AirMatch: An Automated Mosaicing System with Video Preprocessing Engine for Multiple Aerial Feeds*,**

Nida RASHEED(a), Student Member, Waqar S. QURESHI(b), Member, Shoab A. KHAN(c), Manshoor A. NAQVI(d), and Eisa ALANAZI(e), Nonmembers

SUMMARY Surveillance through aerial systems is in place for years. Such systems are expensive, and a large fleet is in operation around the world without upgrades. These systems have low resolution and multiple analog cameras on-board, with Digital Video Recorders (DVRs) at the control station. Generated digital videos have multi-scenes from multi-feeds embedded in a single video stream and lack video stabilization. Replacing on-board analog cameras with the latest digital counterparts requires huge investment. These videos require stabilization and other automated video analysis prepossessing steps before passing it to the mosaicing algorithm. Available mosaicing software are not tailored to segregate feeds from different cameras and scenes, automate image enhancements, and stabilize before mosaicing (image stitching). We present "AirMatch", a new automated system that first separates camera feeds and scenes, then stabilize and enhance the video feed of each camera; generates a mosaic of each scene of every feed and produce a super quality mosaic by stitching mosaics of all feeds. In our proposed solution, state-of-the-art video analytics techniques are tailored to work on videos from vintage cameras in aerial applications. Our new framework is independent of specialized hardware requirements and generates effective mosaics. Affine motion transform with smoothing Gaussian filter is selected for the stabilization of videos. A histogram-based method is performed for scene change detection and image contrast enhancement. Oriented FAST and rotated BRIEF (ORB) is selected for feature detection and descriptors in video stitching. Several experiments on a number of video streams are performed and the analysis shows that our system can efficiently generate mosaics of videos with high distortion and artifacts, compared with other commercially available mosaicing software.

key words: legacy systems, remote sensing, video mosaicing, unmanned aerial vehicles, video stabilization

1. Introduction

Aerial surveillance through aerial imagery is in place for many years. Aerial videos are different from ground videos in terms of basic characteristics due to high altitude, fast speed, and low visibility. Applications of aerial imagery are helpful in amateur shootings, real-time situations like disasters, climate change, surveillance, object detection, and military applications [1]. Multiple images from the same video file are stitched together to have a larger field of view (FOV), which is referred to as mosaicing [2], [3]. The larger FOV helps in applications related to target-based surveillance, disaster management, photography, scene change, change detection in an area, 3D mapping, and military surveillance applications [4]. Mosaicing is a key operation for geographical analysis systems such as ENVI [5], ArcGIS [6], Virtual Mosaic [7], TNTmips [8].

A large base of legacy aerial system is in operation for decades. These systems are equipped with on-board analog cameras with digital video recorder at the control stations. These functional units cannot be replaced with digital counterparts due to compatibility issues, and due to the excessive cost of new sensors and cameras. The unmanned aerial vehicles (UAV) and the on-board analog cameras considered for this research are more than 12 years old. Modern UAVs are equipped with inertial measurement unit (IMU) sensors along with a gimbal for video stabilization during in-flight vibrations. The aerial platforms under research have no such sensors installed for video stabilization during the flight. The platforms are equipped with multiple analog cameras installed at different view angles; with all of them recording simultaneously. Video feeds generated from cameras with multiple scenes are multiplexed into a single video stream and transmitted to the ground station, where it is digitized through a digital video recorder and stored in a hard-drive at the ground station. This is shown graphically in Fig. 1.

To extract useful information from these videos manually,
one has to separate each video feed, align them together, and watch each video file for any object of interest or change detection. The scene change detection due to change in zoom or change in orientation of the camera makes it difficult to keep track of the location being monitored. Instead, a mosaic generated from each video gives a better view of the scene/object of interest and the location. Videos obtained from these legacy platforms require an automated system that has various video pre-processing components for scene separation, contrast enhancements, and video stabilization before generating a mosaic, which would be impossible otherwise due to the low-quality of these videos.

Limited work is found on mosaicing of videos from vintage aerial systems in real-time in literature. Mosaicing of videos from such systems needs the support of other image processing and machine learning operations. A histogram-based correlation method between two consecutive frames is used to detect scene change. In [10], stability in video is achieved by introducing optimal camera paths for salient features with Linear programming framework. In [11], the issue of stabilization for aerial video surveillance is addressed using SIFT point matching, affine transformations, and Kalman and Median filters for VIRAT aerial dataset [12]. In [13], Contrast Limited Adaptive Histogram Equalization (CLAHE) is used for images. In [14], CLAHE is used for medical image enhancement, in combination with the wavelet-based fusion method. In [15], CLAHE is used for real-time videos with fog for improving visibility. CLAHE has proved to work for both colored and gray-scale inputs, as well as, homogeneous and heterogeneous fog.

