x}(x,y)\right)^2 & \sum_{(x,y)}\mathbf{W}(x,y)\left(\frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y)\right)\\ \sum_{(x,y)}\mathbf{W}(x,y)\left(\frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y)\right) & \sum_{(x,y)}\mathbf{W}(x,y)\left(\frac{\partial I}{\partial y}(x,y)\right)^2\end{bmatrix}}_{\text{Harris Matrix}}\begin{bmatrix}u\\v\end{bmatrix} \quad (1.2)$$

Figure 1.2 [51] shows four types of blocks (flat, linear edge, corner, blob). The gradients in both directions of a block in a region with constant intensity (e.g., block A) are nearly zero. As a result, both λ_1 and λ_2 will be almost zero and the surface plot is nearly flat. The block contains a vertical linear edge (e.g., block B) has large horizontal gradient and almost zero vertical gradient. Therefore, λ_1 will be a large positive value with e_1 normal to the edge direction and λ_2 will be nearly zero with e_2 along the edge direction. This creates ambiguity along the edge direction and is known as aperture. The surface plot will resemble a stack of letter V in the direction of the edge. The block contains corners or blobs has large gradient in both directions. Therefore, both λ_1 and λ_2 are a large positive value and the surface plot will resemble a bowl.

Non-maxima suppression [52] is usually applied after all the Harris corners have been identified to ensure that only Harris corners with the highest value of C among its N × N pixels neighborhood specify by some user-selected N is retained. Multiple researchers have proposed a modification to the Harris matrix to make the feature detector more robust which add more computation complexity to the Harris matrix. [51] and [53] give good detail of those works.

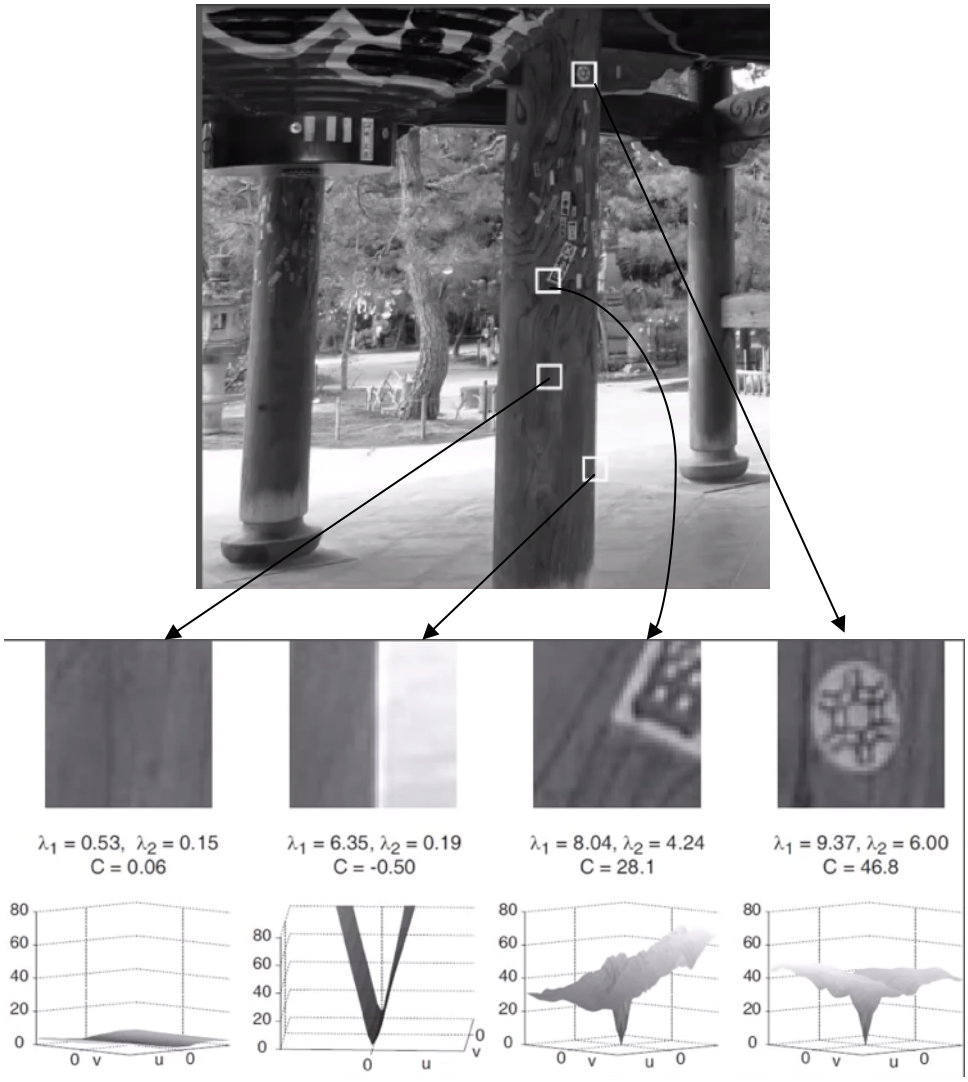


Figure 1.2: Four types of block (Flat, Linear edge, Corner, Blob) and their corresponding surface plot, Harris matrix eigenvalue, and Harris cost [51].

1.2.2 Feature descriptor

Feature descriptors represent the feature mathematically so that different features can be matched. The most basic descriptors are in a form of the vector contains the intensity values from fix size block of pixels centered at the feature location. To compare two features in this format, a sum of square differences or a normalized cross-correlation can be used.

SIFT descriptor [49] is one of the most popular and widely used descriptors in the computer vision community. It is highly distinctive and invariant to some changes in illumination and perspective.

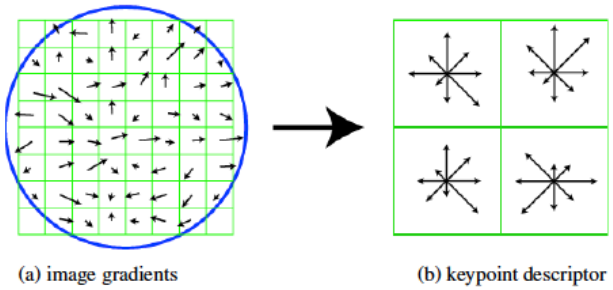


Figure 1.3: (a) 8 × 8 pixels gradient orientation and magnitude of image patch center at feature point. Weighted by Gaussian fall off function in blue circle. (b) 2 × 2 descriptor array with 8 bin weighted gradient orientation histograms [49].

SIFT feature descriptor contains scale, location and orientation information of each feature encoded in a 128-element feature vector. Figure 1.3a shown the gradient magnitude and the orientation of an 8×8 pixels patch around feature points weighted by Gaussian weight shown in the blue circle. [49] suggests using a 16×16 pixels patch.

Multiple descriptors adapted from the idea of SIFT have been purposed, e.g., [50], [54], [55]. [55] also review multiple robust image descriptors and compare their performance.

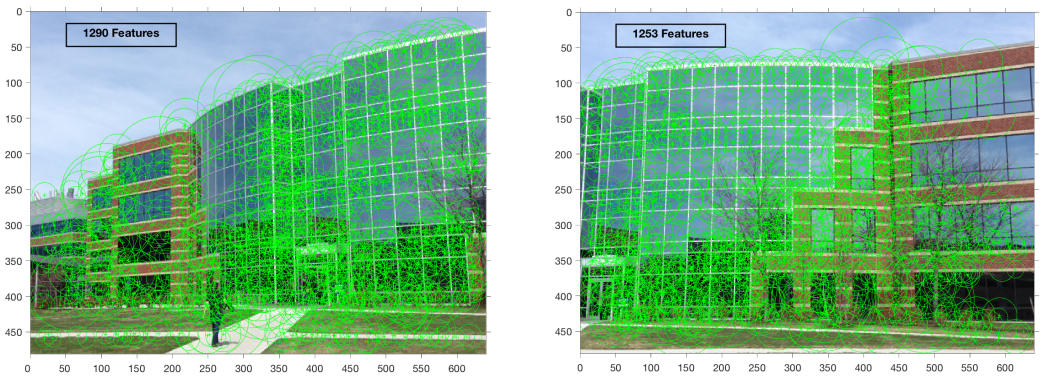


Figure 1.4: Example of registration: feature detection. Source image: MATLAB.

1.2.3 Feature matching

After features and their descriptors from two or more images have been extracted, the next stage is to find feature matches between these images. Since a feature descriptor is in a form of a vector of a real number, the simplest descriptor distance measure is the Euclidean distance. Once the Euclidean distances are calculated, the simplest matching strategy is to return all the matches

within a set fixed threshold. However, a fixed threshold is difficult to set since different parts of the image can have different responses to a single fix threshold. A better strategy is to match the nearest neighbor in feature space and then use a threshold to reduce the number of false-positive, i.e., incorrect matches being returned. [49] and [56] purpose heuristic comparing the nearest neighbor distance with the second nearest neighbor and only return matches that have nearest distance ratio less than a certain threshold (they use the threshold equals 0.8 bases on their experiment). This strategy is very useful in eliminating ambiguous match, which has a high potential to be a false match, caused by aperture problem or multiple duplicate structures in the image such as an image of building exterior with multiple windows. Once the matching strategy is set, a simple exhaustive search can be used to find potential candidates. However, an exhaustive search can take very long and the time increase quadratically with the number of features. Many researchers have proposed an indexing structure to increase the speed of feature matching search. [57] propose Haar wavelet for hashing MOPS descriptor. [58] purpose KD-trees, a multi-dimensional search tree with best bin first search. [59] compare many types of indexing structure and search.

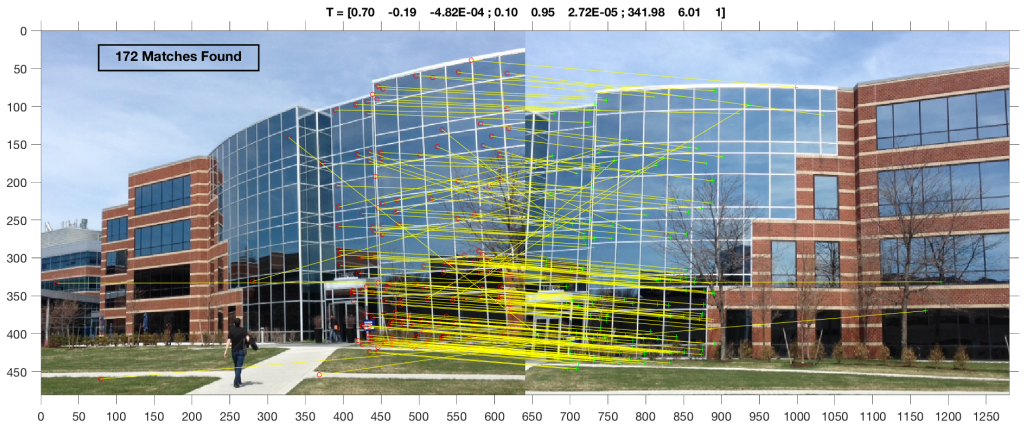


Figure 1.5: Example of registration: feature match and projective transformation estimation. Source image: MATLAB.

1.2.4 Feature-based alignment

Once features and their match have been established, they are used to estimate the parameters of the two-dimension planar motion model which is used to transform the images (i.e., warp) so they can be composited into a single integrated image. [18] presents a very in-depth detail of how to estimate the parameters from matched features. Table 1.1 shows the property of various 2D planar

transformation ranging from the lowest degree of freedom (DoF) of 2 for translation transformation to highest DoF of 8 for projection transformation. Translation, Euclidean, and similarity transformation consist of one or more of these actions: translation (**t**), rotation(**R**), and scaling (**s**). Affine transformation consists of rotations, non-isotropic scaling, and translation. Projective transformation is a combination of similarity, affine, and elation [18], which move the line at infinity resulting in the ideal point (i.e., the point at infinity) to map to a finite point. Projective transformation matrix (homography) is a 3x3 matrix but only have 8 degree of freedom due to the scale ambiguity, which allows one element to be set to one (the last element of the homography is normally set to one). The majority of image mosaic algorithm uses projective transformation because it provides a realistic model of how image projection on to the camera.

Table 1.1: Summary of properties of 2D planar transformation [53].

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\, \boldsymbol{I} \mid \boldsymbol{t} \,\right]_{2\times3}$	2	orientation	
rigid (Euclidean)	$\left[\, \boldsymbol{R} \mid \boldsymbol{t} \,\right]_{2\times3}$	3	lengths	
similarity	$\left[\, s\boldsymbol{R} \mid \boldsymbol{t} \,\right]_{2\times3}$	4	angles	
affine	$\left[\, \boldsymbol{A} \,\right]_{2\times3}$	6	parallelism	
projective	$\left[\, \tilde{\boldsymbol{H}} \,\right]_{3\times3}$	8	straight lines	

Figure 1.6: Example of composition: warp and stitch. Source image: MATLAB.

1.3 Related work

There have been some notable feature-based UAV video mosaic projects during the past decade. Work done in [42] is influenced by machine learning using the bag of word (BoW) [60] feature to register an aerial video frame. However, it is limited to aerial scenes with a lot of roads and buildings. The BoW features are collected and accumulated as the flight goes on. Since consecutive video frames will likely have large areas of overlap, searching a BoW database to find the matches is much faster than the feature matching approach in which all feature descriptors in each frame need to be searched to find possible matches between them. It is reported in [42] that typical computation time per mosaic frame is 162 ms. Researcher in [43] constructs a video mosaic from UAV input video frames using feature-based video mosaic with SURF feature points. A direct linear transformation (DLT) algorithm [18] is used to determine homography and RANSAC is used only to eliminate the outliers from the images feature matching dataset. Pairwise bundle adjustment is used to align the new frame into the existing mosaic. A global bundle adjustment is performed periodically. They report that on average, it takes about 75 ms to produce one mosaic frame; equivalent to about 13 frames per second. The visualization takes an average of about 45 ms, and possibly as much as 50 ms, resulting in the worst case of 20 FPS. In [61] a mosaic algorithm for UAV sequence images is implemented using the Harris corner detector and the pyramid Lucus-Kanade optical flow algorithm [62] to find match feature points. They report that the average processing time per frame is 58 ms. [15] propose real-time system that displays the video stream from mini-UAV intend for search and surveillance application. Their system can display temporal local mosaic with stabilization of terrain subject to the search operation. They use feature-based mosaic algorithm with the Harris corner detector and RANSAC to estimate the Euclidean transformation. They provide viewers with three display modes, mosaic, stabilized view, stabilized view with mosaic and subjectively rate the performance between three display modes. However, the time it takes to perform the video mosaic using this method was not reported.

[63] proposes a framework of estimating the position of the UAV using the image sequences obtained from the UAV. They use a standard feature-based method to perform local alignment between two consecutive frames. The global alignment error reduction is activated when the UAV already visits zones that already exist in the mosaic with overlap greater than 50% of the image size and the correct matches between the overlap part of the image and the mosaic are greater than

a defined threshold. They use the homography resulting from error reduction to compute the motion between two camera positions.

