Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2011, Article ID 413074, 10 pages
doi:10.1155/2011/413074

Research Article

Parallel Tracking and Mapping for Controlling VTOL Airframe

Michal Jama[1] and Dale Schinstock[2]

[1] *Department of Electrical and Computer Engineering, Kansas State University, Manhattan, KS 66506, USA*
[2] *Department of Mechanical and Nuclear Engineering, Kansas State University, Manhattan, KS 66506, USA*

Correspondence should be addressed to Michal Jama, mjama@ksu.edu

Received 29 April 2011; Revised 7 September 2011; Accepted 12 September 2011

Academic Editor: Onur Toker

This work presents a vision based system for navigation on a vertical takeoff and landing unmanned aerial vehicle (UAV). This is a monocular vision based, simultaneous localization and mapping (SLAM) system, which measures the position and orientation of the camera and builds a map of the environment using a video stream from a single camera. This is different from past SLAM solutions on UAV which use sensors that measure depth, like LIDAR, stereoscopic cameras or depth cameras. Solution presented in this paper extends and significantly modifies a recent open-source algorithm that solves SLAM problem using approach fundamentally different from a traditional approach. Proposed modifications provide the position measurements necessary for the navigation solution on a UAV. The main contributions of this work include: (1) extension of the map building algorithm to enable it to be used realistically while controlling a UAV and simultaneously building the map; (2) improved performance of the SLAM algorithm for lower camera frame rates; and (3) the first known demonstration of a monocular SLAM algorithm successfully controlling a UAV while simultaneously building the map. This work demonstrates that a fully autonomous UAV that uses monocular vision for navigation is feasible.

1. Introduction

Improved accuracy, robustness, and capability of real-time processing in video-based simultaneous localization and mapping (SLAM) have recently made it a viable tool in a navigation solution for unmanned aerial vehicles (UAVs). However, obtaining an accurate and robust SLAM solution using only video has only recently been solved, and not in a manner suitable for use on UAVs. These recent solutions require significant modification to be used robustly in the navigation of such a demanding application. However, the reward for successful use of video-based SLAM in these applications is high because vision-based navigation can provide a very affordable technology and because vision sensors have advantages over other sensors currently used in successful SLAM solutions for these applications.

Work presented by Klein and Murray gives a novel approach to vision-based SLAM, which they call parallel tracking and mapping (PTAM). In PTAM, bundle adjustment (BA) is used instead of the typical filtering approach [1, 2]. The algorithm's accuracy and the robustness of their design are superb compared to any known real-time SLAM algorithm based on filtering. A key paper by Montiel et al. compares performance of SLAM algorithms based on filtering and SLAM algorithms based on bundle adjustment [3]. Our work presents a modified version Castle's implementation of PTAM [4], parallel tracking and multiple mapping (PTAMM), for use in navigation on a vertical takeoff and landing (VTOL) UAV. PTAMM is available as an open-source library. This library works well for certain situations, nevertheless, without significant modification, it is very limited in its application for UAV navigation.

Very recent successes in using SLAM in real time for navigation on small VTOL airframes have demonstrated its potential. However, nearly all of these successes have used SLAM solutions based on sensors other than pure vision. For example, several researchers have used LIDAR in their solutions including Bachrach and He [5]. Also, even more recently researchers have been successful in using depth cameras which return a dense array of pixels corresponding to the distance to the object seen by the pixel. However, both the depth cameras and the LIDAR which are small enough

to be carried on a small UAV are very limited on range (on the order of 10 m at best). Furthermore, current versions of the depth cameras cannot be used outdoors, and both are significantly more expensive than simple cameras.

There are several examples of researchers working on video-based SLAM for control of UAV. The authors are only aware of one other successful use of monocular video-based SLAM in guidance of a UAV, other than that presented here. This other successful work is presented by Blösch et al. [6], and it shows that a PTAMM solution can be used to guide a UAV. However, in that work, the UAV moves a very limited distance, with almost no rotation, in a scene where the map is generated prior to flight. It does not address the issues of building the map during flight. Also, Nützi et al. [7] present an approach to estimate a scale in PTAM by using data from an inertial measurement unit (IMU). However, this work presents only simulations and does not document any actual flights. Other similar examples include researchers that collect data during flight and postprocess it offline [8–10].

In our work, the PTAMM algorithm is successfully modified to provide the position measurements necessary for the navigation solution on a VTOL UAV while simultaneously building the map. Furthermore, it is used in flights where large maps are constructed in real time as the UAV flies under position control with the only position measurements for the navigation solution coming from PTAMM.

2. Original PTAMM

This section gives an overview of the PTAMM algorithm. It gives a general overview of the entire algorithm; however, it only gives details in the parts that were modified to improve the algorithm's robustness in use for navigation with VTOL aircraft. For a more detailed description of the entire algorithm, please refer to [1, 2, 11].

PTAMM builds a map of the environment by triangulating objects that are observed as matched features in images taken from significantly different vantage points. These images are called keyframes. At the same time it uses this map of 3D features to estimate the current camera pose by finding the locations of these 3D features in the current image. This is the normal PTAMM operation when it is in the tracking state. However, PTAMM has three states (cf. Figure 1):

(1) uninitialized
(2) tracking
(3) recovery.