One of the earlier attempts for real-time panoramic videos from freely moving cameras is made in [16] in which temporal information from different angles and different devices is used to reduce execution time with the same scenes. Local motion information used for stitching images is used for stabilization in [17]. The deblurring algorithm is applied for compensation of camera motion while spatial smoothness is achieved by optical flow extrapolation. In [18], a solution for stitching panorama for ground videos is presented; using color correction, Kalman filter, and SIFT. In [19], a unified stitching and stabilization model for mobile phone videos is proposed, along with pre-processing and post-processing units. This system is however not real-time and cannot run on a smartphone. The autonomous system for diverse kinds of videos available in the VIRAT data set in [12] is presented in [20]. A hybrid, graph-based stitching model is designed that takes advantage of the continuity in time-domain and repeatability in the spatial domain for aerial videos with artifacts and un-stability. However, artifacts may appear for oblique images. In [21], KLT-based optical flow analysis and scale-invariant distance-vector are used for the generation of mosaic from ottoman documents. The developed system is not a real-time system and performs offline. A modified algorithm for real-time video stabilization and mosaicing is proposed in [22]. Video motion is represented using an inter-frame Homography model based on an optical flow tracker. Kalman filter is employed for smoothing operation. In [23], image operations are applied in the spatial domain, with an iterated projection in the background for Super-Resolution (SR) mosaic construction of low-resolution frames from UAV. In this paper, SIFT is used for computing initial SR mosaic, and three different methods namely: Steepest Descent method (fastest), Conjugate Gradient Method (visually best), and Levenberg-Marquardt algorithms are applied for non-linear optimization. In [24], a spatial and temporal coherent filter for outlier removal, and dynamic key-frame based stitching method is used to reduce the accumulation error for online aerial data. In [25], optical flow is used as a base method to reduce errors for seam-line searching which is generated during mosaicing of UAV videos and produces ghosting effects. In [26], optical flow is used in conjunction with Gaussian Mixture Model for trajectory estimation in abnormal behaviour detection during surveillance. In [27], real-time stitching model is presented to stitch HD videos using SIFT and modified Homography update based on stitching quality evaluation, to remove the artifacts caused by vibration. For offline video stitching, the issue of parallax effects is tackled in [28]. The authors have used Lookup Table (LUT) and the histogram of Gaussian is applied to the points having parallax. Commercially available software systems for generating mosaics for aerial imagery includes AutoStitch [29], AutoPano [30], PTGui [31], Clevr [32], Hugin [33]. However, these software’s lack the automated video pre-processing components required for our video feed (obtained from legacy aerial systems) and produces low-quality mosaics or no mosaics at all. The video pre-processing should include, individual feed separation, scene change detection, image enhancements, and video stabilization.

In this paper, we present “AirMatch”, a new automated system that separates camera feeds and scenes, then for each scene, stabilize and enhance every video feed and generates a mosaic of every feed. Then, it produces a super quality mosaic by stitching mosaics of all feeds for each scene. Our contribution lies in the software framework combined with video preprocessing engine to automate the mosaic generation for the video feeds of these legacy video acquisition systems. We use state-of-the-art algorithms; customized to address the complex nature of the challenge at each video prepossessing stage before the mosaic generation. The rest of the paper is divided into three sections. Section 2 is about system design. Section 3 is about experimental setup and results. Section 4 concludes the research work.

2. System Design

We present a graphical representation of the video acquisition system in Fig. 1. There are 7 cameras installed on the UAV and feeds from all the cameras are multiplexed as one; saved in a single video file at the ground station. Figure 2, shows the basic framework of our implementation. First, ‘M’ number of feeds are separated at the ground station and then ‘N’ number of scenes are separated. For each scene, there are ‘M’ number of feeds. Each feed passes through
Stabilization and enhancement units and a mosaic is generated. In the last phase, all mosaics from the feeds of a single scene are combined to generate a super mosaic. Every module from Fig. 2 is individually explained in this section. A GUI is developed as a part of our software system to make it user-friendly. This system is designed with a complete focus on mosaicing of videos from vintage systems and associated video pre-processing. The broader aim is the creation of a large repository of mosaics, tagged for object recognition and classification. The developed automated system is independent of any requirement for specialized hardware or high processing expensive units. All the modules of our system work in real or near-real time with the minimum response time.