[45] propose an automatic mosaic system for UAV remote sensing video image. They use feature-based method to create the mosaic. The input image is rectified using a geometric correction model calculated from flight attitude parameters recorded during UAV flight. Then using the latitude and longitude information of the image center point, the two images are converted into the same coordinate system to determine the overlap region. They state that they can improve the speed of feature matching by only detect SIFT features where the image is overlapped. They also modify the SIFT detector to enhance the accuracy and improve the noise immunity of the extreme points. Their SIFT feature only has 32 dimensions. They report that the guide feature detection allows them to reduce the number of feature points. They only report the time it takes them to detect SIFT feature in two frames which is 532 ms and 250 ms. [64] propose a feature-based UAV video mosaic algorithm using Moravec corner matching method. They use a three-level image pyramid and perform hierarchical matching. They generate a virtual grid of 40x40 pixels across the first image and perform feature detection using the Moravec corner detector. Then they project the location of adjacent points of every feature in the first frame on to the adjacent frame and perform block based cross correlation matching. They use the match feature points as a tie point and create a mosaic which is then blended together using fade in/fade out technique. They did not provide the time it takes to perform the image mosaic.

The research in [41] involves video that is approximately nadir view with large translation motion when UAV is moving at high speed. They rely on data collecting from UAV flying repetitively in cycles over the same scene. They use a traditional image mosaic algorithm along with their proposed technique to minimize the accumulated error by propagating the registration error correction along each cycle.

[65] propose a method for UAV video summarizations by sectioning aerial video using energy-based graph cut to identify temporal shot boundaries within the video and use feature-based image registration technique to create a mini mosaic from each of the aerial video section. The work from [66] is based on [65]. In [66], the mini mosaic is created periodically (i.e., every 40 frames or less) instead of using the graph cut technique to sectioning the video. To reduced accumulated error(drift) occurs from a traditional frame to frame registration technique, they perform a two

steps image registration. First, they perform image registration on a group of consecutive frames and created a mini mosaic. Then they find global transformation by register the reference frame of each mini mosaic together. They report that it takes a few seconds to several minutes, depending on the complexity of the feature detector, to mosaic each frame.

[40] aims to reduce the bandwidth utilization between drone and ground server by selectively compress and send only subset of frames from the drone to the ground server. Using the drone's location sensor and known camera parameters to derive the transformation from the current image frame to the next image frame and compute the overlap between the two image frames from it. They assume that the video is nadir-view. The predictive model based on the area of overlap, the mosaic component, and the network bandwidth overhead is used to determine which frame to be transmitted and used to compose the mosaic. They create a mosaic from the transmitted frames offline at the ground station using a traditional feature-based image mosaic algorithm with SIFT features.

There are many existing image rectification algorithms for stereo vision based on epipolar geometry [29]-[33]. [34] propose an alternate approach that estimates the rectifying homography directly from point correspondence without computing the fundamental matrix. [35] proposed a rectification method suitable for view morphing application. All of these image rectification algorithms find homography pair that rectify an image pair so that their epipolar lines coincide with matching scanlines while minimizing some measure of distortion. For the rectify images to have horizontally aligned epipolar lines, the rectifying homography consists of projective transformation that moves the epipoles to infinity along the x-axis so that the epipolar lines become parallel and horizontal shear and translation so that the epipolar lines are horizontally aligned.

1.4 Summary

In this work, we investigate an alternative solution to the single-camera aerial video stitching problem and propose a robust algorithm that may address many issues incurred in making panoramic images. Although the majority of existing works assume that the video to be mosaicked is generated from a camera that is strictly pointing downward, they are using a traditional image mosaic algorithm to create the mosaic. When the camera axis points directly downward and UAV only engages in 2D planar motion, e.g., translation and yaw, the aerial video has unique

characteristics. In this case, since the UAV motion is parallel to the image plane, the baseline connecting the camera projection center between two different time instances is also parallel to the image plane, hence the epipoles are at infinity resulting in parallel projection lines (i.e., converge at infinity). Parallel projection preserves both distances and angles so mosaicking images with parallel projection only requires translation. Furthermore, there is no distortion of shape or distance when applying translation to parallel projection images and there is no accumulate error when creating a mosaic since there is no scale and skew components in the transformation. Therefore, using traditional image mosaic algorithms for this type of aerial videos is inefficient due to its high computation overhead and complexity. Perspective image contains both motion parallax and projective distortion. These characteristics make it more challenging to create a mosaic from perspective view images. Existing works that involve perspective videos also rely on further enhancement of the mosaic by other traditional means such as blending, bundle adjustment which increase the processing time and complexity even more.

Base on the unique characteristic of aerial video discuss earlier, we propose a low-overhead video mosaic algorithm called Motion-Based (MoB) video mosaic. In the MoB video mosaic algorithm, feature detection is only performed when the tracker is initialized or when the number of track features falls below a threshold. Tracking features eliminate the need to perform feature matching and homography estimation since translation parameter can be computed directly from feature tracking. Furthermore, the process of tracking feature is highly accurate [62] and hence yield a highly accurate and stable translation parameter. Therefore, it is not necessary to employ the bundle adjustment algorithm or panorama quality enhancement algorithm when using our algorithm to create a video mosaic. MoB video mosaic also provides a framework to reduce the compositing time by compositing only select key frames and reduce composting bandwidth by compositing only select pixels.

Since parallel projection image can be mosaicked efficiently as discussed earlier, we choose to rectify the projective video before mosaicking rather than applying traditional quality enhancement algorithms after creating a mosaic from a projective video. Since existing image rectification algorithms are not applicable or suitable for our MoB framework as discussed earlier, we devise a method to detect perspective projection images and rectify them to parallel projection images without any knowledge of scene geological data or camera parameters. Unlike stereo vision

application, video mosaic application does not require the input image to have horizontally aligned epipolar lines. Therefore, we propose a different approach to rectify the image called Projective to Parallel rectification (P2Prectify) which aims at producing rectified images that have common parallel projection views that are similar in shape and size and have minimum distortion.

1.5 Outline

The rest of this thesis is organized as follows. In chapter 2, we discuss the proposed Projective to Parallel rectification algorithm (P2Prectify). The proposed Motion-Based (MoB) video mosaic algorithm is presented in chapter 3. Experimental comparison with existing video mosaic algorithms using raw video streams generated from a UAV is reported in chapter 4. Conclusions and future directions are presented in chapter 5.

CHAPTER 2: PERSPECTIVE TO PARALLEL RECTIFICATION

2.1 Characteristic of aerial surveillance

Aerial video image can be categorized into parallel and perspective depending on the camera axis angle with the ground. Parallel projection (Orthographic image) occurs when the camera point straight down and the UAV only traverse in 2D plane, i.e., no pitch and no roll, and the camera axis is perpendicular to the ground, resulting in uniform scale and motion throughout the video sequence. When a camera pointing at an angle to the ground or the UAV engaging in a roll motion, the camera axis makes an angle with the ground, resulting in perspective image. The projection lines of the perspective image converge at the center of projection close to the image plane. Figure 2.1[28] shows a relation between perspective and orthographic image.

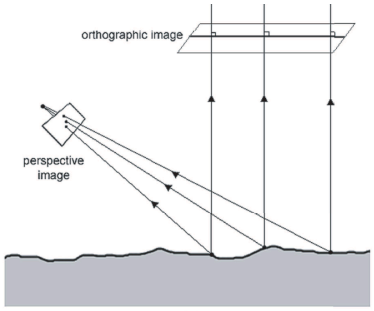


Figure 2.1: The relationship between the perspective projection lines and the parallel or orthographic projection lines [28].

The pinhole camera equation describes relationship between an object's world coordinate $[X_s \ Y_s \ Z_s]^T$ and its coordinates in pixels in the image plane of a pin-hole camera:

$$\mathbf{q} = \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \mathbf{K}[\mathbf{R} \ \mathbf{t}] \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{bmatrix} = \mathbf{Mp} \tag{2.1}$$

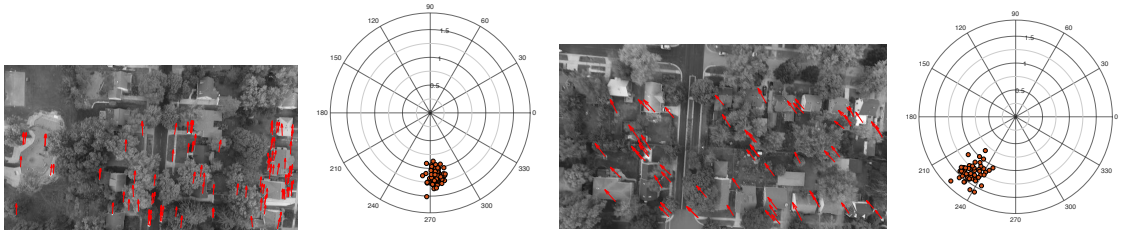
where $\mathbf{q}$ and $\mathbf{p}$ are the homogeneous coordinates and the image pixel coordinates are $\mathbf{i}' = [\frac{u'}{w'}, \frac{v'}{w'}]^T$, w' is the camera's depth of the focal plane. $\mathbf{K}$ is determined by the camera's intrinsic parameters, $\mathbf{R}$ and $\mathbf{t}$ are rotation and translation of the camera's optical center and principal optical axis.

We assume the camera is mounted at the bottom of a drone facing the ground. The drone usually flies straight forward, and the camera will move through simple translation $\Delta\mathbf{t}$. But upon turning,

it may rotate along both the vertical axis (yaw) and the flight path axis (roll) and it will rotate through $\Delta\mathbf{R}$. Thus, for the same world coordinate $\mathbf{p}$, its homogeneous image coordinates will differ by

$$\mathbf{q^{new}} = \mathbf{K}[\mathbf{R} + \Delta\mathbf{R} \quad \mathbf{t} + \Delta\mathbf{t}]\mathbf{p} = \mathbf{q} + \mathbf{K}\left(\Delta\mathbf{R} \cdot [X_s \quad Y_s \quad Z_s]^{\mathbf{T}} + \Delta\mathbf{t}\right) \tag{2.2}$$

If $\Delta\mathbf{R} = \mathbf{0}$ (no roll or pitch), $\mathbf{q}^{new}$ - $\mathbf{q} = \mathbf{K}\cdot\Delta\mathbf{t}$. Furthermore, if the camera is moving horizontally (the drone is not descending or ascending), that is, $\mathbf{\Delta t =} [\delta_x, \delta_y, 0]^T$, then $\mathbf{q}^{new}$ and $\mathbf{q}$ will have the same depth of focal plane. As such, $\mathbf{K}\cdot\Delta\mathbf{t}$ will be proportional to the corresponding image coordinates difference $\Delta\mathbf{i} = \mathbf{i}^{new}$ - $\mathbf{i}'$. $\Delta\mathbf{i}$ is called a *motion vector*. Note that the motion vector is independent of the real-world coordinate $[X_s \ Y_s \ Z_s]^T$ and is only dependent on the drone camera's velocity $\Delta\mathbf{t}$ (displacement per frame). Thus, if there are multiple feature points on the ground, the corresponding motion vectors will be identical. If one plots these motion vectors on a 2D plot, they should be clustered in one specific point.



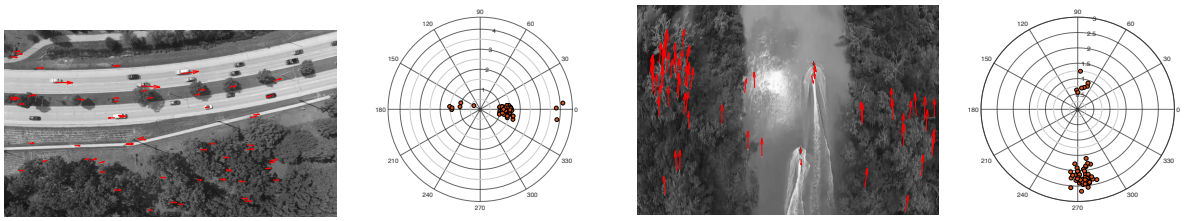
(a) (b)

Figure 2.2: Motion vectors of track points in successive frames of orthographic video and their polar plots.

Figures 2.2 to 2.5 show the motion vectors of track points in successive frames of orthographic and perspective videos. The motion of points in the orthographic video is uniform in both magnitude and phase except for the motion of moving objects in the video. Figure 2.2 shows the motion of points in orthographic video with the UAV flying straight and yawing. In both cases, the magnitude and phase of the motion vectors are uniform with the phase indicating the direction of the UAV heading. Figure 2.3 shows the motion of the points in the orthographic video contains moving cars and boats. In both cases, regardless of the relative speed of the moving object and the UAV, the magnitude of the motion of the non-stationary points, i.e., points on the moving object are highly distinct from those of stationary points. The motion magnitude of non-stationary points is larger than the motion magnitude of stationary points when the object is moving at a faster speed

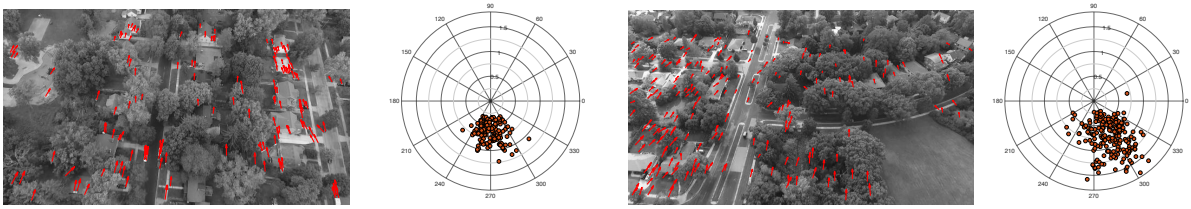
than the UAV (figure 2.3a) and smaller when the object and the UAV are moving at a similar speed (figure 2.3b).

The motions of points in perspective video fluctuate and the fluctuation increases as the angle between the camera axis and the ground decreases as shown in figure 2.4. The magnitude fluctuation indicates the presence of motion parallax and the phase fluctuation indicates the presence of projective distortion (e.g., parallel road on a scene plane are not parallel but instead converge to a finite point). However, motion parallax is less present, and the projective distortion is not present in perspective video with camera's FOV perpendicular to the UAV principle direction (side perspective) as shown in figure 2.5.



(a) (b)

Figure 2.3: Motion vectors of track points in successive frames of orthographic video with moving objects and their polar plot (a) moving cars (b) moving boats.



(a) (b)

Figure 2.4: Motion vectors of track points in successive frames of perspective video with camera FOV align with UAV direction and their polar plots.

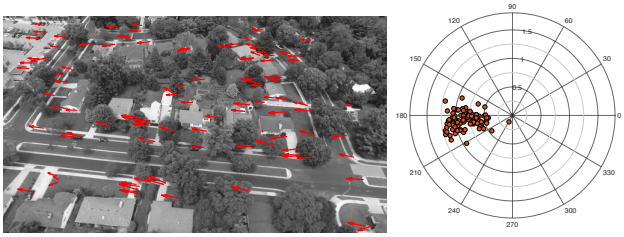


Figure 2.5: Motion vectors of track points in successive frames of perspective video with camera FOV perpendicular with UAV direction and their polar plot.