In the uninitialized state, before the algorithm starts tracking, an initial map of 3D features is created using two frames. The user marks these two video frames, which should be separated by a known distance and should observe the same part of the scene. This known distance sets the scale of the entire map. These frames are stored in the algorithm as the first two keyframes and used throughout the tracking state. After marking a frame as the first keyframe, FAST algorithm [12] is used to detect features in it. Those features are searched for in all the incoming frames. The search uses correlation between patches of images centered around a feature. These patches are used to match it throughout the incoming frames. If the correlation between image patches

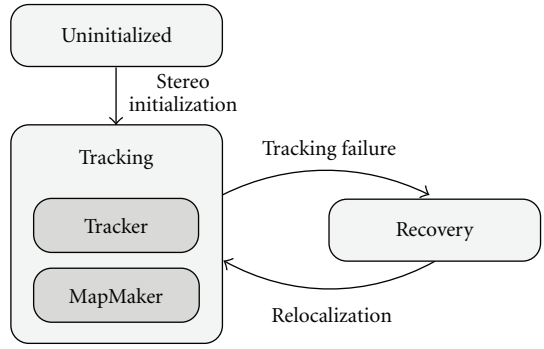

Figure 1: Major parts of PTAMM.

is above a threshold, the feature is declared found in the new frame. Otherwise, it is excluded from further processing. After marking another frame as the second keyframe, all the features that were successfully tracked from the first to the second keyframe are used to determine a homography between the two. The homography is decomposed to find the keyframes' poses by using the Faugeras and Lustman algorithm [13]. The distance between the camera positions is set to the user predefined value. BA is then used to refine the two camera poses and 3D feature locations. An epipolar search is then run to increase the size of map by finding more matches in the FAST features from the two keyframes. The algorithm then moves to the tracking state.

Tracking is performed in parallel in two threads. The first thread, the Tracker, uses information stored in the map to find the camera pose for the current frame. The second thread, the MapMaker, maintains, extends, and improves the accuracy of the information stored in the map. An overview of tasks performed by both threads is shown in Figure 2.

The Tracker uses two stages; a first rough estimate of the camera orientation and a second refined estimate of both orientation and position. In the first stage, the camera orientation is estimated by either a camera motion model or coarse image-based minimization, called small blurry image (SBI). In the default mode, SBI is used and not the motion model. SBI utilizes the Benhimane and Malis algorithm [14, 15] to roughly estimate camera orientation relative to the previous frame. This rough estimate is used as an initial guess in the main tracking algorithm. The main tracking algorithm is based on finding image patches associated with 3D features in the current video frame. For each feature, in the possibly visible set (PVS), a search is done locally around the location predicted by the projection of the 3D feature into the camera frame. This projection uses the initial guess of the current camera orientation computed at the first stage of the Tracker. To reduce the time for the feature search, only locations marked in the current frame as FAST features are considered. To help provide rotational invariance, feature patches are warped accordingly to the camera pose estimate. Once the image templates are matched, a minimization process improves the estimate of the camera pose associated with the current frame. This minimization tries to reduce the difference between actual feature locations and their predicted locations based on the current camera pose estimate.

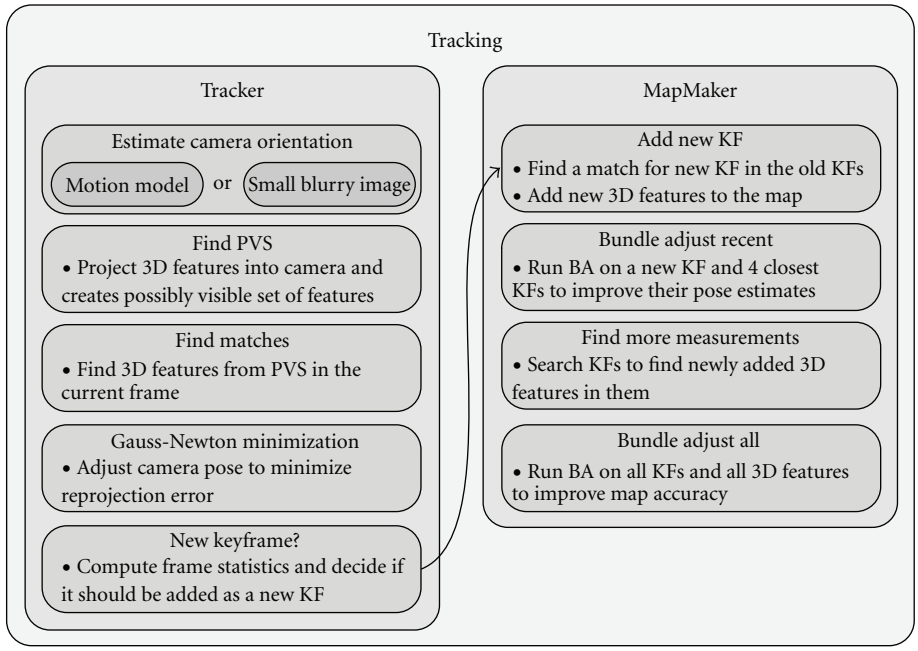

Figure 2: PTAMM tracking overview.

The MapMaker only runs BA when new keyframes are added. Keyframes are added using a normalized measure of distance to nearest keyframe. One key parameter calculated during tracking is the scene depth, which is the average of the distance to the features seen in the current frame. The normalized distance to the nearest keyframe is the distance to this keyframe divided by the scene depth. When the camera is moved to a new location that is further than a predetermined normalized distance from any of the existing keyframes, the new frame is added as a keyframe.