2.1 Feed Separation

Feed separation is the first step in the automated video pre-processing stage. There are in total seven cameras mounted on the aerial platform which are recording simultaneously. All seven on-board camera feeds have a frame rate of 24 frames per second. To stream the feeds to the control station as one multiplexed video feed (at 30 frames/sec), one of the feeds (known as main feed) is transmitted with a frame rate of 24 frames/sec while the others at a rate of 1 frame/sec. The user at the control station can select the “main feed” for manual monitoring. The multiplexed video feed is then given to a digital video recorder and is saved in a hard-disk for offline monitoring and further processing. Our feed separation algorithm de-multiplexes the feed of each camera at the same rate and saves them as separate feeds in a hard-drive. The time complexity of this module is $O(n)$ where $n$ is the number of frames in the input video.

![Fig. 2](image-url)  
**Fig. 2** System diagram at ground station with feed separation, scene separation, stabilization, enhancement, and mosaic generation units

2.2 Scene Separation

After feed separation, the next step in the preprocessing is the scene separation based on the histogram-based shot detection method. Each video feed separated may have multiple scenes as the user may have switched the main feed multiple times during the flight. It is therefore necessary to separate multiple scenes from each feed before the mosaic generation. It also helps in automated indexing within the video and content-based video retrieval. Our scene separation algorithm is based on [9]. Alg. 1 presents a detailed algorithm for our scene separation module for each video feed.

**Algorithm 1: Algorithm for scene separation**

| Input | Separated Video feeds |
|-------|-----------------------|
| Output: Scenes as scene1, scene2, scene3, ... sceneK (K=total scenes detected) |

1. Read video frame by frame as frame0, frame1, ..., frameL. (L=Total frames)
2. Initialize arguments to calculate histogram
   - No. of bins = $N$ (256 for gray scale image)
   - Channels [(0,1)]
3. For each frame 0, 1, 2, ..., L
   - Convert RGB frame to HSV frame
   - Calculate H-S histogram
4. For two consecutive frames (0,1), (1,2), (2,3), ..., (L-1, L)
   - Compare histogram using co-relation method
     \[
     d(H_1, H_2) = \frac{\sum_k (H_1(I) - \overline{H_1})(H_2(I) - \overline{H_2})}{\sqrt{\sum_k (H_1(I) - \overline{H_1})^2 \sum_k (H_2(I) - \overline{H_2})^2}}
     \]
     where $N$ is total bins, $k=1,2$ and $\overline{H_k} = \frac{1}{N} \sum H_k(I)$
   - If compared result value < threshold then scene change is detected

The correlation is used for the comparison of histograms, with threshold values set heuristically to 0.73. A scene is considered different if the values fall below the specified threshold. After scenes are separated as video files, they are stored in the hard disk ready for the stabilization step. Our scene separation module has an absolute success rate for detecting and separating scenes. The time complexity of this module is $O(n)$ where $n$ is the number of pixels of the input frame.
2.3 Video Stabilization

Video stabilization is the next module of our system. Aerial videos of legacy UAVs are inherently unstable due to different factors during the flight, such as a change in camera tilt, mechanical vibrations, and air turbulence. This is the main reason that modern UAVs are equipped with gimbal onboard for stabilization during flight while the aerial systems under research lack such equipment. Video stabilization is a key pre-processing step before the mosaic generation.

**Algorithm 2: Algorithm for stabilization**

**Input**: Unstable Video  
**Output**: Stabilized and Deblurred Video  
1. Read video frame by frame from frame0 to frameL where L = Total frames  
2. For each frame 0, 1, 2, …, L, find key-point features using ORB.  
3. For every two consecutive frames with selected features (0,1), (1,2), (2,3),…(L-1, L)  
   - Convert each frame to gray-scale  
   - Estimate global motion  
     - Use Affine motion transform between two consecutive image pairs  
     - Removal of blurry pixels  
       a. Estimation of removal of high frequency component  
       b. Replacement of the blurry pixels with weighted interpolation of neighboring sharper pixels  
   - Estimate local motion between frames  
     - Compute KLT based corner features  
     - Compute optical-flow based velocities in x, y, and orientation within frames for KLT corner features  
   - Perform image smoothing Use Gaussian Mixture  
     - Use Gaussian Mixture Model for trajectory smoothing  
     - Compute final transformation using the difference between the smoothed trajectory and the original trajectory  
     - Add the final transform to the original transform  
   - Transform the video  
     - Warp the original RGB frame using Affine motion transform obtained in the previous step  
     - Fix the artifacts produced by rotation and translation using scaling and pixel replication for borders  
     - Resize if the sizes do not match  
4. Write all the stabilized RGB frames to video

Alg. 2 shows our algorithm for stabilization which is based on [10], [17] and [37]. Our algorithm incorporates the effects due to the fast motion of aerial videos in real-time. We have used ORB presented in [34] for feature detection to address the issue of rotation and accelerate the process. A two-step process from [17] is used to address the issue of motion blur based on the global motion pair-wise estimation. The first step is the evaluation of the relative blurriness by estimating the high-frequency component removed during pair-wise comparison by using inverse of sum of squared gradient measure. Next, blurry pixels are replaced with the weighted interpolation of neighboring sharper pixels. Optical flow is used for feature trajectory generation. Affine motion transform is used in conjunction with the smoothing Gaussian mixture model for shifting, rotation, and noise reduction. This algorithm works for each $M^{th}$ feed of $N^{th}$ scene. The time complexity of this module is calculated as $O(n)$ by using selective features. Here $n$ is the number of pixels (width x height) of input frame.