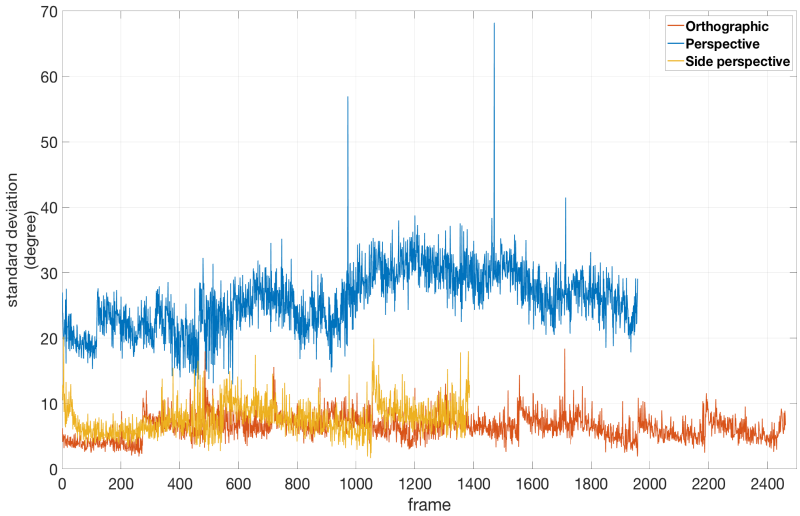


Figure 2.6: Standard Deviation of motion vector phase of orthographic (figure 2.2), perspective (figure 2.4b), and side perspective (figure 2.5) video shown in orange, blue, and yellow respectively.

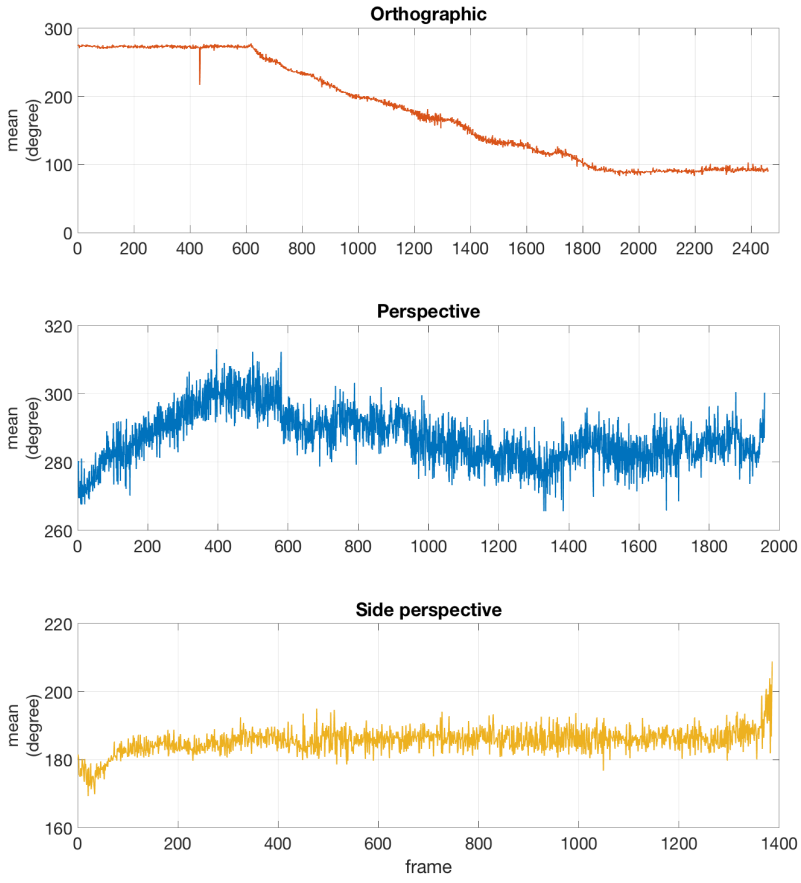


Figure 2.7: Mean of motion vector phase of orthographic (figure 2.2), perspective video (figure 2.4b), and side perspective video (figure 2.5) video shown in orange, blue, and yellow, respectively.

The motion vector magnitude is small and has a low dynamic range except for scenes that contain fast-moving objects. Therefore, the magnitude of the motion vectors can be used to identify anomalous points or points on fast-moving objects. From (2.2), the ideal scenario is when the motion of the points only depends on the UAV velocity. In this scenario, the motion vector of the points is uniform and hence represents global motion that can be used to align the video frame. Therefore, we devise a method to maintain the uniformity of the motion of the points. Figures 2.6 and 2.7 show the standard deviation and the mean of the phase of the motion vectors, respectively. In perspective video, the motion of the points depends on both UAV velocity and coordinate of the corresponding 3D world points as shown in (2.2). As a result, the phase of the motion vectors highly fluctuates as shown in figures 2.6 and 2.7. Therefore, we propose the P2Prectify algorithm which transforms perspective images to parallel images and essentially minimizes the term $\Delta \mathbf{R} \cdot [X_s \; Y_s \; Z_s]^T$ in (2.2) so that the motion of the points is only due to the translation of the camera.

2.2 Background: epipolar geometry and image rectification

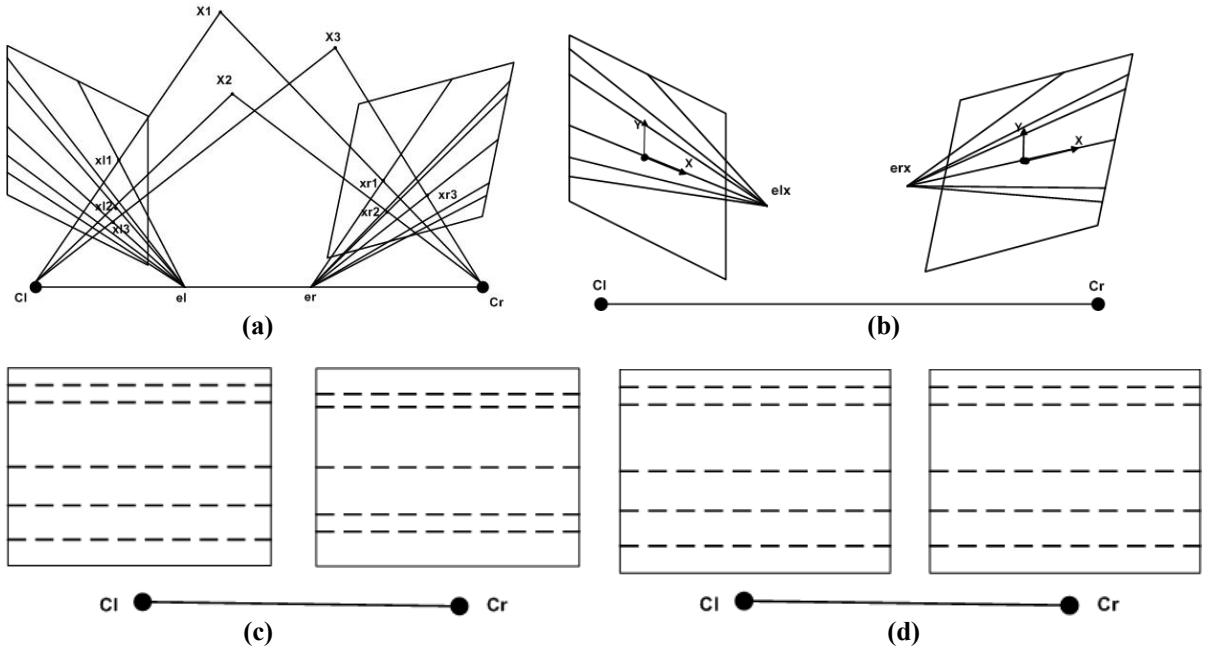


Figure 2.8: (a) Original unrectified images. P_i are point in 3D world, xl_i and xr_i are projection of point P_i on left and right image, el and er are the epipoles of the left and right image, Cl and Cr are the camera center of the left and right image. (b) Transformed images in a so that epipoles resides along the X-axis. (c) Transformed images in b so that epipoles move to infinity along the X-axis. (d) Transformed images in c so that epipolar lines are horizontally aligned.

Epipolar geometry relates image pair taken at the same time instant by two cameras in a different position. Epipolar geometry defines the constraint for correspondence points between image pair, i.e., the correspondence point of the projection of a 3D point on the first image plane must reside somewhere along a line (called an epipolar line) in the second image plane. There are epipolar lines in the second image containing correspondence points for each 3D point in the first image and vice versa. Epipolar lines in each image all intersect at epipoles which is the projection of the camera centers on the image plane of the other camera. Figure 2.8a shows the epipolar geometry of three 3D points (X1,X2,X3). Three epipolar planes, each plane rotates about the baseline connecting the two camera center $\mathbf{Cl}$ and $\mathbf{Cr}$, are generated from 3 set of point, i.e., (Cl,X1,Cr), (Cl,X2,Cr),(Cl,X3,Cr). Every 3D points the scene lies on one of these epipolar planes. Each of the epipolar planes intersects the image planes create a pair of conjugate epipolar line in the two images. All the epipolar lines in one image intersect at the epipoles which is the projection of the camera center of the other view, i.e., $\mathbf{el}$ is the projection of the camera center of the right camera (Cr) on to the image plane of the left camera and vice versa. The epipoles that is located outside the image plane means that the projection lies on the extended image plane. A family of planes that rotates about the baseline is known as epipolar pencil.

The correspondence epipolar line in one image to a point in the other image can be computed from the Fundamental matrix ($\mathbf{F}$) because the relative orientation and intrinsic parameters of the camera between two views are encoded in the fundamental matrix. The fundamental matrix is derived from coplanarity constraint. That is, if three points are on the same plane, the scalar triple product of the three vectors connecting these 3 points, which geometrically is the volume of the parallelepiped defined by these 3 vectors, must be zero. Let's consider the plane define by 3 points in Cl,X1,Cr in figure 2.8a and let $\mathbf{cpl}, \mathbf{b}, \mathbf{cpr}$ be vectors connecting point pair (Cl,X1),(Cl,Cr),(Cr,X1) respectively. Coplanarity constraints of those three space vectors can be expressed as

$$\mathbf{cpl} \cdot (\mathbf{b} \times \mathbf{cpr}) = \mathbf{0} \tag{2.3}$$

$\mathbf{cpl}$ and $\mathbf{cpr}$ represent ray from 3D point to the image plane observe by each camera. Therefore, the directions of these vectors can be derived from the image coordinates $\mathbf{xl_i}$, $\mathbf{xr_i}$. The projection of the 3D points onto the image plane can be formulated as

$$\mathbf{xl_i} = \mathbf{P_l X_i} = \mathbf{K_l R_l}[\mathbf{I_3} \mid -\mathbf{X_{cl}}]\mathbf{X_i}$$

$$\mathbf{xr_i = P_r X_i = K_r R_r [I_3 \mid -X_{cr}] X_i} \tag{2.4}$$

where $\mathbf{K}$ is the camera calibration matrix and $\mathbf{R}$ is a 3x3 rotation matrix representing the orientation of the camera coordinate frame and $\mathbf{X_{cl}}$, $\mathbf{X_{cr}}$ represents the coordinate of the camera centers in the world coordinate frame.

The normalized directions of the vectors **cpl** and **cpr** are $\widehat{\mathbf{xl}} = \mathbf{R_l^{-1} K_l^{-1} xl}$ and $\widehat{\mathbf{xr}} = \mathbf{R_r^{-1} K_r^{-1} xr}$. The base vector, $\mathbf{b}$, can be computed directly from the coordinate of the camera center, i.e., $\mathbf{b} = \mathbf{X_{cl} - X_{cr}}$. Use these variables to formulate the coplanarity constraint as follows

$$\widehat{\mathbf{xl}} \cdot (\mathbf{b} \times \widehat{\mathbf{xr}}) = \mathbf{0} \rightarrow \widehat{\mathbf{xl}}^{\mathbf{T}} [\mathbf{b}]_{\times} \widehat{\mathbf{xr}} = \mathbf{0} \rightarrow \mathbf{xl^T K_l^{-T} R_l^{-T} [b]_{\times} R_r^{-1} K_r^{-1} xr = 0} \tag{2.5}$$

With $[\mathbf{b}]_{\times}$ being a skew-symmetric matrix $\begin{bmatrix} 0 & -b_3 & b_2 \\ b_3 & 0 & -b_1 \\ -b_2 & b_1 & 0 \end{bmatrix}$

As a result, fundamental matrix, $\mathbf{F}$, can be expressed as

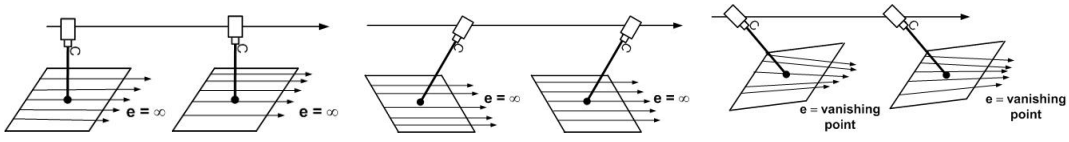
$$\mathbf{F = xl^T K_l^{-T} R_l^{-T} [b]_{\times} R_r^{-1} K_r^{-1}} \tag{2.6}$$

The information about the relative orientation and camera intrinsic parameters are encapsulated in the fundamental matrix. Therefore, point correspondences between two images from uncalibrated cameras is related via Fundamental matrix as follows

$$\mathbf{xl^T F xr = 0.} \tag{2.7}$$

Figure 2.8 illustrates the rectify algorithm wildly use in stereo vision application. The goal is the find a pair of homography that transform left and right image so that they have horizontally aligned epipolar line. Both homography consist of a chain of transformation that move epipoles onto the x-axis, then move epipoles to infinity along the x-axis so that the epipolar lines are parallel, and then transform the images so that their epipolar line are aligned.

2.3 Perspective to Parallel rectification (P2Prectify)



(a) (b) (c)

Figure 2.9: Epipolar geometry of aerial image. (a) Camera point straight down (b) Angled camera with FOV perpendicular to the UAV direction (c) Angled camera with FOV align with the UAV direction.