The MapMaker maintains and extends the set of 3D features and the set of keyframes. When a new keyframe is added, the Mapmaker uses an epipolar search between the new keyframe and nearest existing keyframe to find new feature matches, which are triangulated to extend the map. BA is then performed with this new keyframe and and its four nearest neighbors to further refine poses and 3D feature locations. Then, additional measurements of the new 3D features in other keyframes are added by searching for matches in their FAST features. Finally, BA is run with the entire set of keyframes and 3D features. Because this BA is much more computationally intensive than the Tracker algorithms, it is not capable of being run at the typically required frame rates. Therefore, the MapMaker is implemented in a separate thread that runs at much lower frequency than the Tracker.

PTAMM also implements a recovery algorithm. Recovery is triggered when failure of the Tracker is detected. For each frame, the Tracker computes the number of 3D features successfully identified in it. During normal operation, most of the features in the PVS are found. However, significant changes between consecutive frames decrease number of features found because predicted feature locations in the image are inaccurate due to the initial pose estimate being poor. Similarly, image blur decreases number of features found as a result of the features being smeared. Both of these can be caused by fast camera motion, and the significant interframe change can also be caused by limited frame rate. Without a sufficient number of features or with enough false matches, the Tracker minimization fails and the camera pose estimate drifts away from the true value. Since tracking of the current frame uses the pose estimate of the previous frame, significant loss in tracking accuracy starts a chain effect that usually leads to the failure of tracking. Once critical tracking conditions are detected, that is, when the ratio of features found to the number of features in PVS drops below a threshold for a couple of consecutive frames, the algorithm stops relying on the pose information computed at the previous frames and PTAMM enters recovery. The current frame is compared to the stored keyframes, and the best matching keyframe is selected. Comparison uses zero-mean sum of squared differences and works on subscaled versions of the frames. 2D transformation of the selected keyframe into the current frame is computed using Benhimane and Malis algorithm [14, 15]. This computed transformation is used to estimate a 3D camera rotation that would result in the computed image transformation. The Tracker is run with the initial position being equal to selected keyframe position and initial rotation being equal to computed 3D rotation. PTAMM recovers faster when the camera pose is close to one of the stored keyframes poses.

3. Modifications to PTAMM for Map Building in Navigation

In this section, we describe modifications to PTAMM that are made in an attempt to make it more robust in the application

of navigation for VTOL airframes. First, a modification to the initialization part of the algorithm is described. Next, changes to the keyframe addition and matching procedures are outlined.

3.1. SURF for Initialization. During the initialization, PTAMM uses a correlation-based technique to track the features between frames as the camera moves from the first keyframe to the second. This approach is extremely error prone as any nonsmooth camera movement results in a rapid decrease in the number of features being tracked and forces a restart of the procedure. This prevents initialization in cases where the camera needs to travel significant distances in order to obtain reasonable stereo separation or when being transported by an aerial vehicle. Klein and Murray [16] identified this problem and proposed a two stage initialization. In the first stage, tracking uses a single keyframe and operates using a homography. Features do not have 3D information attached, and camera pose is not accurate. Once the camera is moved away from the first keyframe location, and there is enough stereo separation between the views, the algorithm switches into the normal mode. This approach allows initialization even when the camera movement is not smooth; however, it assumes that initial scene is planar which is in general not true. Also, setting up the scene scale is less robust as the algorithm decides internally when to insert the second keyframe. If external devices like GPS or a barometric altimeter are used to set the scale, their measurement may not be accurate or ready at this time.

We have chosen to implement an initialization procedure that uses the SURF [17] feature detector. After capturing the first keyframe, SURF features are localized and the algorithm waits for the user to move the camera and mark another keyframe. No tracking is performed between the two keyframes, which allows the camera to be moved freely. When the user adds another keyframe, SURF features are again found and matching is performed. The SURF implementation in the OpenCV library [18] is used to perform both feature extraction and efficient tree-based matching that does not rely on an initial estimate of pose. The rest of the initialization proceeds as in the original algorithm.

3.2. Fast Map Expansion. The algorithm for expanding the map in PTAMM works well for moving around an object while viewing it from different locations or for moving in the direction the camera is looking. However, it does not work well when new areas are viewed by rotating. The map can not be expanded by simply pointing the camera into unknown part of the scene because a stereo view of any feature is needed to locate it. This poses challenges for exploration as two stereo-separated frames picturing unmapped part of the scene are needed to expand the map with 3D features from this unknown part of the scene. PTAMM tries to expand the map only when a new keyframe is being added. Before adding a frame as a new keyframe, the algorithm is required to determine

(i) whether the current video frame contains new information useful for the algorithm (part of the Tracker),

(ii) and if yes, which old keyframe should be used for matching features with the new keyframe (part of the MapMaker).

To answer the first question, PTAMM uses normalized distance described in the previous section, and, to answer the second, it uses the keyframe with the smallest linear distance from it. In neither case are viewing angles considered. This approach limits the algorithm's ability to explore the environment. Adding new features becomes difficult once an initial set of keyframes is captured because the camera location is likely to be close to at least one of the keyframes already in the set, although the current view may see a significantly different area due to rotation. This prevents fast and reliable exploration because areas without initialized 3D features stay unmapped. Finally, using the closest keyframe for matching limits, the possible stereo separation and the closest keyframe do not necessarily have the largest overlap of viewing area. We propose modifications changing keyframe handling that focus the algorithm on expanding the map. As a result, exploration is made easier.