2.4 Video Enhancement

The enhancement of video helps in distinct object recognition and making the video crisp. Video enhancement is the next video preprocessing module to address this issue in our system. Simple Histogram equalization failed for the videos due to the global equalization effect, where the contrast was too high in certain test experiments. We used Contrast Limited Adaptive Histogram Equalization (CLAHE) since it is adaptive, responds to varying illuminations, and works best with diverging parameters for different video data-sets. Alg. 3 is the algorithm for the enhancement module and is based on [14]. The clipping ratio is set as 3. The image is first converted to 3x3 smaller images corresponding to 9 tiles and then histogram equalization is performed on each to get the result. The time complexity of this module is calculated as $O(n)$ where $n$ is the number of pixels of the input frame.

**Algorithm 3: Algorithm for video enhancement**

**Input**: Single video file  
**Output**: Enhanced video file  
1. Read video frame by frame as frame0, frame1,.., frameL (L = Total frames)  
2. For every frame 0, 1, 2, …., L  
   - Convert RGB image to Lab format  
   - Split image into Lab Format planes  
   - Set tile size and clipping ratio  
   - Convert original image into smaller images in defined tile sizes,  
     - Apply histogram equalization on the plane [0] (first plane) of these smaller images as this plane holds illumination information only  
   - Convert resulting equalized image back to RGB  
3. Write all enhanced frames to video

2.5 Mosaic Generation

The final stage of our system is high-quality mosaics for each scene and then combining the mosaics to generate super mosaic. An ideal mosaic should have no visible seams with maximum similarity to the input video frames. For low-resolution videos, the generation of the single super-resolution mosaic of a scene is a challenging task. To enhance the quality of the mosaic, a mosaic is generated for each camera feed of a scene and then all mosaics are stitched together to generate a high-quality mosaic. Alg. 4 shows our algorithm for mosaicing that is based on the implementation of Brown et al.[29]. For every $N^{th}$ scene, Alg. 4 is applied
for the mosaic generation of every $M$th feed and $M$ number of mosaics are generated for a scene. These $M$ mosaics are eventually passed through the same Alg. 4. This ensures the generation of a superior quality mosaic for every $N$th scene, achieved by stitching all mosaics from different angle cameras. For feature matching, we choose ORB features with ORB (oriented BRIEF) descriptors [34] heuristically, and found them performing better on our training videos. The time complexity of this module is $O(n)$. Here $n$ is the number of pixels (width x height) of input frame.

**Algorithm 4:** Algorithm for mosaicing

| Input | Video file having one feed from one scene |
|---|---|
| Output | Single mosaic |

1. Convert video to frames, frame 0 to frame $N$ ($N=$Total Number of frames)
2. For all selected frames:
   - Resize all frames at the same scale
   - Detect key-points using ORB
   - Match features using AFFINE
   - Compute descriptors using ORB
   - Estimate Homography using RANSAC
   - Warp images using affine transformation
     - Compensate exposure errors
     - Detect seams and resize masks to the original resolution before resizing
   - Blend to obtain final mosaic

3. Experiments and Results

In this section, we present details of the data set used, experimental setup, the evaluation parameters, results of all the modules individually, and quantitative and qualitative results.

3.1 Implementation Details

We implement our automated system in C++. We use open-source computer vision library ’openCV’ [35] for video I/O and module implementation. The experiments are done on a laptop computer having Intel Core i5-8250U CPU (8th Generation) with 8GB memory.

3.2 Data Set and Characteristics

Data set of aerial videos of actual flights are provided by the users (both day and night) with multiple feeds and multiple scenes embedded in a single video file for each flight. These videos have variable time lengths, different resolutions, different dynamic contrasts, and are unstable. The videos cannot be processed using a commercial off-the-shelf mosaic generation software without our automated video pre-processing engine. Video data provided by the user is not publicly available. However, it can be provided on demand. The data set is divided into testing and training data. Variables like the threshold in scene separation are calculated heuristically from this data set, with at least 15 videos used for training and rest for testing. We have also included the result of a synthetically generated video feed obtained from the internet [36] to demonstrate the effectiveness of our software.