Figure 2.9 illustrates the epipolar geometry of aerial images. The key difference between the epipolar geometry of the aerial image and the epipolar geometry of the general image is that the epipoles of the aerial image lies at infinity along the direction of UAV heading for orthographic

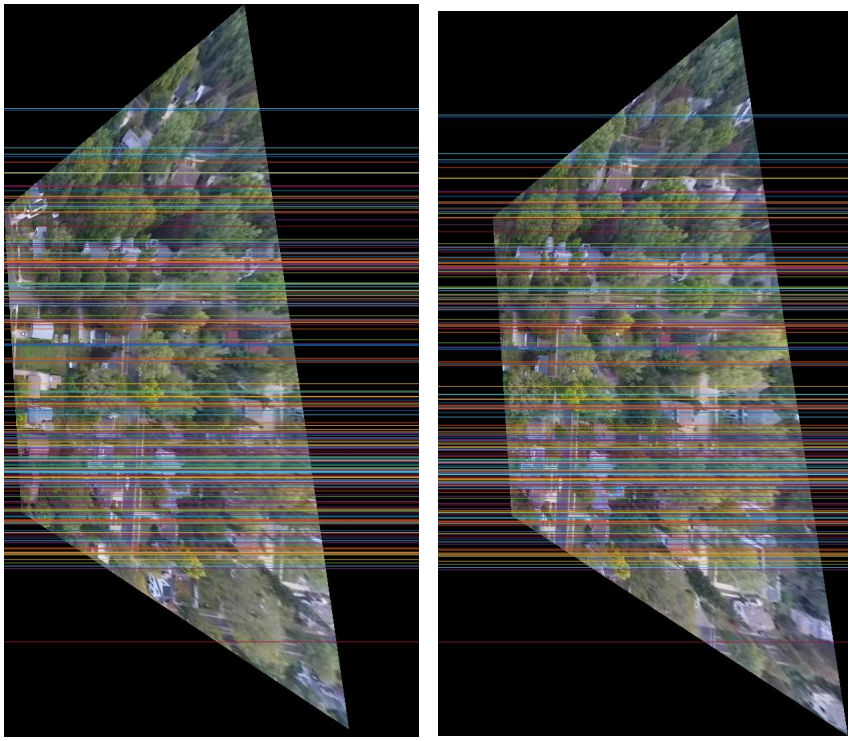


Figure 2.10: Rectified images of image in figure 2.11a using traditional image rectification algorithm discuss in section 2.2.

image and at the horizon (vanishing point) along the direction of the UAV heading for perspective image. That is, epipoles of all of the aerial images in the same flight mission are all at the same points (i.e., at infinity for orthographic image and a vanishing point for perspective image) and thus the same homography can be used to transform perspective image to orthographic image throughout the entire flight mission. In order to produce a good quality mosaic with minimum time, the input images need to have a common parallel projection view. Therefore, using one homography to convert all the perspective image ensure that they all have a common parallel projection view and similar in size.

Figure 2.10 shows the result of using general stereo rectification algorithm to rectify aerial video frames. It can be seen that the size and shape of the rectified images are substantially different from the source image shown in figure 2.11a. As pointed out earlier in section 1.1 that the rectified image generated from the existing image rectify algorithm for stereo vision is usually distorted and hence rectification algorithm use in stereo vision is not suitable to use to rectify the aerial image to be mosaic. Therefore, we develop a different rectification approach, called P2Prectify, specially designed to work with our proposed aerial video mosaic algorithm discuss in chapter 3. Based on section 2.1, point correspondences between 2 parallel projection frames (x, y) and

(x', y') can be written as (x, y) and $(x + \Delta tx, y + \Delta ty)$. The epipolar line passing through these points can be written as $y = mx + c, y' = m'x' + c'$ where $m' = m$ because epipolar lines passing through these points are parallel.

$$y' = m'x' + c'$$

$$y + \Delta ty = m(x + \Delta tx) + c'$$

$$mx + c + \Delta ty = mx + m\Delta tx + c'$$

$$c + \Delta ty = m\Delta tx + c'$$

$$m = \frac{c - c' + \Delta ty}{\Delta tx} \tag{2.8}$$

When UAV travel vertically $\Delta tx \approx 0$ and hence $m = \infty$. It does make sense for the epipolar line of the rectify image to be vertical since that is the principle direction of the UAV. To obtain a good quality mosaic, it is important that the transformed image have similar dimension as the source image as much as possible. Therefore, we are fixing the point where the epipolar line entered the image and rotate the epipolar line so that they are parallel.

The following is the proposed P2Prectify algorithm

i. Select a pair of video frames, $\mathbf{I_f}$, $\mathbf{I_{f+n}}$ that overlapping each other approximately by 50% and have a lot of feature points. Let $[\, w, h \,]$ be the frame size.

ii. Detect and match feature points between these two frames and use those points correspondences to find fundamental matrix and epipolar line using normalized 8-point algorithm described in [10]. For each epipolar line, calculate points that epipolar line intersect image border (i.e., $[\, 1 \;\; 0 \;\; -1 \,], [\, 1 \;\; 0 \;\; -w \,], [\, 0 \;\; 1 \;\; -1 \,], [\, 0 \;\; 1 \;\; -h \,]$)

iii. To exclude points near the video frame boundary, remove any epipolar line that doesn't span the entire image height for both frames. That can be done by removing any epipolar line that intersects the left (i.e., First column of image, $[\, \mathbf{1} \;\; \mathbf{i} \,]$) or right (i.e., last column of image, $[\, w \;\; i \,]$) image border.

iv. Let $((p_1,h),(p_2,h),...,(p_n,h))$ be the points on the bottom image border (i.e., last row of image with height, h) where epipolar lines enter and $((k_1,1),(k_2,1),...,(k_n,1))$ be the points on the top image border (i.e., first row of image) where epipolar lines exit. For both frames, rotate the remaining epipolar lines to be vertical by setting $ki = pi$. The

projection of the feature point (xi_o, yi_o) on the new vertical epipolar line [a 0 c], (xi_p, yi_p) is (pi, yi_o)

v. Using iterative algorithm to estimate homography as described in [10], estimate the geometric transformation of both frames, $\mathbf{H_f}$, $\mathbf{H_{f+n}}$, using the feature points (xi_o, yi_o) and corresponding projected feature points (xi_p, yi_p) as point correspondences.

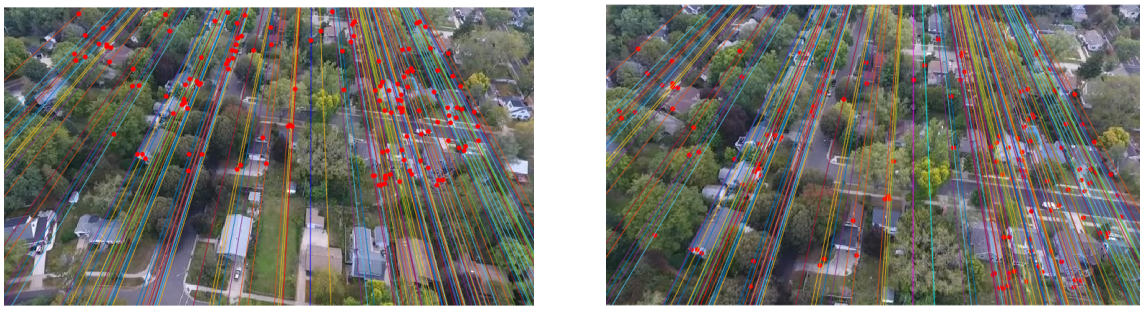
vi. Warp both images, $\mathbf{I_f}$, $\mathbf{I_{f+n}}$ using both transformations obtain in previous step.

vii. Warp successive frames $\mathbf{I_{f+1}}$, $\mathbf{I_{f+n+1}}$ using transformation obtain from step vi, $\mathbf{H_f}$, $\mathbf{H_{f+n}}$ respectively.

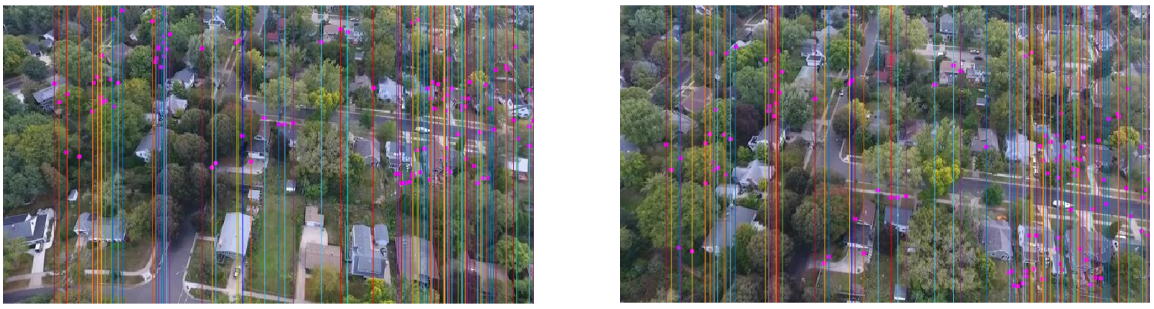
viii. Evaluate the track point motion vectors of both pair of the warped frame (i.e., $\hat{\mathbf{I}}_\mathbf{f}$, $\hat{\mathbf{I}}_{\mathbf{f+1}}$ and $\hat{\mathbf{I}}_{\mathbf{f+n}}$, $\hat{\mathbf{I}}_{\mathbf{f+n+1}}$) and select the transformation, $\mathbf{H_{rec}}$, that result in smaller standard deviation of the motion vector phase, mvp_{std}.

ix. Repeat step v - viii and only estimate the homography of, warp, and evaluate the track point motion vectors of, the frame associate with the selected transformation, $\mathbf{H_{rec}}$, until standard deviation of the motion vector phase, $mvp_{std} < \epsilon\,°$.

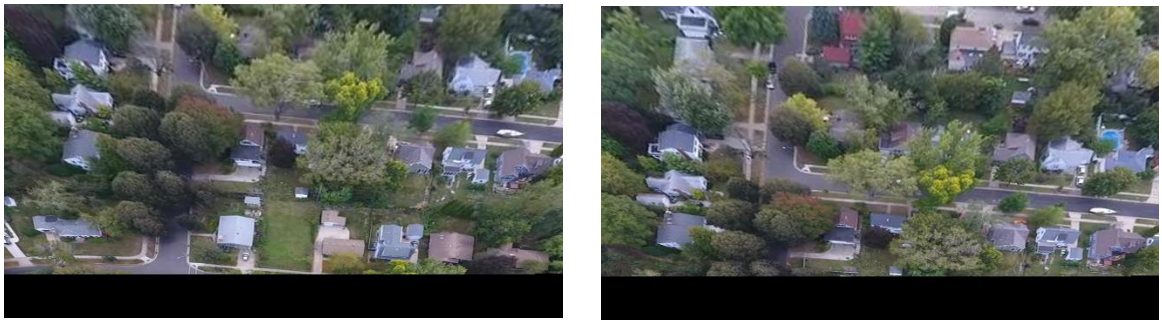
Figure 2.11 illustrates the P2Prectify process and figure 2.12 compares the motion vector of points in consecutive frames between an unrectified perspective view frame and a cropped P2Prectify frame. Based on the study in section 2.1, we empirically set the threshold in step ix of the P2Prectify to $\epsilon = 15\,°$. It can be seen that the magnitude and the phase of the motion vectors of the cropped P2Prectify frame are uniforms. The transformation obtains from P2Prectify can be applied to all the perspective view frames in the entire video segment with the same camera angle. It is common in aerial surveillance application that the UAV camera angle will be fixed for the entire operation. In that scenario, P2Prectify needs only be executed once.



(a)

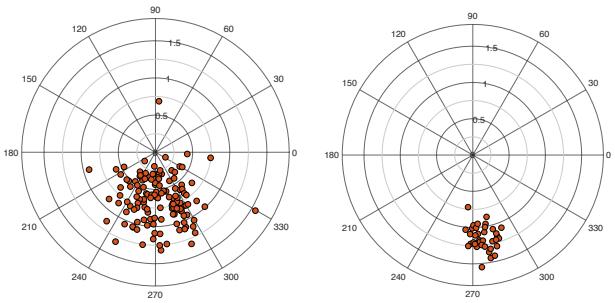


(b)



(c)

Figure 2.11: Example of P2Prectify process. (a) illustrate result of step 1-2 (b) illustrate result of step 3-5 (c) illustrate result of step 6-9 of the P2Prectify algorithm.



(a) (b)

Figure 2.12: Motion vectors of points in (a) perspective view frame (b) P2Prectify frame.

2.4 Summary

Table 2.1 compares the characteristics of stereo rectification and P2Prectify. Points in parallel images have uniform motion vectors. As a result, the parallel images can be mosaicked efficiently. Because the epipoles of all of the aerial images in the same flight mission are all at the same points,

a single transformation can be used throughout the entire video sequence with the same camera angle to transform perspective video frames to parallel video frames. This is crucial because rectified images must have the same viewpoint and size for them to be mosaicked seamlessly with no accumulated error. Therefore, it is more efficient to transform the perspective video to parallel video prior to creating a mosaic rather than minimizing the accumulated error afterwards as commonly done in other works. The goal of P2Prectify is to produce rectified images whose shapes resemble a rectangle as much as possible to minimize the distortion of the final panoramic image. Because of the large distance between the camera and the scene in aerial images, the distance between the points in parallel and perspective images of the same scene is small. And because the parallel images have smaller FOV than the perspective images, P2Prectify algorithm removes points on the edges of the image and modified the slope of the epipolar line within the image to ensure that the geometric relation of points on original epipolar lines and projected points on the modified slope epipolar lines are coherent.

Table 2.1: Summary of characteristics of stereo rectification and P2Prectify.

Stereo Rectification	**P2Prectify**
Epipolar line horizontally aligned	Vertical epipolar line
Projective + horizontal shear and translation	Projective
Projective transformation to move epipoles to infinity along x-axis and linear least square to compute horizontal shear while minimizing distortion	Modify slope by changing epipolar line and image border intersection points while minimizing the SD motion vector phase
Distort, size differs	Parallel projection, low distortion, size similar
Transformation unique per stereo pair	Single transformation for entire video sequence
Stereo	Aerial video mosaic

CHAPTER 3: MOTION-BASED VIDEO MOSAIC

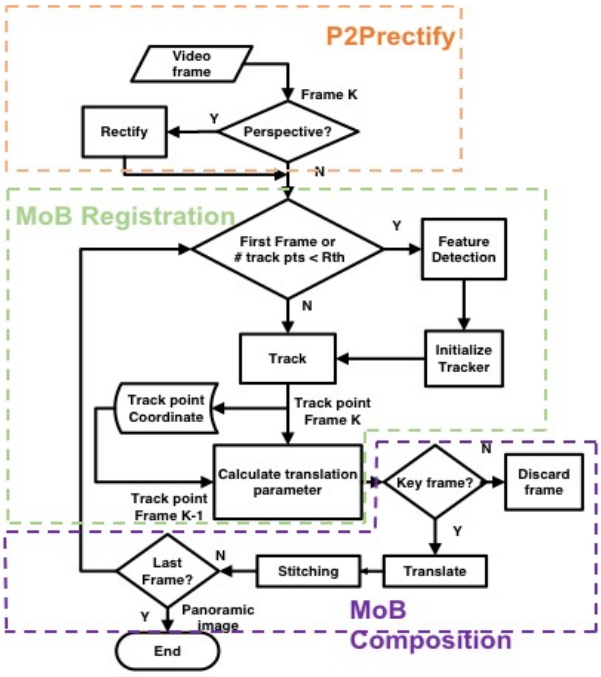


Figure 3.1: Block diagram of Motion-Based video mosaic.

A block diagram of the overview process of the proposed MoB mosaic algorithm is depicted in Figure 3.1. First, perspective videos will be rectified using the method proposed in chapter 2 before mosaicking. Then if the current number of valid feature points falls below a threshold, the FAST [67] feature detecting algorithm will be applied to the acquired frame. The FAST algorithm was chosen for its speed advantage over other more elaborate algorithms such as HARRIS [48], SIFT [49], or SURF [50] detectors. An adaptive non-maximal suppression algorithm [52] is then applied to evaluate a strength metric of the candidate feature points. Those surpassing a threshold (S_{th}) are selected as tracked feature points of the current frame. The feature point detection will be performed at the initial frame or when there is too little overlap between the consecutive frame which occurs when scenery change. Otherwise, it will not be performed. Instead, a Lucus-Kanade tracking (KLT) algorithm [62] will be performed. The current frame is then translated by the amount specified by the motion vector magnitude of the track points. Then only the selected key frames are composited into the mosaic. In this work, we consider the mosaicking of video clips shot from a camera onboard UAV with constant zoom and UAV flying at a constant height. We assume that the distance between the camera and the scene is sufficiently far such that the depth variations of foreground or background objects are negligible. And the displacement of any

moving foreground object over successive frames would also be negligible --- although it may be significant over longer periods, say tens or hundreds of frames. We also will not consider any occlusive objects (e.g., cloud) that may be much closer to the camera than the intended scene.

