View Extension for Teleoperated MAV

Ren Foo Lim, Akihiko Torii, Masatoshi Okutomi
Tokyo Institute of Technology
Tokyo, Japan
{rfoo@ok., torii@, mxo@} ctrl.titech.ac.jp

Abstract

In this paper, we propose a method to extend the field of view (FoV) of cameras mounted on Micro Aerial Vehicles (MAVs). The idea is to stitch together appropriate sections of the panorama to the camera frame. The proposed system efficiently performs view extension by fusing fast tracking and feature descriptor matching into the stitching algorithm. The quality of the extended view is further improved by refining the transformation using weighted least squares. We demonstrate the success of our system in real cases where the distance from one panorama to the next panorama is 10m.

1 Introduction

Vision based MAV systems are getting popular[3, 6, 9, 10] because cameras are lightweight and can capture large amounts of information. This is important since current MAVs have limited flight time and load capacity. The information provided by cameras can now be used to replace GPS sensors in localisation [4] and replace laser scanners in capturing 3D information[14]. The greatest advantage of cameras is that they allow the operator to explore the environment without actually being there. Unfortunately, teleoperating MAVs is difficult because the camera has a narrow FoV. This forces the operator to remember the environment so that they can avoid obstacles.

Common methods to increase FoV include: adding more cameras and using large FoV lenses or mirrors. Adding more cameras will increase the weight and communication load while large FoV lenses and mirrors reduces image quality. In this paper, we will solve this problem through software implementation. The basic idea is to use panoramas to increase the horizontal FoV to 180° while maintaining the information contained in the camera frame, as shown in Figure 1. The required panorama can be obtained from databases such as Google Street View. This way, the MAV will only need to have one camera on board.

The idea of using panoramas to extend FoV has been explored previously by Zhang et al.[15] where larger FoV was achieved by repeating sections of the input frame while using the panorama as a guide. Prior to that is the work by Humphrey et al.[5] which combines two perspective images simultaneously taken from one large FoV camera and another narrow FoV camera to achieve the same effect. Zhang's work cannot be used when there are no repeated structures in the environment while Humphrey's work cannot provide 180° FoV.

Our system makes two contributions by improving on the basic image stitching algorithm that is described in Szeliski's book[12]:

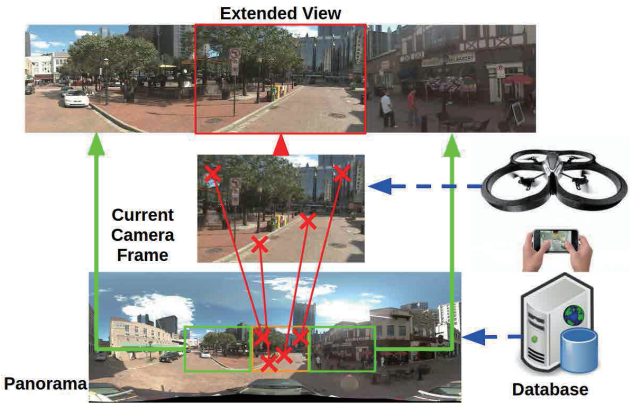

Figure 1. An outline of the proposed system; showing how images from a MAV's camera (red box) and panorama from a database are used to increase the FoV. Red crosses mark the matched features between the panorama and the camera frame. (This Figure is best viewed in colour.)

- Improve efficiency by reducing the amount of computation performed (Section 3).
- Produce visually good extended view when viewed as a continuous image sequence (Section 4).

2 Algorithm Overview

At the core of the system is the image stitching algorithm as detailed in Szeliski's book[12]. A flowchart showing the overview of the algorithm is shown in Figure 2.

The inputs for our algorithm are (A) a panorama from the database and a camera frame from the MAV. We first search for the region in the panorama which matches the camera frame. This is done by (C) extracting SURF[1] features, (D) matching them and (E) computing the transformation matrix from the matches. Once we know where the matched region on the panorama is, we can use the surrounding region to extend the FoV. This is done by (F) warping and cropping the panorama such that the horizontal FoV is 180° while the vertical FoV is the same as the camera frame before stitching the panorama to the camera frame. The final image will consist of three sections: the middle section is the camera frame while the left and right sections are obtained from the warped panorama.