3.3 Feed Separation

Figure 3 shows an example of the separated camera feeds, each from a different angle. The larger sized frame shown on the right-side displays main camera frames; while frames are shown on the left-side windows display frames extracted from the other six cameras. The camera feed with part of the UAV visible and the feed with higher tilt is not used for generating mosaic. However, those frames are segregated in the scene separation step. All seven feeds are saved into the hard disk for the next preprocessing step.

3.4 Scene Separation

Figure 4 shows 4 scenes separated from a single video file. This sample video has clips of four different areas, shot at separate times in a single flight.
3.5 Stabilization

Figure 5 shows snapshots of unstable frames from 4 different videos in part (i-iv), with the results of each frame being stabilized and corrected for the mosaic generation in Fig. 5(a-d). Good frame quality is achieved after going through the stabilization module. Stability graphs are added in the results section for the output validation.

3.6 Enhancement

A sample snapshot of a video frame is shown in Fig. 6 to demonstrate the effectiveness of the applied algorithm. It can be seen from Fig. 6 that enhancement application results in better boundaries and segregation of objects. Objects like roads, trees, and buildings can be better recognized after the application of the enhancement model. This helps in-video tagging and object tracking.

3.7 Mosaic Generation

Figure 7 shows three examples of super-mosaics generated from the provided dataset using $M$ individual feed mosaics. Alg. 4 is followed for the generation of super-mosaic as well. Figure 8 shows different super-mosaics. In Fig. 8(a)-8(d), we can see crisp resultant mosaics of daytime videos, while Fig. 8(e) shows the mosaic of video captured at nighttime using an Infrared camera. As we can see that the generated mosaics have clear boundaries of objects, no seam-lines are detected and have the maximum area covered. This is extremely helpful in object detection and change analysis in a sudden event.

3.8 Qualitative Comparison Results

A comparison is shown in Fig. 9 for our software with state-of-the-art systems like AutoPano [30] and PTGui [31]. Both software produce good results on average for the video which is fed after the feed separation, scene separation, stabilization, and image enhancement but needs image frames to be fed into the system with videos of good quality. Both of them are not tailored to work for such complicated video feeds from legacy systems. PTGui performs poorly for aerial videos with parallax effects. Both could not generate mosaics for a few tested videos. As an example, mosaic of [36] could not be generated using AutoPano, shown in Fig. 9(b) as Vid 3.

We have collected ratings for AutoPano, PTGui, and AirMatch from a pool of 50 participants belonging to the organizations who are the actual users and are currently using our implemented system. Users graded mosaics of the same area generated by each software based on the quality parameters derived from the challenges described in Sect. 2.
Fig. 8 Super mosaics of 5 different videos: (a)-(d) are the mosaics of normal day time videos, (e) is generated for the infra-red video feed. All the mosaics have maximum area coverage, no visible seam lines, and excellent object segregation.

Fig. 9 Visual comparison of PTGui, Autopano, and AirMatch. First row: Vid 1, second row: Vid 2, third row: Vid 3. (a) First column from left are the results of PTGui, (b) center column shows the results of AutoPano and (c) the last column shows the results of AirMatch.

which are defined in Table 1 with weights against each and 10 marks are assigned to each quality. In Table 1, the maximum area covered corresponds to the minimum error, with a higher score leading to a low error. It is measured visually by matching the generated mosaic with original video scenes to find similarity. The weights are selected based on the response percentage of users. We carefully picked out the characteristics and asked the users initially to mark the most important ones. Then for each characteristic, we calculated the percentage based on the total number of the participants and added that numerical value to our weight chart. As can be seen from Table 1 that we have the high-
Table 1  Quality assessment grading of the individual characteristics for AutoPano, PTGui and AirMatch

| Serial No. | Quality Characteristic | Weight ($w_i$) | Grade Average ($g_i$) AutoPano | Grade Average ($g_i$) PTGui | Grade Average ($g_i$) AirMatch |
|------------|------------------------|----------------|-------------------------------|----------------------------|-------------------------------|
| 1          | Clarity                | 0.4            | 8.03                          | 8.24                       | 9.31                          |
| 2          | Sharpness              | 0.5            | 7.35                          | 7.92                       | 9.37                          |
| 3          | Seamline               | 0.3            | 7.14                          | 8.25                       | 9.2                           |
| 4          | Parallax               | 0.4            | 6.82                          | 7.24                       | 9.28                          |
| 5          | Complete Area Coverage | 0.5            | 6.93                          | 6.87                       | 9.35                          |

Fig. 10  Video stabilization graphs for videos of Fig. 9 with y-axis showing displacement of pixels within a frame around the respective axis, and x-axis depicting the frames of the video. The top row shows graphs of original videos while the bottom row shows stabilized graphs with a smoother trajectory than the original. The difference in pixels displacement in both trajectories is due to a parameter which controls the radius of the before and after frames for comparison.