A modified condition for adding a new keyframe is described here. The algorithm always adds a keyframe if the original, distance-based criterion is fulfilled. If it is not, a new keyframe can be added based on the camera viewing direction. The viewing direction is compared with that of all keyframes within the threshold for normalized distance used to add keyframes. If the angle between current frame's viewing direction and a viewing direction of all keyframes in the described set is above a threshold, we add the current frame as a new keyframe. The difference between viewing directions is just an angle between the two vectors representing them. This modification alone breaks the next part of keyframe addition—matching. The closest keyframe may not have significant stereo seperation and may not have significant overlap. Modification of the keyframe selection for matching is described next.

To ensure valid triangulation, only keyframes having enough separation from the current frame should be considered for matching. To maximize the number of new 3D features added in the unmapped regions, the keyframe selected by the algorithm should have a large overlap with the candidate frame in the region that is missing features.

Two approaches to accomplishing this were implemented and evaluated. In the first one, the closest point of intersection of the camera viewing vectors is found to obtain a 3D point whose distance to the camera locations is compared to scene, depths for the keyframes. The difference between expected point depth and the actual depth is used as a quality measure. A small difference suggests that the camera is looking at a similar area of the scene, and therefore a keyframe with the lowest difference is used for matching. In the second approach, matching is run on scaled versions of the frames. This utilizes FAST features that were already found for tracking of the frame. A keyframe that has the highest number of matches with the candidate frame is selected for a full matching on the full-resolution frames.

Practical evaluation of the two approaches reveals that while the first method gives instantaneous results, the second

method finds the best keyframe to match far more often. The second method was chosen as the default implementation. Note that time requirements of the second method are significantly higher than the first one. However, additional computations do not disturb tracking as keyframe addition is a part of the MapMaker thread and it does not delay the Tracker. Figure 3 shows a new keyframe and the keyframe chosen for matching using the original and new algorithms.

4. Modifications for Handling Low Frame Rate Video

Because our intended application of PTAMM might require use of significantly reduced computational power with an onboard processor, the frame rate becomes an important consideration. In this section, we describe adaptations of the algorithm, aimed at allowing successful tracking even when changes in successive frames are significant. Although our initial implementation for navigation uses the results from offboard processing of the video with PTAMM, we intend to move this to an onboard processor. Reduced processing power results in the decrease of the number of frames that can be processed per second. This in turn leads to more significant differences between consecutive frames used by the Tracker. This phenomenon is further intensified when sudden accelerations are present (e.g., during wind gusts). Large image differences between consecutive frames are the usual cause for the Tracker to fail.

To estimate the camera orientation that is used to seed the Tracker, PTAM uses what it refers to as small blurry images (SBI). The SBI algorithm uses the Benhimane and Malis algorithm [14, 15] which runs on 16 times subscaled and blurred versions of two frames. The Benhimane and Malis algorithm finds a 2D transformation (rotation and translation) converting the previous frame to the current frame. The returned 2D transformation is used in a minimization that finds a 3D rotation. The computed 3D rotation is applied to the pose of the previous frame to compute current frame pose.

PTAMM also includes a motion model for the initial pose estimate to replace the SBI algorithm. In our evaluation, the SBI algorithm was found to behave much more reliably, especially when tracking at higher speeds or when direction of movement was changed rapidly. Also, it is the default mode set by the PTAMM creators. However, when video frame rate is lowered and the camera motion is significant, neither approach seeds the Tracker with a good orientation estimate and the Tracker fails. We describe two approaches that attempt to address this problem.

The first approach uses external measurements from sensors onboard the aircraft which measure angular rate. Most unmanned airframes are equipped with an inertial measurement unit (IMU), including accelerometers and gyroscopes, that are used for flight stabilization. Gyroscopes provide angular rate measurements for rotation along the X, Y, and Z axes of a body-fixed coordinate frame. The output of the gyroscopes are numerically integrated between camera frames using what is commonly known as the the "strapdown equation" in the navigation community. This provides an incremental update to orientation between frames and is used to replace the SBI algorithm.

The second approach extends the SBI algorithm by seeding it with multiple starting points for the minimization. In the original implementation of the SBI algorithm, the Benhimane and Malis algorithm is run once with an initial seed of zero rotation and zero translation. However, due to the low computational cost of this algorithm, it is feasible to run the algorithm multiple times when processing a frame without incurring significant time delay. Therefore, PTAMM was modified to detect failure of the Benhimane and Malis algorithm and turn on an "early recovery mode" where it is seeded with several initial values.

Failure is detected based on the final value of a cost function. The cost function is formulated as a difference between image intensities for the current frame and the previous frame transformed by the 2D transformation under estimation. Final cost is scaled by the inverse of the total number of pixels in the overlap area between the transformed and the candidate frame. If it exceeds a threshold a failure is assumed. The "early recovery mode" runs Benhimane and Malis algorithm starting with 27 different seeds of the 2D transformation. This transformation includes three variables: one rotation and two translations. The 27 different values are derived from all possible combinations of three values for each of the variables. The rotation can take values of $(-45^\circ, 0^\circ, +45^\circ)$, and the translations can take values of $-1/3$ frames, 0 frames, $+1/3$ frames. The solution with the lowest cost is chosen as base for a seed for the main minimization in the Tracker.