3.1 MoB registration

From chapter 2, motion vector of points in a consecutive parallel video frames is proportional to the drone camera's velocity and that the corresponding motion vectors of feature points in the same image are all the same. Thus, it is more efficient to identify matching feature correspondence by tracking feature points rather than detecting them independently in every frame and then finding matches among those features. We chose the computationally efficient Lucus-Kanade tracking (KLT) algorithm [62] to track features across frames due to its accuracy and fast convergence rate. Motion vectors can be trivially calculated from the location of the track points in each frame. The principle of the KLT algorithm is based on spatial intensity gradient guided search combine with the Newton-Raphson iterative optimization technique. The algorithm iteratively updates the transformation parameters until it converges. The goal of KLT is to find the transformation parameters, $\mathbf{p}$, such that the sum of square errors between the template ($\mathbf{T}$) and the image ($\mathbf{I}$) in the region of interest, $\mathbf{R}$, expressed by (3.1) is minimized.

$$\sum_{\mathbf{x}\in\mathbf{R}}\left[\mathbf{I}\left(\mathbf{W}(\mathbf{x};\mathbf{p})\right)-\mathbf{T}(\mathbf{x})\right]^{2} \tag{3.1}$$

Since we use KLT to track a feature from frame to frame, $\mathbf{T}(\mathbf{x})$ represents a small region, $\mathbf{R}$ (we use 5x5 window), surrounding the feature to be tracked at frame $\mathbf{t}$ and $\mathbf{I}\left(\mathbf{W}(\mathbf{x};\mathbf{p})\right)$ is the warped image patch at frame $\mathbf{t+1}$. Under the assumption that the UAV is engaging in constant height cruising, translation can be used as a warp function, $\mathbf{W}$, in (3.1) with translation parameters as the warp parameter $\mathbf{p} = [t_x, t_y]$. The algorithm assumes that the initial estimation of $\mathbf{p}$ is known and performs a minimization of (3.2) iteratively with respect $\Delta\mathbf{p}$ until it converges as shown in (3.3) and (3.4).

$$\sum_{\mathbf{x}\in\mathbf{R}}\left[\mathbf{I}\left(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})\right)-\mathbf{T}(\mathbf{x})\right]^{2} \tag{3.2}$$

$$\mathbf{p}\leftarrow\mathbf{p}+\Delta\mathbf{p} \tag{3.3}$$

$$\text{Repeat until } \|\Delta\mathbf{p}\|\leq\epsilon \tag{3.4}$$

The feature is dropped when the dissimilarity, measured by RMS residual of the feature between the current frame and the next frame, grows too large.

Once the number of valid track points within the overlapping region falls below a reinitialize threshold (r_{th}), FAST feature detection [57] will be performed on the current frame and those points with strength metric surpassing a threshold (S_{th}) will be selected as track features and used to reinitialize the tracker. We define the frame where the tracker is reinitialized as a break frame. The feature detection algorithm is time-consuming and computationally intensive. Therefore, r_{th} should be small to minimize the number of tracker reinitialization but not too small that it will affect the quality of the mosaic. Strong threshold (S_{th}) indicates the number of track points. It should be at least two times higher than r_{th} so that there are enough valid track points to prevent excessive tracker reinitialization but not so high as to slow the tracker down. When a scene has a highly fluctuated motion vector magnitude which occurs in scenes with moving objects with significant different relative velocity to the UAV or scenes with perspective projection, a high number of track points is required to better estimate the global motion. On the other hand, a high number of track points is not required for parallel projection scene with no moving object because it has a uniform motion vector magnitude.

Corollary 3.1. Geometric transformation model for parallel image is translation

Proof. The motion vectors of points in parallel image can be used to represent the frame global motion as follows. Let $(x_1,y_1)(x_1',y_1')$ and $(x_2,y_2)(x_2',y_2')$ be two pairs of point correspondences between two consecutive frames. Let $\mathbf{MV1} = \begin{bmatrix} mv1_x & mv1_y \end{bmatrix}$, and $\mathbf{MV2} = \begin{bmatrix} mv2_x & mv2_y \end{bmatrix}$ be motion vectors of (x_1,y_1) and (x_2,y_2). (3.5) is the linear equation of similarity transformation related these 2 pairs of point. (3.6) to (3.9) are solutions of (3.5).

$$\begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \end{bmatrix} = \begin{bmatrix} x_1 + mv1_x \\ y_1 + mv1_y \\ x_2 + mv2_x \\ y_2 + mv2_y \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 \\ y_1 & -x_1 & 0 & 1 \\ x_2 & y_2 & 1 & 0 \\ y_2 & -x_2 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ t_x \\ t_y \end{bmatrix} \tag{3.5}$$

$$\mathbf{X'} = \mathbf{HA}$$

$$\mathbf{A} = (\mathbf{H^T H})^{-1} \mathbf{H^T X'}$$

$$a_1 = 1 + \frac{c_x(x_1 - x_2) + c_y(y_1 - y_2)}{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.6}$$

$$a_2 = \frac{c_x(y_2 - y_1) + c_y(x_1 - x_2)}{-[(x_1 - x_2)^2 + (y_1 - y_2)^2]} \tag{3.7}$$

$$t_x = mv1_x + \frac{c_y(y_1 x_2 - x_1 y_2)}{-[(x_1 - x_2)^2 + (y_1 - y_2)^2]} \tag{3.8}$$

$$t_y = mv1_y + \frac{c_x(y_1 x_2 + x_1 y_2)}{-[(x_1 - x_2)^2 + (y_1 - y_2)^2]} \tag{3.9}$$

where $c_x = mv1_x - mv2_x$ and $c_y = mv1_y - mv2_y$

When motion vectors of the track points between two consecutive video frames are uniform, the values of c_x and c_y are very close to zero. Therefore, the fractions in (3.6) to (3.9) approach zero. As a result, $a_1 \approx 1$, $a_2 \approx 0$, $t_x = mv1_x$, and $t_y = mv1_y$.

3.2 MoB composition

Sequential aerial video frames have high temporal redundancy where the same scene can be seen in a video sequence over a long period of time. Due to the large overlap between consecutive frames, the most efficient way to stitch the frames is to only overlay the pixels in the portion of the current frame that does not overlap with the previous frame. Non-overlap stitching is suitable for orthographic video since it contains no perspective changes between consecutive frames. However, weak motion parallax is still present in the side perspective video. Therefore, we develop a stitching technique base on the Voronoi diagram called Incremental Orthographic (IO) stitching to deal with stitching side perspective video, hence P2Prectify need not be performed on this type of video.

3.2.1 Incremental Orthographic (IO) stitching

Referring to figure 3.2, the blue and green boxes represent the fields of view from two successive frames with optical center labeled as C and D respectively. Pixel position and orientation in the green box is transformed using the estimated displacement $\mathbf{t}_{xy}$ in section 3.1. The dashed line is perpendicular to the line segment CD and intersects it at its center. We stipulate that to the left side of the dashed line; the blue frame should be used as part of the stitched image. To the right of the

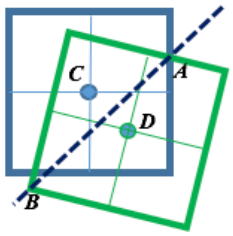


Figure 3.2: Overlap of successive frames and Voronoi partition. C and D are two optical centers.

dashed line, the green frame should be used. This Voronoi partition ensures pixels selected are closest to the corresponding optical center where no projective distortion occurs. Let **c** and **d** be the coordinates of C and D respectively. Then the dashed line can be described by the equation:

$$\frac{(\mathbf{c}-\mathbf{d})^T \mathbf{r}}{\|\mathbf{c}-\mathbf{d}\|} = \frac{1}{2}\|\mathbf{c}-\mathbf{d}\| \tag{3.10}$$

where $\mathbf{d} = \mathbf{R}_{xy}\mathbf{c} + \mathbf{t}'_{xy}$

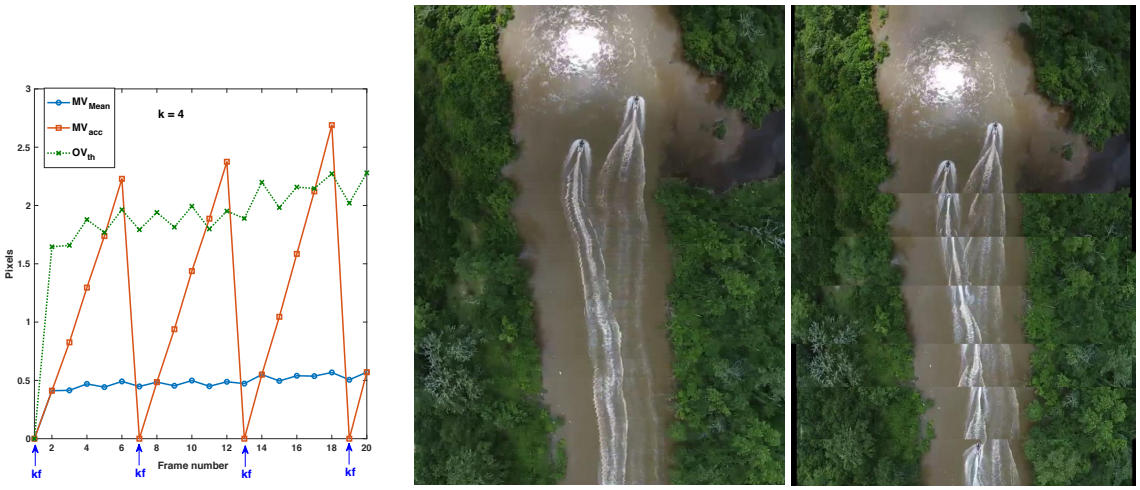
Since we always transform perspective images into orthographic images before creating a mosaic, the stitched video frame is always orthographic and hence $\mathbf{R}_{xy} = \mathrm{I}$. Therefore, once $\mathbf{t}_{xy}$ is estimated, we simply translate the current frame (green box) into the stitched image coordinates (blue box) and then, test the new coordinates against (3.10). If the pixel coordinate falls to the left of the line, it will be removed from the green box. Finally, the intensity values of the remaining green box will overwrite those of the stitched frame at corresponding pixel coordinates.

3.2.2 Motion vector guided key frame selection

Aerial video is temporally and spatially dense therefore can be represented by a subset of frames without losing surveillance information or any search targets. By compositing only selected key frames, compositing time can be reduced substantially. The mean of the motion vectors for each frame is accumulated until the accumulated value reaches a certain overlap threshold (OV_{th}). We define the value of OV_{th} as a function of the mean of the motion vector magnitude as follows

$$OV_{th} = \mathrm{k} \cdot \langle |\mathbf{MV}| \rangle \tag{3.11}$$

Since we assume that the flight velocity is constant, we set the overlap threshold factor, k, to be constant. The frame associated with an accumulated motion vector value equal or above OV_{th} is selected as a key frame. Only the key frames and the break frames (frames where tracker re-initialization is activated) need to be composited into the mosaic. The higher the value of OV_{th}, the lower the number of key frames will be. Based on our experiment on various videos, we found the frequency of key frame selection is every 3-4 frames when k = 2, every 5-6 frames when k = 4, and every 11-12 frames when k = 10. Figure 3.3 illustrates key frame selection when k = 4 and the effect of different OV_{th}. When there are moving objects in the scene, it is vital that the value of k is small enough to prevent motion parallax from affecting the quality and the correctness of the final panorama as shown in figures 3.3b and 3.3c.



(a) (b) (c)

Figure 3.3: (a) Key frame selection in 20 frames span with k = 4. (b) MoB with k = 2. (c) MoB with k = 50.

Using this method to reduce the number of frames to be composited into the mosaic, along with using the proposed incremental orthographic stitching or non-overlap stitching describe in section 3.2 as compositing mode to reduce the number of pixel value overwrites per frame provides a large reduction in compositing time.

3.2.3 Stitching mode selection

IO stitching and non-overlap stitching reduces the number of redundant pixels overlay and hence reducing the composition time and bandwidth. Parallel image created from strictly downward-pointing camera or P2Prectify can be stitch with non-overlap stitching because parallel projection

and translation transformation preserve shape and geometric, so the mosaic does not need to be updated by overlay all the pixels. IO stitching can be used with weak perspective images (i.e., weak parallax, and no projective distortion) due to its orthographic pixel selection mechanism. When the scenes involve non-stationary objects that are in the foreground, 100% overlay stitching might be necessary to correctly update the position of the non-stationary object. Figure 3.4 shows an example of pixel selection of each stitching mode.

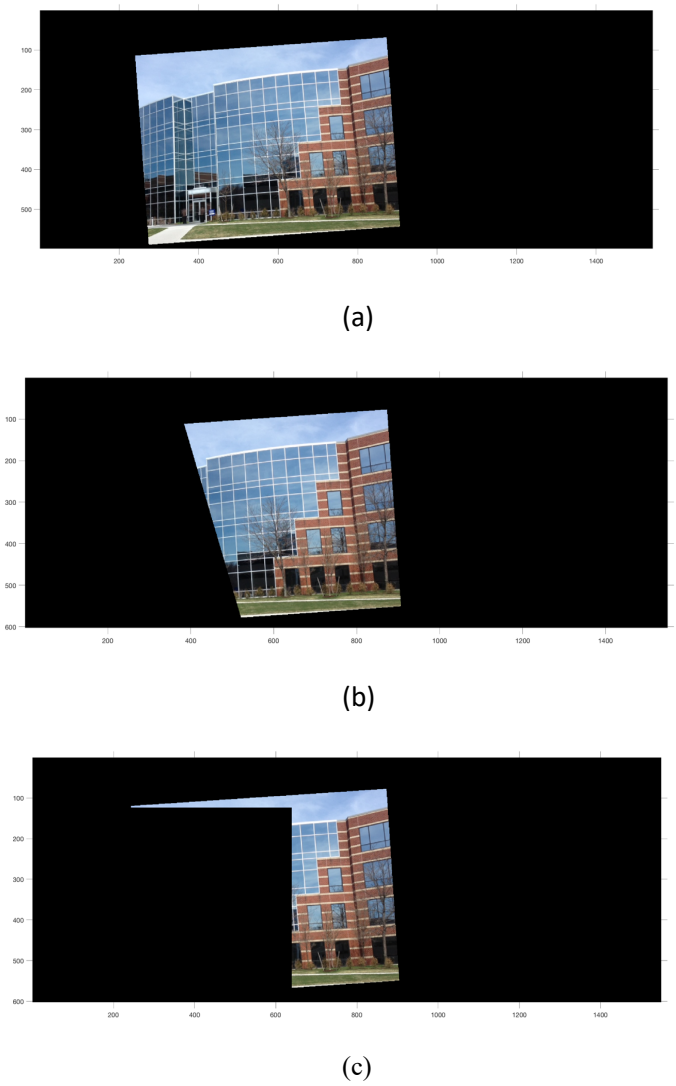


Figure 3.4: Pixel selection of different stitching modes. (a) 100% overlay (b) IO (c) Non-overlap. Source image: MATLAB.