Table 2  Time comparison of AirMatch with PTGui and AutoPano for videos of 22 seconds (720×576 resolution)

| Video Name | Total Frames | Time taken for Stabilization | Time taken for Mosaic Generation |
|------------|--------------|------------------------------|----------------------------------|
|            |              | AirMatch | PTGui | AutoPano | AirMatch | PTGui | AutoPano |
| Vid1       | 547          | 35.7     | N/A   | N/A      | 39.2     | 25.5  | 10.7     |
| Vid2       | 550          | 31.4     | N/A   | N/A      | 28.2     | 24.5  | 22.7     |
| Vid3       | 544          | 19.7     | N/A   | N/A      | 17.6     | 17.1  | 13.6     |

The qualitative assessment shows that the developed system performs very well for the videos from vintage aerial systems. The generated mosaics have complete area coverage, with least parallax effects; helpful in object identification and change detection. The presence of all the pre-processing modules helps the user to save time, cost, and energy. This is what makes our system better from the other counterparts.

3.9  Quantitative Comparison Results

This section focuses on the quantitative aspects of AirMatch, in comparison with PTGui and AutoPano. Camera separation and enhancement are real-time with the same fps as of source. Scene separation has an average rate of 15 fps for videos of 720x576 resolution. The stabilization frame rate depends upon the video quality. The shakier a video is the longer time needed to stabilize it. Usually, stabilization takes longer than the mosaic generation. Mosaic generation takes different time on different videos of the same length due to feature variation. As an average, we are getting 15 – 17 fps. Stabilization is the key pre-processing step to generate meaningful mosaics. Figure 10 shows stability graphs using optical flow for videos of Fig. 9, for the validation of our stabilization module using [37]. A comparison of the trajectories of the original videos with the resultant videos shows smoothing with the removal of noise. The difference in the axis in both trajectories is due to a parameter that controls the radius of the before and after frames for comparison. Increasing or decreasing this parameter affects the output quality which in return, affects the scale of the output graphs. For videos that are captured on a more jittery platform, a high value of this parameter is more effective.

Time comparison of videos shown in Fig. 9 is provided in Table 2. These results correspond to the single scene separated from each video with very high-level distortion, zooming effects, occlusions, parallax effects, sudden shifts in camera angle, and unrecognizable objects. The video feed of Vid1 has the most distortion, resulting in the maximum time consumed in stabilization and mosaicing for our average videos. The timing of AirMatch modules is near real-time and comparable with PTGui and AutoPano. Stabiliza-
tion, scene, and feed separation modules are not part of both compared systems. The timing for stabilization is more than the timing of mosaic generation since stabilization involves frame correction and replacement of faulty frames. Among all modules, feature matching between candidate images in mosaicing and stabilization becomes a bottleneck for the overall system. However, our system is near real-time, and results obtained are always within acceptable seconds range; depending upon the length and quality of the input video.

4. Conclusion

In this paper, we present an automated video preprocessing engine for video feeds of legacy UAVs that first separates camera feeds and scenes. Then for each scene, stabilize and enhance every video feed and generates a mosaic of every feed. Eventually, it produces a super quality mosaic by stitching mosaics of all feeds for each scene. Our new framework is independent of specialized hardware requirements and generates effective mosaics. Our contribution lies in the software framework, combined with video preprocessing to automate the mosaic generation for the video feeds of these legacy video acquisition systems. We use state-of-the-art algorithms; customized to address the complex nature of the challenge at each video pre-processing stage before the mosaic generation. The system effectively separates the feeds and scenes, stabilize, and enhance separated scenes and generate a mosaic of every scene separately. We present a qualitative and quantitative comparison with commercially available software like PTGui and AutoPano to support our claim. Scene separation, mosaicing, and stabilization modules are near real-time for our system, while the rest of the modules are real-time. Our future work also includes object detection and classification from generated mosaics. Research presented in this paper is more quality-based, adhering to the main aim of maximum area coverage analysis; with clarity and sharpness added to the generated mosaics.