Our contribution here is the improvement in the computational efficiency and visual representation. Efficiency is improved by reducing the amount of SURF computation through (B) feature tracking and by reducing the number of features matched in the matching

phase (Section 3). The appearance of the extended view is improved through (E) the refinement of the transformation matrix and the suppression of unstable transformations (Section 4).

3 Improving the Efficiency

We have analysed the matching accuracy and computation time of SIFT[8], SURF[1], ORB[11] and BRISK[7] on pairs of images. From our experiments, we found that SIFT outperforms all the other descriptors in terms of matching accuracy but is the slowest to compute. ORB and BRISK were significantly faster than the rest but had low matching accuracy. This may be due to the large distortion in the panorama. Finally, SURF performed almost as well as SIFT and can be compute faster compared to SIFT. Taking into account both accuracy and computation time, we decided to use SURF.

Choosing efficient descriptors alone will not increase the efficiency of the system. In order to improve efficiency, we exploit the fact that the camera frames are image sequences (live previews) with the same object appearing in multiple frames. Therefore, there is no need to recompute features that have already been computed previously. To achieve this, we use optical flow[2] to track the position of features that have already been detected and mask them in current camera frame (Process B in Figure 2). This way, the system will only compute SURF descriptors on new features (Process C) and the position of existing features are updated without recomputing their descriptors.

To further increase the computational speed, our system performs matching (Process D) using only a subset of the panorama features. This is explained by Figure 3. Features in this subset are determined by first finding the rectangular region of the panorama that matches the previous camera frame (red box). The width and height of this rectangle is then doubled to include nearby features. The features in this expanded rectangle (green box) are the features in the matching subset.

4 Producing Good Extended View

The quality of the extended view depends on the transformation computed (Process E of Figure 2). We compared similarity, affine and homography transformation by computing overlap errors among them. This was done by performing image subtraction between the camera frame and the region of the warped panorama overlapped by the camera frame and calculating the sum of squared difference (SSD) per pixel for a video sequence with 600 frames. The results are shown in Figure 4. Similarity transformation is chosen because it has the least median error. Although homography's SSD per pixel error is comparable to that of similarity, its higher degree of freedom also causes more deformation to the panorama. This can be seen in Figure 5. Notice how affine and homography transformations cause the buildings to be more tilted than similarity transform.

Similarity transform is computed in two steps:

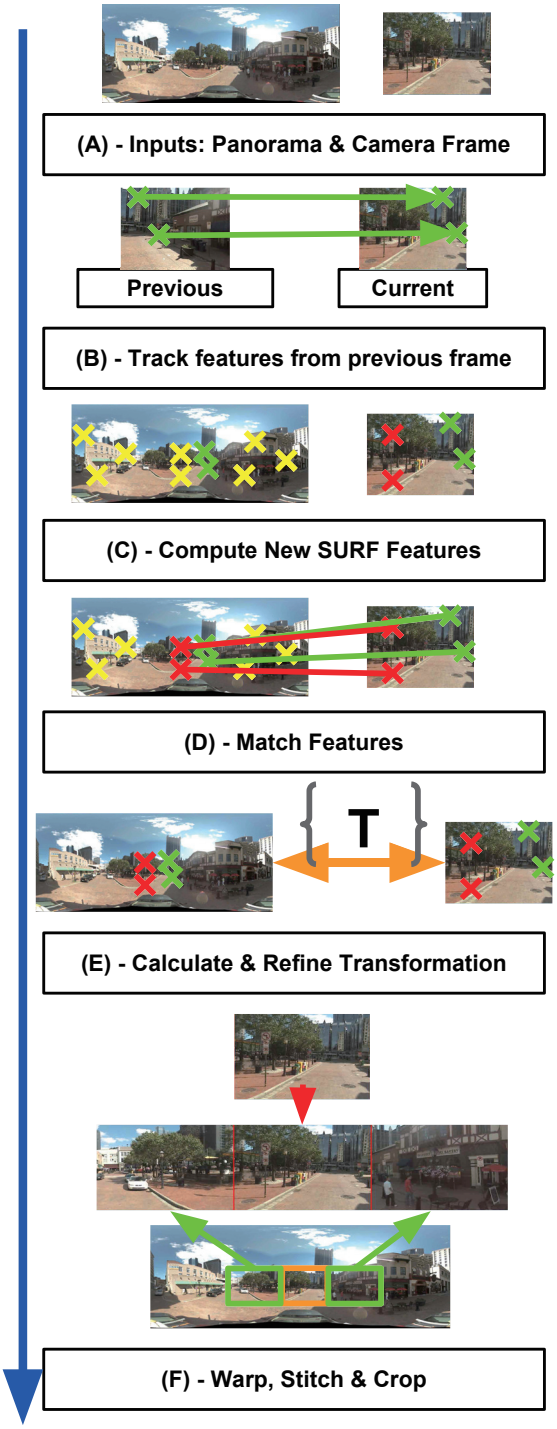

Figure 2. Algorithm Overview. We improve the efficiency and visual representation by extending the basic stitching algorithm (C, D, F). The main modifications are in feature tracking (B) and transformation matrix computation (E). The yellow, green and red crosses mark features detected in the panorama, previous camera frame and current camera frame respectively.

Figure 3. Features located outside the green rectangle on the panorama (**left**) are not used in the matching process. The red rectangle is the region that matched the previous camera frame (**right**)

1. Calculate the best fit transformation to remove outliers

2. Refine transformation to improve appearance

RANSAC can be used to remove outliers but this is not enough because it will produce different transformation matrices due to the randomness in seed generation. This results in large changes (unstable transformations) in the extended view even when the camera is not moving. To alleviate this, we use RANSAC only as an initialisation step to determine the first transformation matrix and inliers. The inliers are then tracked in new camera frames and then used to calculate new transformation matrices instead of using RANSAC. We call this the inlier tracking method. The position of the tracked inliers are obtained from the feature tracking process (Process B in Figure 2). Our system automatically switch back to using RANSAC when there are not enough inliers.

The inlier tracking method consists of three steps: The first step is to estimate the transformation using the inliers that was tracked from previous camera frames. The estimated transformation is then used to find new inliers in the new camera frame. The transformation matrix is then recomputed using both new and old inliers.

Image stitching artifacts sometimes appear in the extended view because the edge of the camera frame does not match to the panorama well enough. To reduce this effect, the transformation matrix is refined by applying exponentially weighted least squares on all the inliers. Inliers near the edge of the camera frame are given more weight so that the panorama sections will join to the camera frame smoothly. By doing so, the seam where the panorama joins to the camera will become less visible and since no modification is made to the camera frame, it still retains all the original information it has before.

5 Experiment Results

We evaluated our system on 2 dataset:

1. **Google Pittsburgh Research Dataset**[1] captured by a panorama camera rig mounted on top of a car that was driven along a stretch of road. The input video was made by creating perspective cut-outs (640 × 480 pixels) from the panoramas (3328 × 1664 pixels).

[1]Provided and copyrighted by Google

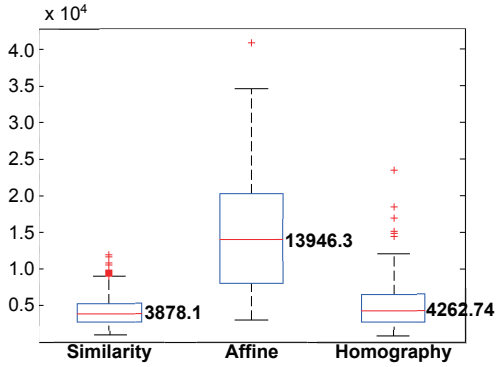

Figure 4. A comparison of the SSD per pixel between Similarity, Affine and Homography transformation. The median error for each transformation is displayed beside each plot.

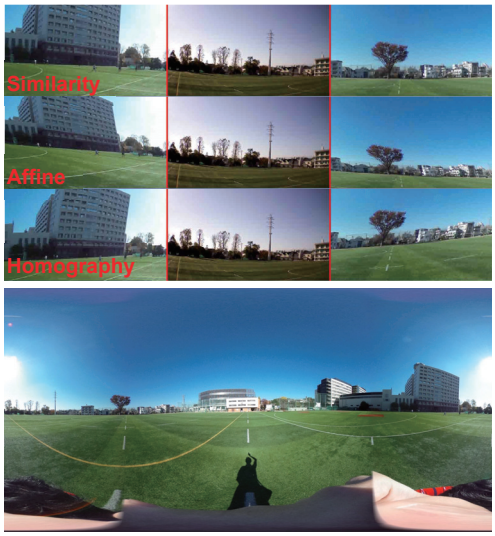

Figure 5. Extended views produced by Similarity, Affine and Homography transformations and the corresponding panorama used.