We also performed tests when the default single-seed minimization is allowed 10 times more iterations than in the original PTAMM implementation. However, test results (not shown in this paper) indicate that increasing number of iterations does not improve the results of the Tracker suggesting that the minimization must be converging to a non global minimum.

5. Evaluation of Tracking Performance

To evaluate changes to PTAMM, two tracking experiments were performed. The first used a motion that had a single degree of freedom where a rotational motion stage was used to generate a known motion. However, the parameters of the motion like the speed and direction of movement were easily changed without changing other parameters. The second experiment used a freehand motion with 6 degrees of freedom. In both experiments, all the data used by PTAMM were created prior to running the algorithm.

5.1. Stage Results. The camera was attached to a rotation stage. The stage offers a communication protocol that was used to query its orientation. Video data and stage position measurements were acquired and stored during a stage rotation encompassing 160 deg (cf. Figure 4). In the resulting dataset, each video frame has a corresponding measurement of the stage rotation assigned to it. This allows simulation of arbitrary motion in the single degree of freedom of the rotation stage. Rotations following a sine wave were used for the evaluation. The PTAMM video input was modified to

(a) (b) (c)

Figure 3: Example of matching a newly added keyframe. From left to right: newly added keyframe, PTAMM match for a new keyframe, proposed FAST-based match. For the new method, note the overlap of the left parts of the keyframes which allows for addition of new 3D features to the scene. Using original PTAMM match would not initialize new features.

Figure 4: Image created by stitching together with some of the images used in the motion simulation. Stitched image covers 160 deg of rotation.

use stored images. A frame rate limiting mechanism was also added. Additionally, PTAMM was run to create and save a map with features covering the entire area seen by the camera during the stage swath so that the same map was used for all experiments.

Evaluation of the method using gyroscope measurements requires simulation of these measurements. Stage rotation measurements are used for that purpose. Due to camera mounting, stage rotation corresponds to rotation around the Y axis in the camera frame. To simulate the measurements from the IMU, an angular rate was approximated for the Y axis using the difference in angular position measurements from the stage and the time from the simulation. Guassian white noise was added to this approximation, where the variance of the noise was obtained from data collected from the IMU on the airframe used in experiments discussed later.

A series of runs were performed using the described simulation setup to evaluate PTAMM behavior under different video frame rates. For each frame rate, a sinusoidal rotation was simulated and appropriate video frames were fed into PTAMM. In all cases, the amplitude of the motion was set to 80 deg; however, the frequency of the sine wave was varied from 0.05 Hz to 1 Hz. This corresponds to an average angular velocity being varied from around 2.5 deg/second to over 50 deg/second. Right before the start of the test, the saved map was loaded into PTAMM and locked to prevent any modifications. The motion sequence was started, and PTAMM was allowed tracking for 30 seconds. Tracking results for each frame were recorded, and the ratio of the number of features found in the frame to the number of features in the PVS was used as a performance metric.

Four algorithms were compared; two standard PTAMM approaches: motion model and the SBI algorithm and two proposed algorithms: the extended SBI algorithm and use of gyroscope measurements. The results of experiments are presented in Figure 5. In all cases, the modified methods improve the ability of the Tracker to estimate camera pose at higher motion speeds. Improvement is more clear at lower frame rates. The method using gyroscopes gains advantage over extended SBI at low frame rates and at high motion frequencies due to decreased overlap between frames caused by the motion.

5.2. IMU Results. Another experiment used data captured during an aggressive freehand camera motion. Video input and gyroscopes measurements provided by the IMU attached to the camera were timestamped and saved at the PC. A PTAMM map of the scene was created before running the tests and stored as before. The captured video sequence was input to PTAMM at various frame rates.

Results are presented in Figure 6. Extended SBI performs better than the default version at the lower frame rates; however, the difference is not as significant as in the previous results. Gyroscope-based tracking performs very poorly though and is not shown in the figure because it failed in all cases. This might be the result of a couple of factors. Different time delays encountered on communication links from camera and from IMU can create a time offset between

