Monocular Parallel Tracking and Mapping with Odometry Fusion for MAV Navigation in Feature-lacking Environments

Duy-Nguyen Ta, Kyel Ok and Frank Dellaert

***Abstract*— Despite recent progress, autonomous navigation on Micro Aerial Vehicles with a single frontal camera is still a challenging problem, especially in feature-lacking environments. On a mobile robot with a frontal camera, monoSLAM can fail when there are not enough visual features in the scene, or when the robot, with rotationally dominant motions, yaws away from a known map toward unknown regions. To overcome such limitations and increase responsiveness, we present a novel parallel tracking and mapping framework that is suitable for robot navigation by fusing visual data with odometry measurements in a principled manner. Our framework can cope with a lack of visual features in the scene, and maintain robustness during pure camera rotations. We demonstrate our results on a dataset captured from the frontal camera of a quadrotor flying in a typical feature-lacking indoor environment.**

I. Introduction

Using mainly a single frontal camera for obstacle detection and avoidance to enable fully autonomous navigation on Micro Aerial Vehicles (MAVs) in unknown areas is still a big challenge, although camera is a desirable sensor for MAVs due to their limited payload and power capabilities [1], [2]. The frontal monocular camera design is also common on commercial MAVs[1]. Other sensors such as laser scanners are either heavy or power hungry, hence not preferred for lightweight MAVs in long-term navigation tasks. However, latest achievements in MAV navigation either rely on laser scanners [3], [4], [5] and known 3d maps [6] or use stereo cameras [7], [8] but require human-specified waypoints for collision-free trajectories [7].

Most state-of-the-art monocular SLAM systems are not yet ready for autonomous navigation. These systems typically build a map of corner-type features, which do not contain enough information for obstacle detection and avoidance, therefore, they need to rely on human assistance for robot navigation [7]. Moreover, feature-based monoSLAM systems might fail in common indoor environments, as in Fig. 1. This is because monoSLAM often requires an ample amount of visual features, but these environments do not possess many distinct corner-type features, while the few that are present are mostly far-away, providing only little information to localize the camera reliably. Also, due to the lack of motion parallax needed to triangulate new landmarks in the scene, pure monoSLAM systems will easily break down when the camera undergoes a pure yaw or pitch motion toward an unknown region. This further limits applications of pure monoSLAM in robot navigation, since pure yaw is a very common robot motion. Using inverse-depth representation [9] can mitigate this issue; however, its non-Gaussian priors might decrease the system's robustness. This helps explain why successful applications of monocular SLAM on micro-air vehicles use a bottom-facing camera, in which the camera undergoes only a limited yawing and pitching motion compared to the frontal camera [10], [11]. Unfortunately, bottom-facing cameras do not help in obstacle detection.

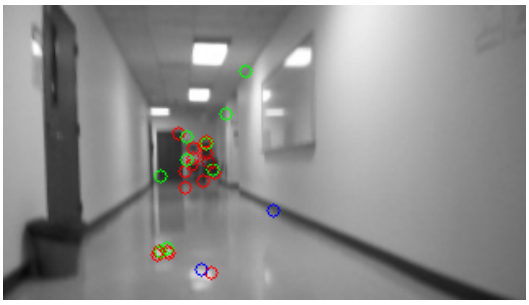


Fig. 1: Feature-based monoSLAM is not suitable for robots with a frontal camera, especially in this type of indoor environment with only a few features that are mostly far away along the robot's direction. Our framework for robot navigation overcomes these limitations by fusing visual and odometry measurements together in a principle manner.

Instead of using corner-type features, our previous system proposed a special type of features which capture the structure of the environment [2], allowing the robot to infer and avoid lateral walls. However, while controlling aerial vehicles requires fast responses, inferring scene structure from elementary features is still not fast enough, contributing largely to the latency and nonrobustness of the whole system.

This paper presents a parallel framework for robot navigation, improved upon the well-known Parallel Mapping and Tracking (PTAM) [12] framework, to increase the responsiveness and robustness of the whole system. Although PTAM is originally designed for Augmented Reality (AR), its parallel framework enables fast camera localization with a real-time tracker and global map building with a keyframe-based mapper, both of which are desirable in robotics applications. The tracker guarantees a fast response to changes in the environment, while the mapper builds a high-quality map of the environment and performs tasks such as structure inference. Also, the keyframe-based and parallel structure of PTAM have proven the system superior over traditional monocular SLAM filtering methods [13].

The authors are with the Center for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, Georgia, USA,
{duynguyen, kyelok, dellaert}@gatech.edu

[1]http://ardrone2.parrot.com/

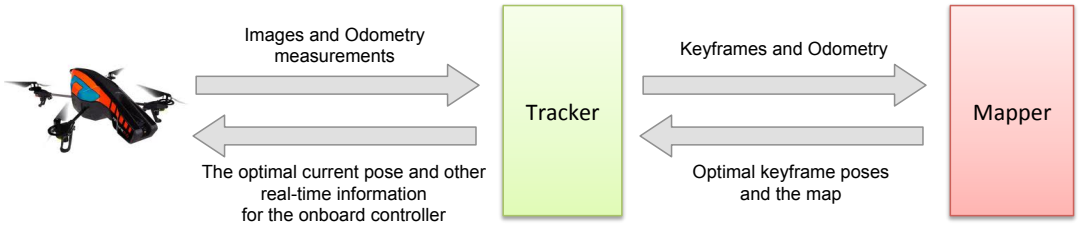


Fig. 2: Overall framework of our system

Recent years have also witnessed an increasing interest in using PTAM for robot navigation [10], [11]. However, beside the aforementioned problems of using pure monocular SLAM for robot navigation, the particular parallel framework of PTAM does not provide a flexible mechanism to fuse visual measurements with other sensor data such as odometry or IMU, making PTAM in robotics not as strong as it is in its own AR domain. To address such limitations, some recent work [10], [11] attempts to combine PTAM with odometry or IMU measurements, but do not integrate them tightly. These systems simply treat PTAM as a "black box" and combine PTAM's results with IMU measurements in an outer loop; hence, the visual and IMU measurements are not fused together in the same system, nor are the fused results propagated back to PTAM to prevent it from failing, e.g., when there is a lack of visual features in the scene.