References

[1] Y. Xu, J. Ou, H. He, X. Zhang, and J. Mills, “Mosaicking of Unmanned Aerial Vehicle Imagery in the Absence of Camera Poses,” Remote Sensing, vol.8, no.3, p.204, March 2016.
[2] T. Lu, S. Li, and W. Fu, “Fusion Based Seamless Mosaic for Remote Sensing Images,” Sens and Imaging, vol.15, no.1, p.101, Nov. 2014.
[3] T. Kekec, A. Yildirim, and M. Unel, “A new approach to real-time mosaicing of aerial images,” Robotics and Autonomous Systems, vol.62, no.12, pp.1755–1767, Dec. 2014.
[4] M. Li, D. Li, B. Guo, L. Li, T. Wu, and W. Zhang, “Automatic Seam-line Detection in UAV Remote Sensing Image Mosaicking by Use of Graph Cuts,” IJGI, vol.7, no.9, p.361, Aug. 2018.
[5] “ENVI - The Leading Geospatial Image Analysis Software,” https://www.harrisgeospatial.com/Software-Technology/ENVI, accessed Feb. 11, 2019.
[6] “ArcGIS,” http://www.arcgis.com/index.html, accessed Feb. 14, 2019.
[7] “Virtual Mosaic,” https://www.textronystems.com/products/virtual-mosaic, accessed Feb. 14, 2019.
[8] “TNTmips Geospatial Software Solution - MicroImages,” https://www.microimages.com/products/tntmips.htm, accessed Feb. 14, 2019.
[9] N.I. Radwan, N.M. Salem, and M.I. El Adawy, “Histogram Correlation for Video Scene Change Detection,” in Advances in Computer Science, Engineering & Applications, vol.166, D. Wyld, J. Zizka, and D. Nagamalai, eds., pp.765–773, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
[10] M. Grundmann, V. Kvatra, and I. Essa, “Auto-directed video stabilization with robust L1 optimal camera paths,” Proc. 2011 IEEE Conf. Comput. Vis. Pattern Recognit., pp.225–232, June 2011.
[11] A. Walla, A. Wall, and A.M. Alimi, “Video Stabilization for Aerial Video Surveillance,” AASRI Procedia, vol.4, pp.72–77, 2013.
[12] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J.T. Lee, S. Mukherjee, J.K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai, “A large-scale benchmark dataset for event recognition in surveillance video,” CVPR 2011, pp.3153–3160, June 2011.
[13] J.B. Zimmerman, S.M. Pizer, E.V. Staal, J.R. Perry, W. McCartney, and B.C. Brenton, “An evaluation of the effectiveness of adaptive histogram equalization for contrast enhancement,” IEEE Trans. Med. Imaging, vol.7, no.4, pp.304–312, Dec. 1988.
[14] B.B. Singh and S. Patel, “Efficient Medical Image Enhancement using CLAHE Enhancement and Wavelet Fusion,” ICA, vol.167, no.5, pp.1–5, June 2017.
[15] G. Yadav, S. Maheshwari, and A. Agarwal, “Contrast limited adaptive histogram equalization based enhancement for real time video systems,” 2014 Int. Conf. Advances in Computing, Communications and Informatics (ICACCI), Delhi, India, pp.2392–2397, Sept. 2014.
[16] M. El-Saban, M. Izz, and A. Kaheel, “Fast stitching of videos captured from freely moving devices by exploiting temporal redundancy,” 2010 IEEE Int. Conf. Image Process., pp.1193–1196, Sept. 2010.
[17] Y. Matsushita, E. Ofek, Weina Ge, X. Tang, and H.-Y. Shum, “Full-frame video stabilization with motion inpainting,” IEEE Trans. Pattern Anal. Mach. Intell., vol.28, no.7, pp.1150–1163, July 2006.
[18] W. Xu and J. Mulligan, Panoramic Video Stitching, 2012.
[19] Y. Nie, T. Su, Z. Zhang, H. Sun, and G. Li, “Dynamic Video Stitching via Shakiness Removing,” IEEE Trans. Image Process., vol.27, no.1, pp.164–178, Jan. 2018.
[20] T. Yang, J. Li, J. Yu, S. Wang, and Y. Zhang, “Diverse Scene Stitching from a Large-Scale Aerial Video Dataset,” Remote Sensing, vol.7, no.6, pp.6932–6949, May 2015.
[21] E. Saykol, “Keyframe-based video mosaicing for historical Ottoman documents,” Turk J Elec Eng and Comp Sci, vol.24, no.5, pp.4254–4266, 2016.
[22] J. Dong and H. Liu, “Video Stabilization for Strict Real-Time Applications,” IEEE Trans. Circuits Syst. Video Technol., vol.27, no.4, pp.716–724, April 2017.
[23] A. Camargo, Q. He, and K. Palaniappan, “Performance Evaluations for Super-Resolution Mosaicing on UAS Surveillance Videos,” Int. J. Advanced Robotic Systems, vol.10, no.5, p.249, May 2013.
[24] J. Li, T. Yang, J. Yu, Z. Lu, P. Lu, X. Jia, and W. Chen, “Fast Aerial Video Stitching,” Int. J. Advanced Robotic Systems, vol.11, no.10, p.167, Oct. 2014.
[25] W. Zhang, B. Guo, M. Li, X. Liao, and W. Li, “Improved Seam-Line Searching Algorithm for UAV Image Mosaic with Optical Flow,” Sensors, vol.18, no.4, p.1214, April 2018.
[26] N. Rasheed, S.A. Khan, and A. Khalid, “Tracking and Abnormal Behavior Detection in Video Surveillance Using Optical Flow and Neural Networks,” 2014 28th Int. Conf. Advanced Information Networking and Applications Workshops, BC, Canada, pp.61–66, May 2014.
[27] Y. Lu, B. Liu, W. Li, and N. Yu, “Fast Video Stitching for Aerially Captured HD Videos,” Int. J. Image Grap., vol.18, no.03, p.1850018, July 2018.
[28] M.U. Kakli, Y. Cho, and J. Seo, “Minimization of Parallax Artifacts
in Video Stitching for Moving Foregrounds,” IEEE Access, vol.6, pp.57763–57777, 2018.
[30] A. Jenny, “Autopano,” kolor, https://autopano.kolor.com/, accessed Jan. 26, 2019.
[31] “PTGui,” New House Internet Services BV, 2001, https://www.ptgui.com/, accessed Jan. 27, 2019.
[32] Clementine, CleVR Ltd., www.clevr.com, accessed Jan. 25, 2019.
[33] P. d’Angelo, http://hugin.sourceforge.net/, accessed Feb. 29, 2019.
[34] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” 2011 Int. Conf. Comput. Vis., pp.2564–2571, Nov. 2011.
[35] itseez, “Open Source Computer Vision Library,” https://github.com/opencv
[36] “Niagara Falls Helicopter Flight,” Hoosier Tim’s Travel Videos, https://www.youtube.com/watch?v=J05B49CO4al, June 2012.
[37] N. Ho, “Simple video stabilization using OpenCV,” SIMPLE VIDEO STABILIZATION USING OPENCV, 20-Feb-2014.