3.3 Summary

MoB video mosaic improves the registration process by tracking the features across the frames instead of detecting and matching the features independently frame by frame. This significantly reduces the number of feature detection operations and eliminates the matching operation. Furthermore, MoB registration uses 2 DoF translation model whose parameters can be obtained directly from track points locations rather than other transformations with higher DoF such as projective or affine transformation whose parameters must be estimated from matched feature correspondences.

In addition to scaling and skewing components in those transformations with higher DoF, the misregistration between image pairs due to mismatch point correspondence caused accumulated error that distort the mosaic over time. Minimizing the accumulated error requires minimizing the misregistration between all image pairs to find a globally consistent set of transformation parameters and optimization with large data, which is time consuming and computationally intensive. Therefore, it is not practical to perform on video mosaic due to the large number of features and frames. Contrary, MoB mosaic uses translation obtained from the motion vector instead of projective transformation to align frames, and there is no accumulated error that distorts the mosaic over time.

Motion vector properties are used in MoB composition to select key frames and stitching modes. Compositing only selected key frames and using only selected pixels to stitch reduces the composition time substantially.

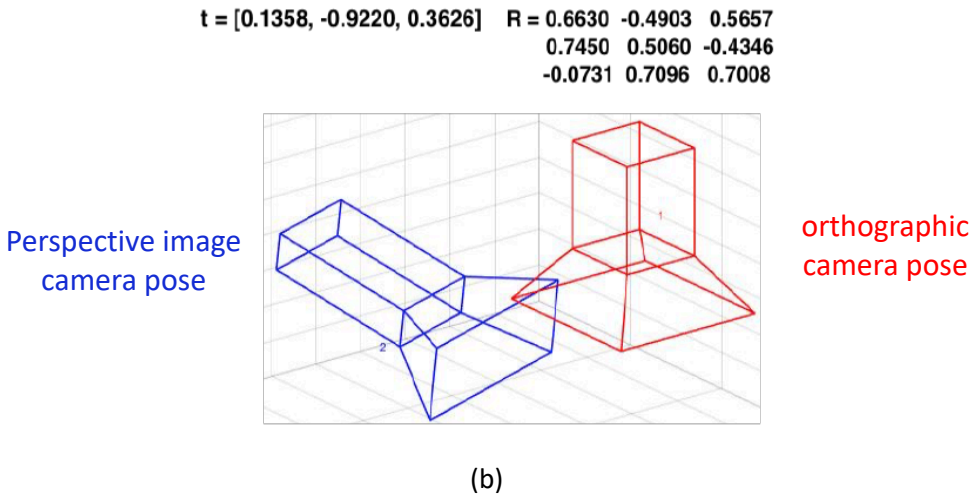
CHAPTER 4: EXPERIMENTS AND PERFORMANCE

Our algorithms are implemented in MATLAB version R2017b and our experiments are conducted on a 2009 IMAC with 2.66 GHZ intel 2 core duo processor, 8 GB DDR3 memory, and NVIDIA GeForce 9400 graphic card. To test the robustness of our algorithm, we include test videos that consist of moving objects, perspective view, and less texture area, in addition to the orthographic video. Our experiment consists of two orthographic videos (River, Big C), one side perspective video (Side) and two perspective videos (Perspective, Horseshoe). Experiments on flight paths containing altitude change, angle change, and slanting are also evaluated. The resolution of all the videos is 640×360.

Furthermore, MoB is applied to the UMCD dataset [69] and its performance is compared against the mosaic algorithm proposed in [69].

4.1 P2Prectify algorithm

(a)

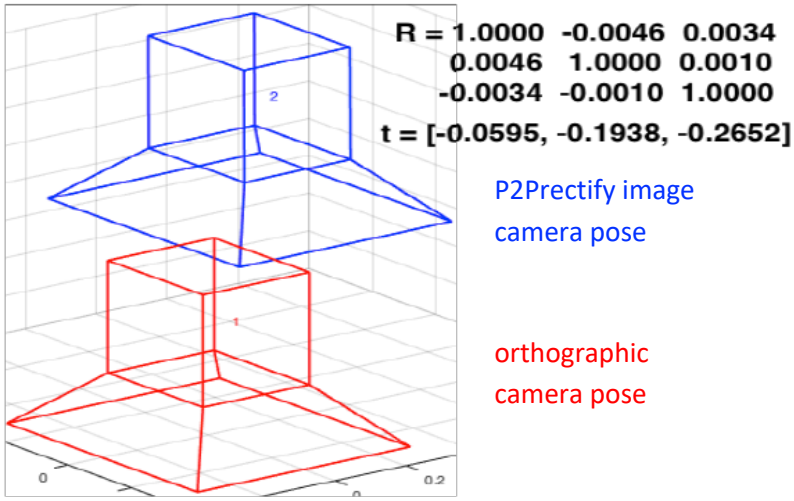


(b)

Figure 4.1. (a) Epipolar lines of orthographic and perspective images (b) Camera pose of perspective image relative to the camera pose of orthographic image.

To evaluate our P2Prectify algorithm, we obtain the UAV intrinsic parameter by performing camera calibration [68]. Once we obtain the UAV intrinsic parameter, we can estimation the virtual camera pose of the P2Prectify frame relative to the camera pose of the orthographic frame from the matching correspondence. Figures 4.1 and 4.2 show epipolar lines and relative camera pose between orthographic image and perspective image before and after rectifying the image with our proposed P2Prectify algorithm respectively. It can be seen that the epipolar lines in figure 4.2a are parallel and the orientation of the relative camera pose in figure 4.2b is approximately equals to the identity matrix. These results indicate that the P2Prectify algorithm successfully transforms the perspective image to orthographic image.

(a)



(b)

Figure 4.2. (a) Epipolar lines of orthographic and P2Prectify images (b) Virtual camera pose of P2Prectify image relative to the camera pose of orthographic image.

4.2 MoB algorithm

To compare our mosaic algorithm with traditional feature-based image mosaic algorithm (FB), we implement a feature-based image mosaic algorithm consists of feature detection and matching, robust estimator with no bundle adjustment for estimating the geometric transformation from matching points correspondence, linear interpolation for warping the image, and composition by overlay images with no further composite enhancement. Higher number of inliers improves the accuracy and increases the rate of convergence of the robust estimator, so the number of feature points is set to 200.

There are a few parameters in our MoB algorithm which we empirically set, and test based on our experiment as follows:

Reinitialize threshold (r_{th}): The tracker is reinitialized when the number of track feature points fall below r_{th}. Since reinitialization require the use of the computationally intensive feature detection algorithm, setting this threshold too high can slow the algorithm down due to excessive reinitialization. On the other hand, setting this threshold too low can negatively affect the quality of the mosaic due to insufficient amount of points to accurately represent global translation motion. Therefore, $r_{th} = 8$, even though translation has 2 degree of freedom that can be computed with a minimum of a pair of point correspondence.

Strong threshold (S_{th}): S_{th} indicates the number of track points. Experiment shows that the number of tracker reinitializations is substantially reduced for all cases when $S_{th} = 2.5\ r_{th}$. However, when the magnitude of the motion vectors fluctuates --- which is the cases with videos containing perspective changes and moving objects with significant different relative velocity to the UAV, a higher number of feature points is essential for a better estimation of global motion. Therefore, $S_{th} = 200$ in experiments with perspective videos (perspective and horseshoe) and video with moving objects (BigC). Otherwise, $S_{th} = 20$.

Overlap threshold factor (k): To maintain reasonable trade-off between number of key frames and mosaic quality, k = 10 in most of the experiments except the river experiment. For the river experiment, more key frames are needed to achieve seamless panoramic image. In which case, k = 2 to ensure that the position of the boats and the wake are portrayed accurately in the mosaic.

***Stitching mode**:* We use IO stitching to stitch the Side video as discussed in section 3.3. All perspective videos (Perspective and Horseshoe) are P2Prectified and thus, can be treated as orthographic videos. All orthographic videos (including P2Prectified perspective videos) are then stitched with non-overlap stitching. One exception applies to the River video, where the UAV is flying at a high speed and at a low altitude. In this case, it is necessary to overlay all pixels in the current frame on top of previous frame (100% overlay) to ensure that the boat position and the wake made by them are constantly updated.

Table 4.1: MoB parameters.

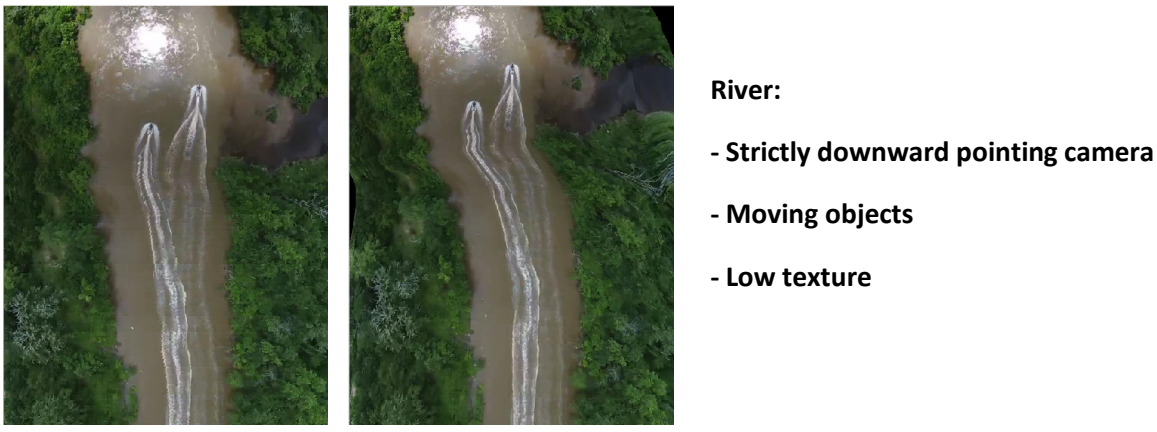
	River	**Big C**	**Side**	**Perspective**	**Horseshoe**
# Frame	301	3735	1387	1959	2691
# key frame	88	338	124	178	238
# break frame	5	13	8	69	28
P2Prectify	No	No	No	Yes	Yes
reinitialize threshold (r_{th})	8	8	8	8	8
strong threshold (s_{th})	20	200	20	20	20
overlap threshold, k	2	10	10	10	10
composite mode	100% overlay	Non-overlap	IO	Non-overlap	Non-overlap

Table 4.1 provides a summary of the parameters used in MoB. The number of break frames (see section 3.1) indicates the number of feature detections used in MoB registration and the number of key frames (see section 3.2) indicates the number of frames used in MoB composition. P2Prectify (see chapter 2) is applied to perspective video frames to transformed them into orthographic frames. The MoB registration parameters, the MoB composition parameters, and the stitching mode are defined according to the discussion earlier in this section. P2Prectify videos have more break frames because FOV is reduced when rectified. The Perspective video is shot from a faster speed UAV than the Horseshoe video, resulting in a higher number of break frames.

4.3 MoB experimental results

We will now present and discuss the results of our MoB algorithm and compare with the results of the FB algorithm performed on the five videos described earlier in this section.

In figure 4.3, although FB algorithm is using larger sampling of feature points (200) than MoB (S_{th} = 20), MoB captures the movement of the boats and the wake accurately using only 88 key frames and yields better quality mosaic.



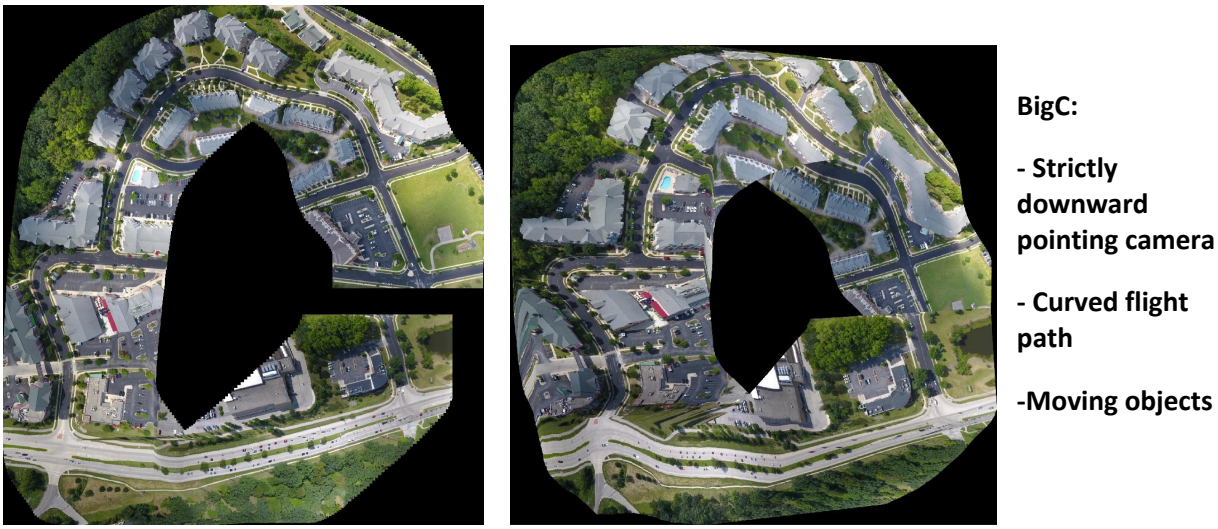
(a) (b)

Figure 4.3: Video mosaic result of River video using (a) MoB with 88 key frames and S_{th} = 20, and 100% overlay stitching (b) FB.

In figure 4.4, despite the scene complexity and part of this video is created from UAV with yaw motion, our algorithm still can produce high quality mosaic while the mosaic produced from FB algorithm is highly distorted.

Accumulated error strongly affects the quality of perspective video mosaic produced from FB algorithm as shown in figures 4.5b, 4.6b, 4.7b but does not affect our algorithm. Figure 4.5a shows our IO stitching orthorectification capability.

Applying P2Prectify algorithm to perspective video before registering them substantially improve the quality of the mosaic shown in figures 4.6a and 4.7a especially the quality of the perspective mosaic created from perspective video contain UAV yawing motion as seen in figure 4.7a.



(a) (b)

Figure 4.4: Video mosaic result of BigC video using (a) MoB with 338 key frames, S_{th} = 200, and non-overlap stitching (b) FB.