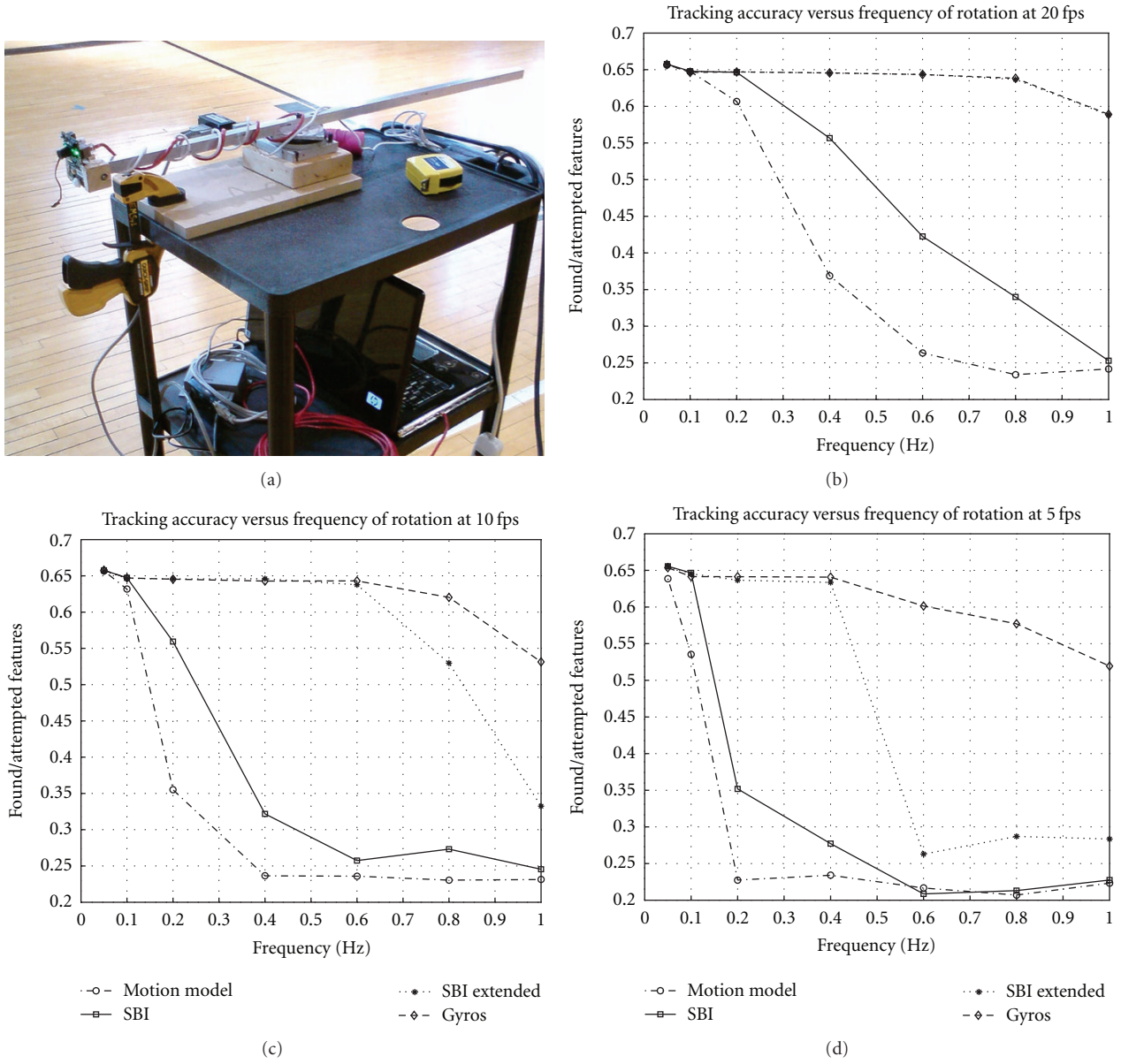

Figure 5: (a) Setup used to capture data for motion simulation. Comparison of the tracking quality for different methods of estimating camera pose seeded to the Tracker and at different video frame rates (b) 20 fps, (c) 10 fps, and (d) 5 fps.

the data from each. The estimate of the orientation of the IMU relative to camera may have been inaccurate as well.

6. PTAMM Airframe Integration

Integration of PTAMM with the airframe used for testing involved both hardware and software modifications to a multirotor UAV. An annotated picture of the airframe is shown in Figure 7. In the current implementation, the PTAMM solution runs on a laptop computer in the Linux operating system, rather than on board the aircraft. Therefore, the images are transferred to the ground either directly through an Ethernet cable or through an 802.11n wireless transceiver. The camera is an AXIS M1054, an IP security camera, with the resolution set at 800 × 500 pixels with 84 deg in the horizontal field of view. The wireless transceiver is capable of transmitting well over 30 frames per second with this resolution, which is not a limitation because the PTAMM algorithm on the laptop processes approximately 15 to 20 frames per second in normal operation.

The original PTAMM software uses a fisheye camera lens distortion model. However, this is not suitable for our camera, and therefore a more common second-order radial distortion polynomial model was added. This included adding a closed form solution for the forward distortion model and an iterative solution for the inverse of this model.

The PTAMM algorithm communicates with the OpenPilot autopilot through a 2.4 GHz serial communications

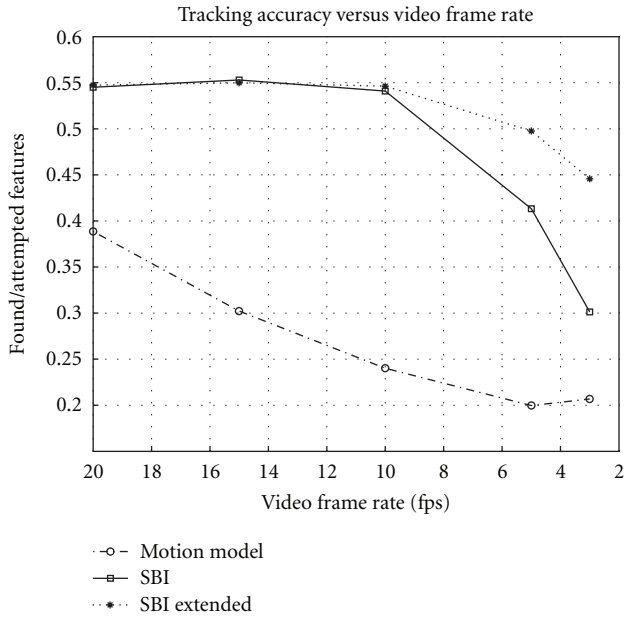

Figure 6: Comparison of the tracking quality for a real image sequence played at different frame rates.