Nida Rasheed has received her M.S. degree in Computer Engineering from the College of Electrical and Mechanical Engineering, National University of Sciences and Technology in 2013 and currently she is a Ph.D. Scholar with same institute. She received her B.S. degree in Computer Engineering from the University of Engineering and Technology, Taxila in 2009. Her research interests include video and image processing, machine learning and aerial surveillance systems. She has more than 10 years of experience in teaching and research at COMSATS University Islamabad.

Wagar S. Qureshi is a computer vision consultant and an electrical engineer with over 16 years of experience, in industry and academia. He completed his Doctor of Engineering in 2015 with focus on computer vision applied to agriculture and surveillance. His expertise includes mostly in applied research in computer vision, image processing, and machine learning for robotics, precision agriculture, 3D scanning and video surveillance applications using different imaging sensors. He is a co-inventor of a 3D scanning mechanism being used in da Vinci Color that is the world’s first FFF full-color 3D printer by XYZPrinting Inc. He is working as a faculty member at the Department of Mechatronics Engineering, NUST College of Electrical and Mechanical Engineering. He is Co-PI at Robots Design and Development Lab, National Center for Robotics and Automation. He won HEC-technology development fund grant that evolved to a startup “Pak Zar Zameen”, which provides precision agriculture services and incubated at Telenor Velocity program and hosted at NUST Incubation center. He published 5 journal publication 9 conference papers, 1 US patent granted (the same patent published at China and Europe), and one local design patent.

Syed Manshoor Ali Naqvi is pursuing MS in Computer Science from School of Electrical Engineering and Computer Science (SEEECS), National University of Science and Technology (NUST) Pakistan. He received his B.E. degree in computer engineering from College of Electrical and Mechanical Engineering (CEME), NUST in 2017. He is a software developer at Pak Zar Zameen start-up based on precision agriculture services at Innovation Centre of CEME, NUST. He worked for Center for Advanced Research in Engineering (CARE) Islamabad as Research Assistant in 2018. He has worked with National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan on Selfie Drone Project for real-time pose estimation.

Eisa Alanazi received the B.Sc. degree in information systems from King Saud University, in 2007, and the M.Sc. and Ph.D. degrees from the University of Regina, Canada, in 2011 and 2017, respectively. He is currently an Assistant Professor with the Department of Computer Science, College of Computers and Information Systems, Umm Al-Qura University, Saudi Arabia. His research interest includes preference learning and reasoning.