(a)

Side:

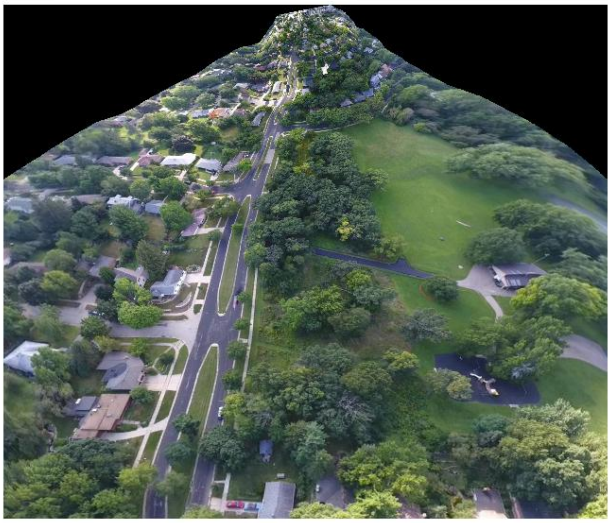
Angled-view camera

(b)

Figure 4.5: Video mosaic result of side video (a) MoB with 124 key frames, S_{th} =20, and IO stitching (b) FB.

Perspective:

Angled-view camera



(a) (b)

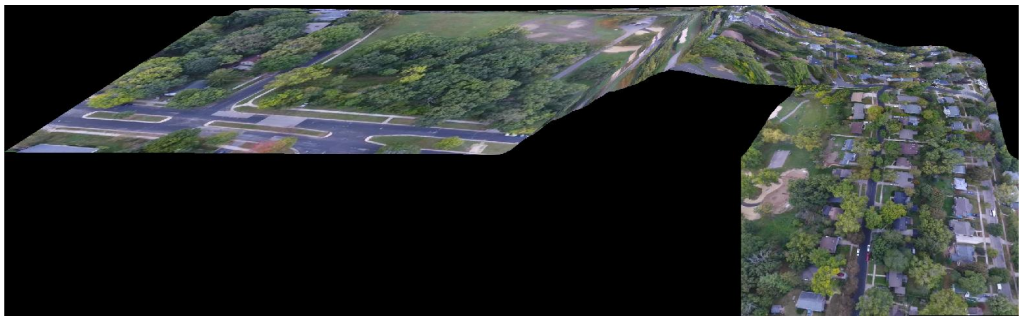
Figure 4.6: Video mosaic result of Perspective video using (a) MoB with P2Prectify, 173 key frames, S_{th} = 20, and non-overlap stitching *(*c) FB.

Horseshoe:

- Angled-view camera

- Curve flight path

(a)



(b)

Figure 4.7: Video mosaic result of horseshoe video using (a) MoB with P2Prectify, 238 key frames, S_{th} = 20, and non-overlap stitching (b) FB.

Table 4.2 compares the registration, the composition, and the mosaic time per frame between MoB and FB. Although Perspective and Horseshoe videos have more break frames and need to be rectified, the registered time increases only slightly. This is because there is still a very low number of break frames compared to the number of frames of the entire video and only key frames are rectified. The number of break frames indicates the number of feature detections used in MoB registration and the number of key frames indicates the number of frames used in MoB composition.

Table 4.2: P2Prectify and MoB execution time.

	$t_{register}$ (msec)		$t_{composite}$ (msec)		t_{total} (msec)	
	MoB	FB	MoB	FB	MoB	FB
River	14.2	84.4	15.4	50.5	29.6	134.9
Big C	13.9	95.7	16.1	214.1	30	309.8
Side	15.6	116.1	11.07	78.5	26.67	194.6
Perspective	19.17*	142.2	8.4	39.1	27.57	181.3
Horseshoe	12.68*	123	6.8	280	19.48	403

* mosaic time including P2Prectify

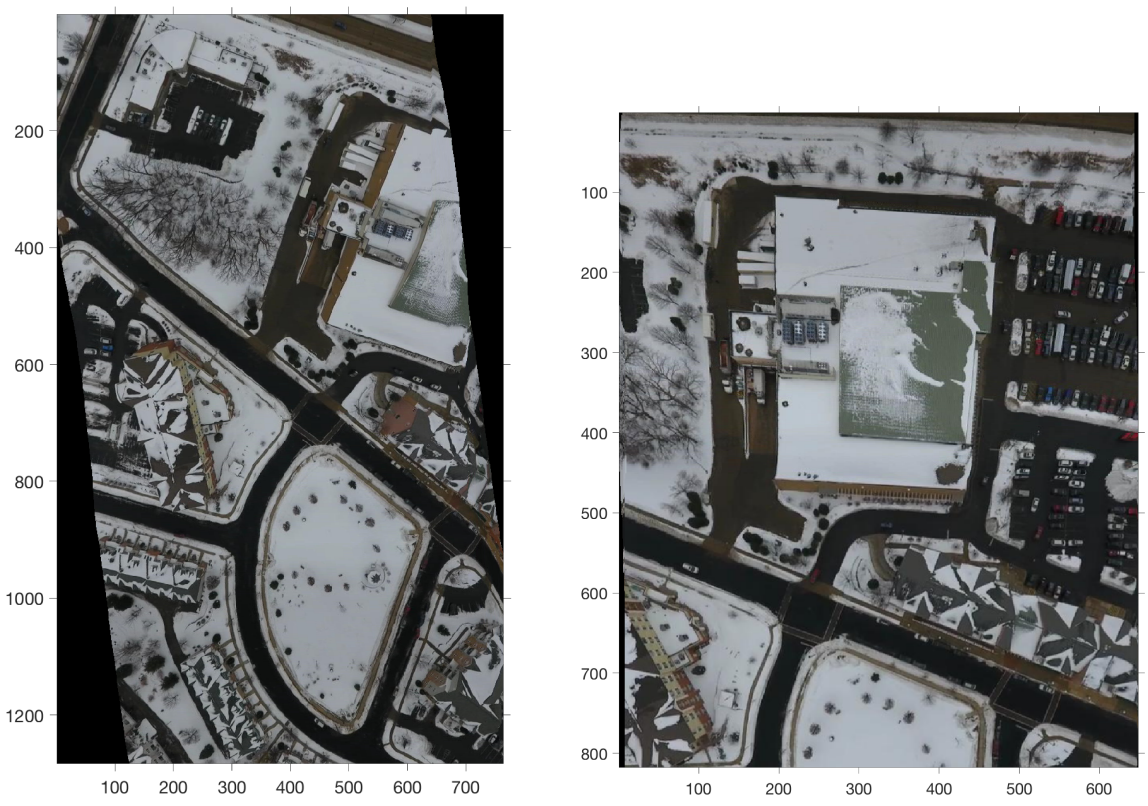
All the mosaics created from MoB use only a translation vector derived directly from the mean of the motion vectors. As a result, the execution time of MoB registration remains quite constant between different videos. The composite time of both P2Prectify video is less due to the reduced FOV, so there are fewer pixels to composite. Both registration and composition time of MoB is significantly smaller than FB. This is achieved by (a) reducing the number of feature detections, (b) simplifying the registration process by using only motion vectors, and (c) reducing the number of frames used in composition. In addition to reducing the total mosaic time, MoB also reduces the number of redundant pixels overlay operations by approximately 50% when using IO stitching and more than 87% when using non-overlap stitching.

Figure 4.8 shows mosaics generated from two non-constant flight altitude videos with different camera zoom using MoB. In this experiment, flight altitude gradually changes from 350 ft to 300 ft. Since the distance from the camera and the scene is large, MoB can still handle the gradual change in altitude.

The robustness of MoB is further tested against other algorithm [69]. We apply MoB to the UMCD data set [69] and the resulting panoramic images are compared against those reported in [69]. Figures 4.9 - 4.10 clearly show that MoB yields panoramic images with lower distortion. It can be seen that the accumulated error caused the UMCD panoramic images to taper while MoB does not.

Figure 4.11 shows 2 pairs of frames in which camera angle change causing the SD of the motion vector magnitude to spike. Therefore, P2Prectify is applied to the key frames that are perspective indicated by the high SD of motion vector magnitude. Figure 4.12 shows the panoramic images of the video with camera angle change. It can be seen that applying P2Prectify to the perspective portion of the video enables the frames to align correctly due to all the frames have the same orthographic projection view while the panoramic image generated from FB mosaic is severely distorted.

Figure 4.13 shows that our P2Prectify and MoB algorithms are capable of creating a low-distortion panoramic image from a slant flight path perspective video.



(a) (b)

Figure 4.8: Video mosaic result from 2 videos with flight altitude change from 350 ft to 300 ft with different camera zoom.

(a)

(b)

Figure 4.9: Video mosaic result of UMCD data set [69] using (a) MoB (b) method in [69].

(a) (b)

Figure 4.10: Video mosaic result of UMCD data set [69] using (a) MoB (b) method in [69].

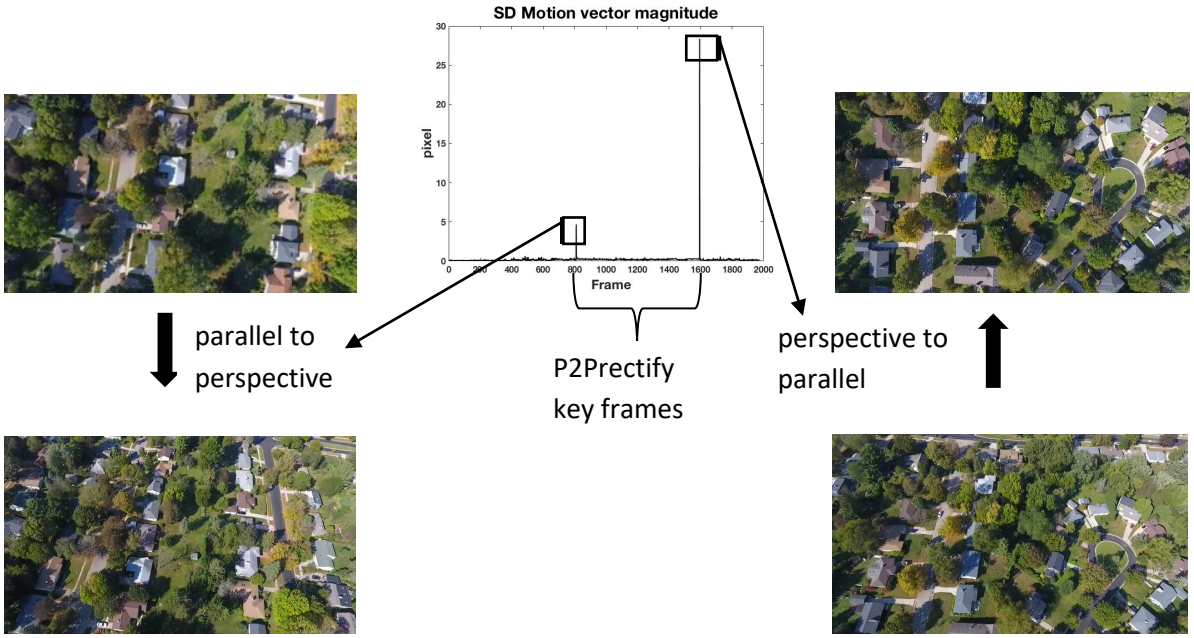


Figure 4.11: Camera angle change mid-flight causing SD of motion vector magnitude to spike.

(a) (b)

Figure 4.12: Video mosaic result of video with camera angle change mid-flight using (a) MoB with P2Prectify (b) FB.

(a) (b)

Figure 4.13: Video mosaic result of perspective video with slant flight path using (a) MoB with P2Prectify (b) FB.

CHAPTER 5: CONCLUSIONS AND FUTURE DIRECTIONS

5.1 Conclusions

Our proposed P2Prectify algorithm rectifies the perspective image to parallel image so that it can be composited into a mosaic using less time and less computation and still yield high quality. Since the image produced by UAV can be considered as planar, we believe it is feasible to generate the parallel image using only point correspondences between the two perspective images. As a result, we use the epipolar geometry to facilitate the rectification process which we be able to successfully rectify the perspective image to orthographic image using only point correspondences. Although rectifying images without the knowledge of the position of the points in the scene can cause some error in the rectified image, it does not impact the overall quality of the mosaic generated from them using our proposed MoB mosaic algorithm. This due to MoB mosaic algorithm use only motion vector to translate the frame to be composite, and hence highly robust against noise. Our proposed MoB mosaic algorithm produces a fast mosaic with less computation and no accumulated error. Since the motion vector of each track feature point is a byproduct of the KLT tracker, which can be computed with a minimal impact on computational overhead, the amount of time and computation required for the image registration process is substantially reduced. Furthermore, motion vector guided composition and selective pixel stitching algorithms (i.e., non-overlap, IO) reduce composition time and bandwidth utilization substantially by compositing only selected key frames and overlaying only selected pixels while still maintaining a high-quality resultant mosaic. Our proposed IO stitching can be used to composite weak perspective images so that they do not

Table 5.1: Summary of types of aerial video, P2Prectify and MoB applicability, and characteristics.

Image type	P2Prectify	Stitching mode	MV phase	MV magnitude
Parallel	No	Non-overlap	Uniform	Uniform
Perspective	Yes	Non-overlap	Fluctuate	Fluctuate
Side	NO	I/O	Uniform	Slightly fluctuate

need to be rectified. Table 5.1 gives a summary of the types of aerial videos presented in this research and their associated P2Prectify and MoB mosaic methods along with their characteristics.

5.2 Future directions and open issue

Further development of P2Prectify

- Frame selection for P2Prectify: Develop selection criteria for good source frame pair. For example, study the overlap pattern and overlap size that will yield a good rectify result.
- Feature selection for P2Prectify: Currently, the P2Prectify algorithm only filters out points at the edges where the epipolar lines do not span the entire image. It would be beneficial to further investigate what property (e.g., motion vector pattern) and location of the features that will yield a good rectify result.

Further improvement of MoB mosaic

Various thresholds dictate the performance of our algorithm as discuss in section 4.2. This research provides an algorithm that works with constant flight parameters (e.g., velocity, altitude, and camera angle) and hence determining the value of these thresholds empirically works well. It will be interesting to create a mechanism to adaptively change the value of these thresholds when flight parameters change. For example, using motion vector magnitude to detect speed change or the presence of moving objects and use it to change the frequency of the key frame selection by changing the value of the overlap threshold. Using motion vector magnitude to identify moving objects and adaptively change the stitching mode with the presence of moving objects in the scene. Using the pattern of the motion vector phase to detect perspective change and activate the P2Prectify algorithm or adapt the stitching mode. A large spike in the motion vector phase can indicate the frame that should be excluded from key frame selection as shown in figure 5.1. In this case, the windy condition causes this anomaly.

In this research, since we assume that the flight velocity is constant, we set the value of the overlap threshold factor, k, as a constant. However, for non-constant flight velocity, the value of k should be such that it is inversely proportional to the motion vector magnitude. Setting the overlap threshold factor in this manner is practical since when the motion vector magnitude is large, k is small and hence, more key frames are selected to prevent loss of surveillance. Conversely, when

the motion vector magnitude is small, k is large, and fewer key frames are selected to reduce excessive video frame overlay. Therefore, the relation between the flight velocity and the value of k should be established so that the function is bounded. For example,

$k = U - \langle \|\mathbf{mv}\| \rangle$ where U is the upper range of k and $\langle \|\mathbf{mv}\| \rangle$ is mean of the normalized motion vector magnitude between 1 and n.