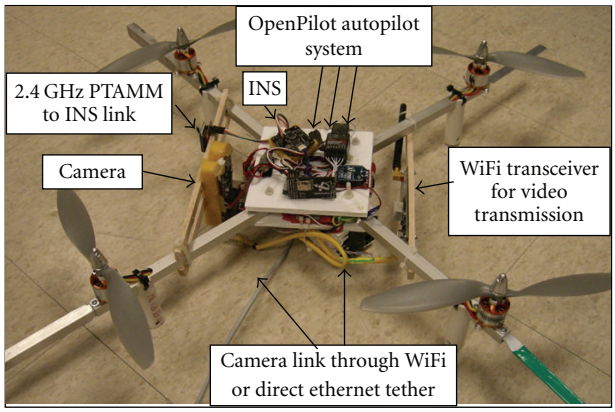

Figure 7: Multirotor airframe modified for PTAMM.

transceiver connected to the INS board of the autopilot. The OpenPilot system is an open source autopilot with hardware consisting mainly of an INS board, a main control board, a GPS receiver, and a 2.4 GHz serial communications link with the ground station. The INS board contains the IMU sensors and a microcontroller, which implements the navigation solution. The authors have contributed to several parts of the development of OpenPilot and most significantly to the development of the navigation solution.

The navigation solution in OpenPilot is an EKF implementation of an INS. It takes 3-axis accelerometer measurements and 3-axis rate gyro measurements as the inputs to the dynamic system modeled in the EFK. In addition, it uses a 3-axis magnetometer, GPS position, and GPS velocity as the measurements of the outputs of the system. The dynamic model used is a 6DOF kinematic model of a rigid body. When using PTAMM with the navigation solution, the

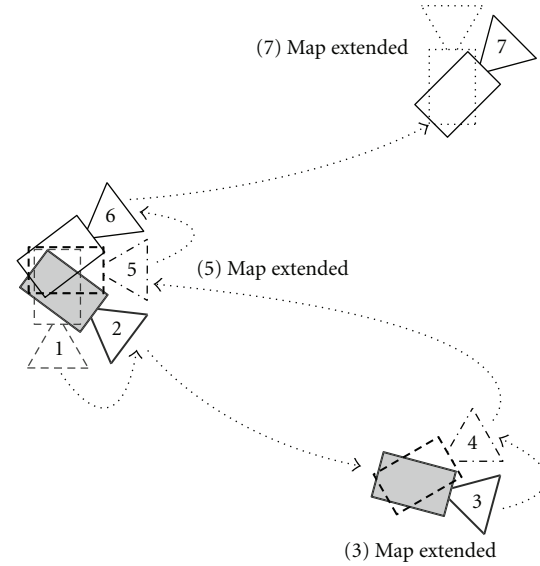

Figure 8: Map building with the new keyframe addition. Camera successive positions where new keyframe is added are numbered from 1 to 7. (1) PTAMM after stereo initialization, (2) camera rotated to point into part of the scene without initialized features, (3) camera moved forward to obtain stereo separation, keyframe added, and new features initialized, (4) camera rotated again to look at another uninitialized part of the scene, (5) camera moved back for stereo, new features added, and so forth.

GPS is not used. Rather, the position measurements from PTAMM replace the GPS position measurements, and no velocity measurements are used. Also, when using PTAMM, the magnetometers are not used, so the yaw angle of the airframe is not observable without another measurement. Since the both PTAMM and the INS represent orientation with a quaternion, the simplest solution to this is to add an additional output/measurement to the INS which corresponds to the forth element of the quaternion. This element of the quaternion is highly correlated with yaw.

7. Test Flights

The exploration and navigation capabilities of the modified PTAMM algorithm were verified in test flights. As shown in Figure 7, the camera looks horizontally as it is mounted on the UAV. Therefore, rotating the UAV around the Z axis (yaw) points it towards new parts of the scene. To add to the map features in a new area, PTAMM needs two images (frames) of that area separated in space to allow for a stereo view. It is possible to generate large maps during flights under stable control using the navigation solution being generated with position and yaw measurements from PTAMM. This is done fairly easily with the modifications, however, would be very difficult with original algorithm. A possible routine for moving and rotating the camera to extend the map is presented in Figure 8. The map is created by moving in and out in a "circle," with multiple keyframes being added at the center and two keyframes being added at the rim of the "circle."

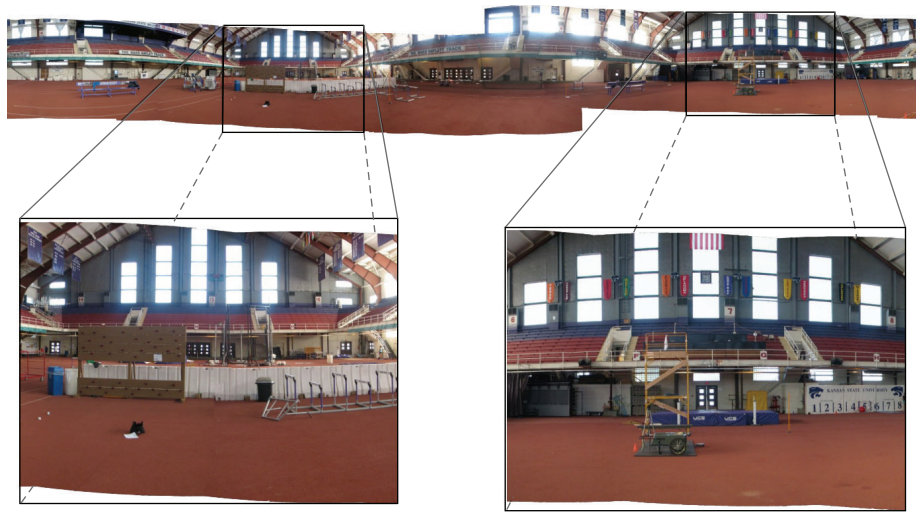

Figure 9: Panoramic view of a field house. One end of the field house is approximately 25 meters from the camera, other is around 125 meters from the camera.