2. **Real MAV captured video** (640 × 360 pixels) taken by a drone flying in a large field. The required panoramas (3584 × 1792 pixels) were captured using RICOH THETA[2] camera.

We implemented the system in Python. The required image processing functions were obtained from OpenCV. The system can operate between 3 to 5 frames per second on a core-i7 computer with 16Gb RAM. Scenes with more features will require more time for feature computation and matching.

Throughout the experiment, we manually select the panorama closest to the camera frame as the input panorama. This was done to separate the problem of panorama selection from view extension.

Extended views for translational and rotational motion are shown in figures 6 and 7. Our system can produce good extended views even for panoramas that are 5m away from the camera centre (first image in Figure 6), demonstrating that it can be used in real

[2]`https://theta360.com`

Figure 6. View extension results for the Google Pittsburgh Research Dataset. The distance between the position of the panorama camera and MAV camera was varied between 5m(**top**) and 0m(**second from bottom**). **Bottom**: Panorama used. The MAV camera frame is highlighted by the red box.

Figure 7. View extension result for real MAV captured video (first seven images) and the panorama used (Bottom). The MAV camera frame is highlighted by the red box.

Figure 8. Examples of the poor extension caused by features accumulating in the centre of the camera frame. **Bottom Left:** Panorama used. **Bottom Right:** Inliers (green dots). (This Figure is best viewed in colour)

world cases since the distance between the panoramic images in Google Street View is about 10m[13].
Limitations: Poor extended views are produced when there are not enough matched features near the edge of the camera frame. Examples are shown in Figure 8. Note the obvious discontinuity in the region where the panorama connects to the camera frame. Despite this, there is still enough information to aid navigation.

6 Conclusion

We have proposed a view extension system that can extend the horizontal FoV up to 180° without the need for any hardware modification. Our system takes in a panorama and the camera frame as inputs and stitches them together to form the extended view. The system's efficiency was increased by reducing the number of features computed and matched while the appearance of the extended view was improved by refining the transformation matrix and suppressing unstable transformations. Good extended views can be produced even when the distance between the camera and the nearest panorama is 5m, demonstrating that it can be used in real cases.

As future work, we plan to update the panorama in the database using the camera frame so that more recent information is also available in the database.

Acknowledgments: This work was partially supported by the Grants-in-Aid for Scientific Research (no. 25240025) from the Japan Society for the Promotion of Science.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417. Springer, 2006.

[2] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5, 2001.

[3] Christophe De Wagter, Sjoerd Tijmons, Bart DW Remes, and Guido CHE de Croon. Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system. In *ICRA*, pages 4982–4987. IEEE, 2014.

[4] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. In *IROS*, pages 4557–4564. IEEE, 2012.

[5] Curtis M Humphrey, Stephen R Motter, Julie A Adams, and Mark Gonyea. A human eye like perspective for remote vision. In *SMC*, pages 1650–1655. IEEE, 2009.

[6] Alex G Kendall, Nishaad N Salvapantula, and Karl A Stol. On-board object tracking control of a quadcopter with monocular vision. In *ICUAS*, pages 404–411. IEEE, 2014.

[7] Stefan Leutenegger, Margarita Chli, and Roland Yves Siegwart. Brisk: Binary robust invariant scalable keypoints. In *ICCV*, pages 2548–2555. IEEE, 2011.

[8] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[9] Christian Mostegel, Andreas Wendel, and Horst Bischof. Active monocular localization: Towards autonomous monocular exploration for multirotor mavs.

[10] Jesus Pestana, Jose Luis Sanchez-Lopez, Pascual Campoy, and Srikanth Saripalli. Vision based gps-denied object tracking and following for unmanned aerial vehicles. In *SSRR*, pages 1–6. IEEE, 2013.

[11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *ICCV*, pages 2564–2571. IEEE, 2011.

[12] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.

[13] Akihiko Torii, Josef Sivic, Toma Pajdla, and Masatoshi Okutomi. Visual place recognition with repetitive structures. In *CVPR*, pages 883–890. IEEE, 2013.

[14] Andreas Wendel, Michael Maurer, Gottfried Graber, Thomas Pock, and Horst Bischof. Dense reconstruction on-the-fly. In *CVPR*, pages 1450–1457. IEEE, 2012.

[15] Yinda Zhang, Jianxiong Xiao, James Hays, and Ping Tan. Framebreak: dramatic image extrapolation by guided shift-maps. In *CVPR*, pages 1171–1178. IEEE, 2013.