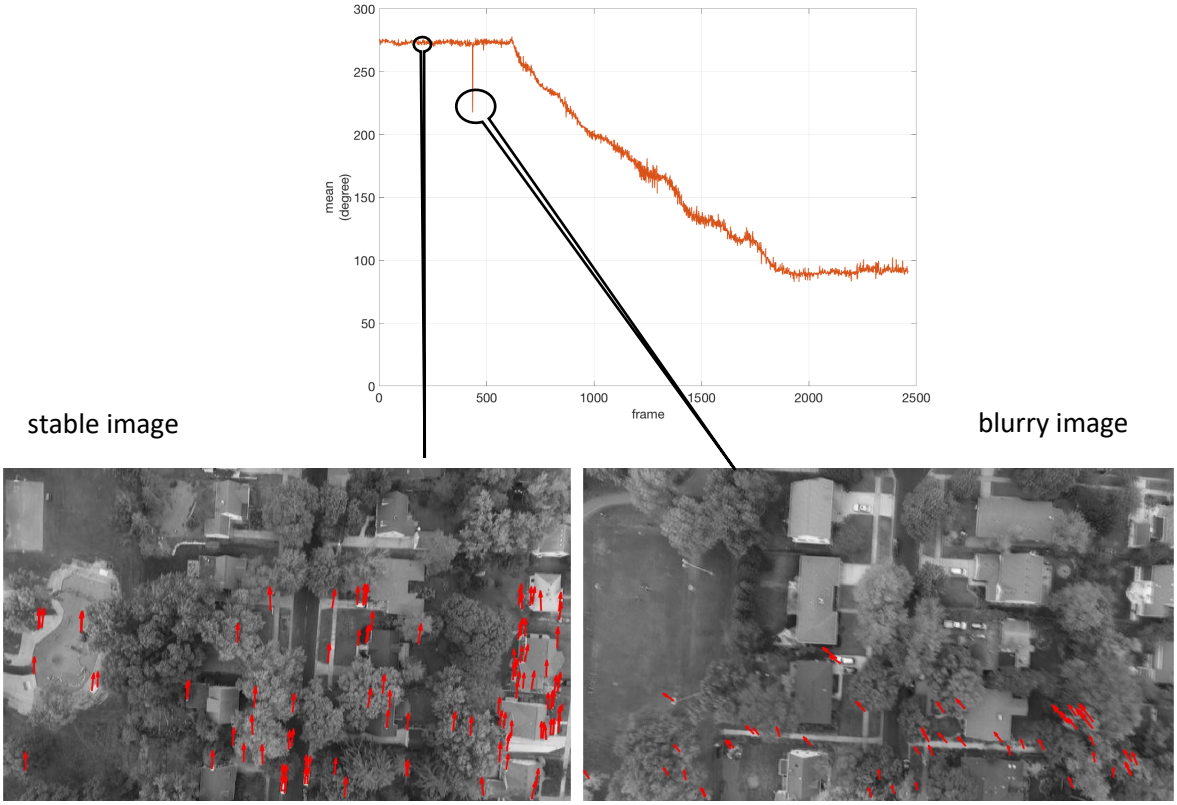


Figure 5.1: Anomaly-frame detection with motion vector phase.

Open issue about moving object

Currently, MoB can handle moving objects in the scene well since the distance between the camera onboard UAV and the scene is large, the motion of the moving objects is typically small compared to the camera-induced scene motion. Therefore, they do not cause motion parallax and can be stitched using MoB. However, in low-altitude flights or flights that produce perspective videos, moving objects may cause motion parallax and may affect the quality of the mosaic created from MoB. Although aerial video tracking is a well-studied application [14], [70], it is not clear how to deal with moving objects in video mosaic application. In tracking application, the static

background mosaic is constructed from a finite amount of video frames and usually remain unchanged for the entire operation. The moving objects are then tracked across the background mosaic. On the other hand, the video mosaic application needs to stitch an infinitely amount of the incoming video frames. How to identify and represent the moving objects in the mosaic remains an interesting issue.

REFERENCES

[1] http://www.dji.com/product/phantom-3

[2] Zhou, Guoqing. "Near real-time orthorectification and mosaic of small UAV video flow for time-critical event response." *Geoscience and Remote Sensing, IEEE Transactions on* 47.3 (2009): 739-747.

[3] Rango, Albert, et al. "Unmanned aerial vehicle-based remote sensing for rangeland assessment, monitoring, and management." *Journal of Applied Remote Sensing* 3.1 (2009): 033542-033542.

[4] Koh, L., and S. Wich. "Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation." *Tropical Conservation Science* 5.2 (2012): 121-132.

[5] Laliberte, Andrea S., et al. "Multispectral remote sensing from unmanned aircraft: Image processing workflows and applications for rangeland environments." *Remote Sensing* 3.11 (2011): 2529-2551.

[6] Frankl, Amaury, et al. "Using image-based modelling (SfM–MVS) to produce a 1935 ortho-mosaic of the Ethiopian highlands." *International Journal of Digital Earth* 8.5 (2015): 421-430.

[7] de Sá Rodrigues da Silva, Mariana, et al. "Testing DJI Phantom 4 pro for urban georeferencing." *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 42.1 (2018).

[8] Anurogo, Wenang, et al. "A simple aerial photogrammetric mapping system overview and image acquisition using unmanned aerial vehicles (uavs)." *Geospatial Information* 1.1 (2017).

[9] Sheng, Yongwei, and Douglas E. Alsdorf. "Automated georeferencing and orthorectification of Amazon basin-wide SAR mosaics using SRTM DEM data." *IEEE Transactions on Geoscience and Remote Sensing* 43.8 (2005): 1929-1940.

[10] Juel, Anders, et al. "Spatial application of random forest models for fine-scale coastal vegetation classification using object based analysis of aerial orthophoto and DEM data." *International Journal of Applied Earth Observation and Geoinformation* 42 (2015): 106-114.

[11] Zhang, Yunsheng, et al. "Seamline optimisation for urban aerial ortho-image mosaicking using graph cuts." *The Photogrammetric Record* 33.161 (2018): 131-147.

[12] Chen, Q., Sun, M., Hu, X., Zhang, Z., 2014. Automatic Seamline Network Generation for Urban Orthophoto Mosaicking with the Use of a Digital Surface Model. Remote Sensing 6, 12334-12359.

[13] R. Kumar *et al*., "Aerial video surveillance and exploitation," in *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1518-1539, Oct. 2001.

[14] Sadd Ali, Mubaraj Shah, “COCOA: tracking in aerial imagery”, Proc. SPIE 6209, Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications III, 62090D (5 May 2006).

[15] Morse, B.S.; Gerhardt, D.; Engh, C.; Goodrich, M.A.; Rasmussen, N.; Thornton, D.; Eggett, D., "Application and evaluation of spatiotemporal enhancement of live aerial video using temporally local mosaics," *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, vol., no., pp.1,8, 23-28 June 2008

[16] Szeliski, Richard. "Image mosaicing for tele-reality applications." *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*. IEEE, 1994.

[17] R.Hartley and A.Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, NY, USA, 2000. ISBN 0521623049.

[18] Triggs B., McLauchlan P.F., Hartley R.I., Fitzgibbon A.W. (2000) Bundle Adjustment - A Modern Synthesis. In: Triggs B., Zisserman A., Szeliski R. (eds) Vision Algorithms: Theory and Practice. IWVA 1999. Lecture Notes in Computer Science, vol 1883. Springer, Berlin, Heidelberg

[19] Szeliski, Richard. "Image alignment and stitching: A tutorial." *Foundations and Trends in Computer Graphics and Vision* 2.1 (2006): 1-104.

[20] M. Uyttendaele, A. Eden and R. Skeliski, "Eliminating ghosting and exposure artifacts in image mosaics," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Kauai, HI, USA, 2001, pp. II-II.

[21] Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D. H., and Cohen, M. F. (2004). Interactive digital photomontage. ACM Transactions on Graphics (Proc. SIGGRAPH 2004): 292 – 300.

[22] Szeliski, R., Uyttendaele, M., and Steedly, D. (2008). Fast Poisson Blending using Multi-Splines. Technical Report MSR-TR-2008-58, Microsoft Research.

[23] H. S. Sawhney and R. Kumar, "True multi-image alignment and its application to mosaicing and lens distortion correction," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 3, pp. 235-243, March 1999.

[24] https://www.flir.com/products/black-hornet-prs/

[25] https://news.utexas.edu/2013/07/29/ut-austin-researchers-successfully-spoof-an-80-million-yacht-at-sea/

[26] P.R.Wolf. *Elements of photogrammetry*. McGraw-Hill, New York, 1974

[27] Forstner, W., Wrobel, B. P., *Photogrammetric Computer Vision*. Springer, 2016

[28] Sheikh, Yaser, et al. "Geodetic alignment of aerial video frames." *Video Registration*. Springer, Boston, MA, 2003. 144-179.

[29] Hartley, Richard I. "Theory and practice of projective rectification." *International Journal of Computer Vision* 35.2 (1999): 115-127.

[30] Papadimitriou, Demetrios V., and Tim J. Dennis. "Epipolar line estimation and rectification for stereo image pairs." *IEEE transactions on image processing* 5.4 (1996): 672-676.

[31] Loop, Charles, and Zhengyou Zhang. "Computing rectifying homographies for stereo vision." *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. Vol. 1. IEEE, 1999.

[32] Fusiello, Andrea, and Luca Irsara. "Quasi-euclidean uncalibrated epipolar rectification." *2008 19th International Conference on Pattern Recognition*. IEEE, 2008.

[33] Gluckman, Joshua, and Shree K. Nayar. "Rectifying transformations that minimize resampling effects." *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. IEEE, 2001.

[34] Isgro, Francesco, and Emanuele Trucco. "Projective rectification without epipolar geometry." *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. Vol. 1. IEEE, 1999.

[35] Seitz, Steven M., and Charles R. Dyer. "View morphing." *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996.

[36] https://www.microsoft.com/en-us/research/product/computational-photography-applications/image-composite editor/

[37] http://matthewalunbrown.com/autostitch/autostitch.html

[38] https://www.ptgui.com/

[39] Rumama Aktar *et al*.,"Geospatial content summarization of UAV aerial imagery using mosaicking", Proc. SPIE 10645, Geospatial Informatics, Motion Imagery, and Network Analytics VIII, 106450I (27 April 2018)

[40] A. Chowdhery and M. Chiang, "Model Predictive Compression for Drone Video Analytics," *2018 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, Hong Kong, 2018, pp. 1-5.

[41] E. Molina and Z. Zhu, "Persistent Aerial Video Registration and Fast Multi-View Mosaicing," in *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2184-2192, May 2014.

[42] Botterill, T.; Mills, S.; Green, R., "Real-time aerial image mosaicing," *Image and Vision Computing New Zealand (IVCNZ), 2010 25th International Conference of* , vol., no., pp.1,8, 8-9 Nov. 2010.

[43] Breszcz, M., Breckon, T.P. & Cowling, I. Real-time Mosaicing from Unconstrained Video Imagery for UAV Applications. Proc. 26th International Conference on Unmanned Air Vehicle Systems (2011).

[44] Guangyuan Zhang; Zhengfang Zhu; Guannan Si; Xiaolin Wei, "Mosaic method based on feature points detection and tracking for unmanned aerial vehicle videos," in *Natural Computation (ICNC), 2014 10th International Conference on*, vol., no., pp.948-952, 19-21 Aug. 2014

[45] Yang, Yang, et al. "A real time mosaic method for remote sensing video images from UAV." *Journal of Signal and Information Processing* 4.3B (2013): 168.

[46] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, June 1981.

[47] Torr, P. H. S., and A. Zisserman, "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry," *Computer Vision and Image Understanding*, 2000.

[48] Harris, C., and M. J. Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*. August 1988, pp. 147–152.

[49] David G. Lowe., "Distinctive Image Features from Scale-Invariant Keypoints. " *Int. J. Computer. Vision* 60, 2 (November 2004)

[50] Bay, H., A. Ess, T. Tuytelaars, and L. Van Gool. "SURF:Speeded Up Robust Features." *Computer Vision and Image Understanding (CVIU)*. Vol. 110, No. 3, 2008, pp. 346–359.

[51] R.J. Radke. Computer Vision for Visual Effects. Cambridge University Press, NY, USA, 2000. ISBN 0521623049

[52] Brown, M., Szeliski, R., and Winder, S. (2005). Multi-image matching using multi-scale oriented patches. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005), pp. 510–517, San Diego, CA.

[53] R.Szeliski. Computer Vision: Algorithms and Applications. Springer Science & Business Media, 2010

[54] Ke, Yan, and Rahul Sukthankar. "PCA-SIFT: A more distinctive representation for local image descriptors." *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 2. IEEE, 2004.

[55] Mikolajczyk, Krystian, and Cordelia Schmid. "A performance evaluation of local descriptors." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27.10 (2005): 1615-1630.

[56] Brown, Matthew, and David G. Lowe. "Invariant Features from Interest Point Groups." *BMVC*. Vol. 4. 2002.

[57] Brown, M., Szeliski, R., and Winder, S., "Multi-image matching using multi-scale oriented patches," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 510–517, San Diego, CA.

[58] Beis, Jeffrey S., and David G. Lowe. "Indexing without invariants in 3d object recognition." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 21.10 (1999): 1000-1015.

[59] Muja, Marius, and David G. Lowe. "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration." *VISAPP (1)* 2 (2009).

[60] Nist´er, D., Stew´enius, H.: Scalable recognition with a vocabulary tree. In: CVPR (2006)

[61] Guangyuan Zhang; Zhengfang Zhu; Guannan Si; Xiaolin Wei, "Mosaic method based on feature points detection and tracking for unmanned aerial vehicle videos," in *Natural Computation (ICNC), 2014 10th International Conference on*, vol., no., pp.948-952, 19-21 Aug. 2014

[62] Lucas, Bruce D. and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision,"*Proceedings of the 7th International Joint Conference on Artificial Intelligence*, April 1981, pp. 674–679.

[63] Caballero, Fernando, et al. "Improving vision-based planar motion estimation for unmanned aerial vehicles through online mosaicing." Robotics and Automation, 2006. ICRA 2006.

[64] Li, Chaokui, et al. "The UAV Video Image Stitching Based on Improved Moravec Corner Matching Method." *International Journal of Remote Sensing Applications* 2 (2012).

[65] R. Viguier *et al*., "Automatic Video Content Summarization Using Geospatial Mosaics of Aerial Imagery," *2015 IEEE International Symposium on Multimedia (ISM)*, Miami, FL, 2015, pp. 249-253.

[66] Runaba Aktar, et al. "Geospatial content summarization of UAV aerial imagery using mosaicking." Proc. SPIE 10645, Geospatial Informatics, Motion Imagery, and Network Analytics VIII, 106450I (27 Aprial 2018)

[67] Rosten, E., and T. Drummond, "Machine Learning for High-Speed Corner Detection." *9th European Conference on Computer Vision*. Vol. 1, 2006, pp. 430–443.

[68] Z. Zhang, "A flexible new technique for camera calibration," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330-1334, Nov. 2000

[69] D. Avola, L. Cinque, G. L. Foresti, N. Martinel, D. Pannone and C. Piciarelli, "A UAV Video Dataset for Mosaicking and Change Detection from Low-Altitude Flights," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2018).

[70] Irani, Michal, and Prabu Anandan. "Video indexing based on mosaic representations." *Proceedings of the IEEE* 86.5 (1998): 905-921.