Figure 9 shows a panoramic view of a 360 deg scene in a field house. It was possible to generate a map of nearly the entire 360 deg field of view while controlling the UAV with the PTAMM navigation solution and simultaneously building the map. This was done by rotating and moving back and forth a distance of about four meters. The two extreme ends of the field house, which are broken out in this figure, were approximately 25 meters and 125 meters away from the flight location. Despite the small stereo separation, the only portion of the map where the UAV developed poor stability was when it was facing the end of the field house that was furthest away.

Holding the UAV in hand, an entire 360 deg map was generated in the same way. In this case, however, the map was closed through the full rotation. The UAV was then subsequently flown using this map in the navigation solution. It was possible to make a full 360 deg rotation with stable control and a deviation in position of about one meter. While holding the quad in one position and manually rotating 360 deg, the PTAMM position solution showed a deviation of about half a meter. This indicates that part of the 1 meter motion while being controlled was due to the dynamic response to the disturbances of the half meter position errors in PTAMM solution.

8. Conclusions and Recommendations

The map building capabilities of PTAMM are significantly improved for the application of UAV navigation, and the robustness under low frame rate video is improved by the modifications described in this paper. The SLAM solution is successfully modified to provide the position measurements necessary for the navigation solution on a VTOL UAV while simultaneously building the map. It was used in flights where large maps were constructed in real time as the UAV flew under position control with the only position measurements for the navigation solution coming from SLAM. The main contributions include (1) extension of the map building algorithm to enable it to be used realistically while controlling a VTOL UAV and simultaneously building the map; (2) improved performance for low frame rates; (3) the first known demonstration of a monocular SLAM algorithm successfully controlling a UAV while simultaneously building the map. The work described in this paper demonstrates that a fully autonomous UAV that uses monocular vision for navigation is feasible, although several aspects of the solution still need improvement for practical application with robust performance.

Future work on the monocular SLAM algorithm for control of VTOL UAV is needed to make it more robust. This work would likely include two major things: (1) moving it to an on board processor that is more tightly linked with the inertial measurement sensors; (2) tighter integration with the control system. For example, in some cases, after adding a new keyframe, the map changes significantly, which causes large disturbances for the control system if the UAV is being controlled with this map. Tighter integration with the control system and consideration of effects such as these would significantly improve robustness.

References

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR '07)*, Nara, Japan, November 2007.

[2] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *Proceedings 10th European Conference on Computer Vision (ECCV '08)*, pp. 802–815, Marseille, France, October 2008.

[3] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Real-time monocular SLAM: why filter?" in *Proceedings of International Conference on Artificial Reality and Telexistence*, Adelaide, Australia, December 2010.

[4] R. O. Castle, G. Klein, and D. W. Murray, "Video-rate localization in multiple maps forwearable augmented reality," in *Proceedings of the 12th IEEE International Symposium on*

Wearable Computers (ISWC '08), pp. 15–22, Pittsburgh, Pa, USA, September-October 2008.

[5] A. Bachrach, R. He, and N. Roy, "Autonomous fight in unknown indoor environments," *International Journal of Micro Air Vehicles*, vol. 1, pp. 217–228, December 2009.

[6] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '10)*, pp. 21–28, May 2010.

[7] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of IMU and vision for absolute scale estimation in monocular SLAM," *Journal of Intelligent and Robotic Systems*, vol. 61, no. 1–4, pp. 287–299, 2011.

[8] J. H. Kim and S. Sukkarieh, "Airborne simultaneous localisation and map building," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '03)*, pp. 406–411, September 2003.

[9] J. Artieda, J. M. Sebastian, P. Campoy et al., "Visual 3-D SLAM from UAVs," *Journal of Intelligent and Robotic Systems*, vol. 55, no. 4-5, pp. 299–321, 2009.

[10] F. Caballero, L. Merino, J. Ferruz, and A. Ollero, "Vision-based odometry and SLAM for medium and high altitude flying UAVs," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1–3, pp. 137–161, 2009.

[11] R. O. Castle and D. W. Murray, "Object recognition and localization while tracking and mapping," in *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality*, pp. 179–180, IEEE Computer Society, Los Alamitos, Calif, USA, 2009.

[12] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proceedings of the European Conference on Computer Vision*, vol. 1, pp. 430–443, May 2006.

[13] O. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, no. 3, pp. 485–508, 1998.

[14] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 943–948, October 2004.

[15] S. Benhimane and E. Malis, "Homography-based 2D visual tracking and servoing," *Journal of Robotics Research*, vol. 26, no. 7, pp. 661–676, 2007.

[16] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR '09)*, pp. 83–86, Orlando, Fla, USA, October 2009.

[17] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[18] G. Bradski, "The openCV library," *Dr. Dobb's Journal of Software Tools*, 2000.