Our system overcomes the aforementioned difficulties of monocular SLAM in robot navigation, and achieve robustness with rotationally-dominant movements and texture-poor indoor environments. The system is similar to PTAM in that it consists of a mapper that builds a map of the environment, and a tracker that localizes the camera within that map. However, our system differs from PTAM and its variants in the following key ways: first, while PTAM assumes a simple decaying velocity motion model, our system uses the odometry measurements to replace the motion model. Second, while PTAM purely relies on a large number of visual features to localize the camera, our system uses these odometry measurements to deal with a lack of visual features in the environment. Third, while other PTAM-IMU fusion work treats PTAM as a black box [10], [11], our system fuses visual and odometry measurements together in the same framework in a principled way, preventing breakdowns due to either a lack of features in the environment or a lack of motion parallax in the camera movement.

We demonstrate our framework on a dataset captured from the frontal camera of an AR.Drone quad-rotor flying in a typical indoor environment. We present a simple loop-closure detection method for indoor environments with straight orthogonal hallway segments. For comparison, we show the results of PTAM and how it breaks down in this type of environment. Finally, we evaluate our system's mapping results using the ground-truth map, and compare its accuracy against the PTAM's and the quad-rotor's own estimates.

II. The Framework

Fig. 2 shows the overall layout of our framework, consisting of a tracker and a mapper running in parallel. The ***tracker*** receives a stream of timestamped images and odometry measurements from the robot. Its task is to localize the current camera in real-time within the optimized map of local landmarks created and maintained by the mapper. Basing on the localization results, it then decides if a new keyframe and odometry measurements should be added to the existing map to cover new parts of the scene. Performing these tasks in real-time, the tracker's localization results benefit time-critical tasks such as controlling the robot to follow a trajectory, or avoiding obstacles.

The ***mapper*** periodically receives a new keyframe image from the tracker, as well as odometry measurements between this new keyframe and the previously received keyframe. These odometry measurements are created by accumulating all odometry measurements of the intermediate frames between the two keyframes. The mapper's task then is to build a global map of visual landmarks in the environment, and simultaneously compute the optimal keyframe poses. After finishing an iteration, the mapper updates the tracker with the newly optimized local map and keyframe poses, which are then used by the tracker to localize the current camera pose in real-time. Due to the inherently slower pace of the mapper and the fact that it does not have to obey real-time constraints, other time-consuming tasks such as inferring the structure of the environment can also be added to the mapper thread, to provide contextual information of the environment.

III. The Mapper

Unlike PTAM, which only uses visual measurements, we employ both visual measurements and odometry measurements between keyframes to build the map. We denote the set of unknown camera poses and landmarks by $X = \{x_i\}_{i=1}^{n}$ and $L = \{l_j\}_{j=1}^{m}$ respectively, the set of all visual measurements by $Z = \{z_{ij}\}$, with z_{ij} the visual measurement of landmark j viewed from camera i, and the set of all odometry measurements $B = \{b_{ik}\}$, with b_{ik} the odometry measurement between the camera poses i and k.

The map-building problem is then to recover the maximum a posteriori (MAP) estimate, given by

$$\begin{aligned} X^*, L^* &= \underset{X,L}{\operatorname{argmax}}\, p(X, L, Z, B) \\ &= \underset{X,L}{\operatorname{argmax}}\, p(x_1) \prod_{i,k} p(x_i | x_k, b_{ik}) \prod_{i,j} p(z_{ij} | x_i, l_j) \end{aligned}$$

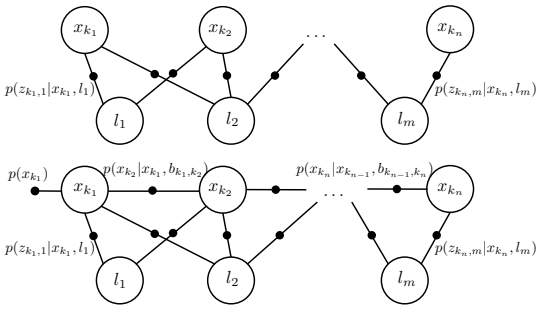


Fig. 3: The factor graphs of PTAM's mapper (top) and our mapper (bottom)

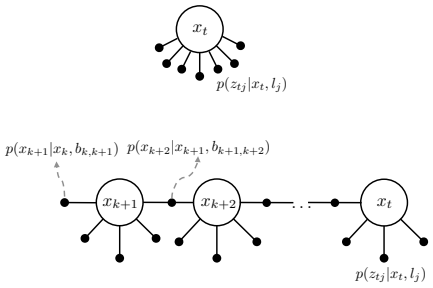


Fig. 4: The factor graphs of PTAM's tracker (top) and our tracker (bottom)

This general map-building problem can be posed in terms of inference in a factor graph [14]. As shown in the bottom of Fig. 3, the camera poses x_i and the landmarks l_j are represented as variable nodes (white circles) in the graph. The factor nodes (black dots) in the graph represent the prior densities $p(x_1)$ on the variable nodes, the motion models $p(x_i|\, x_k, b_{ik})$ between two poses x_i and x_k given the odometry measurement b_{ik}, and the measurement likelihood models $p(z_{ij}|\, x_i, l_j)$ constraining a pose x_i and a landmark l_j, given the corresponding visual measurement z_{ij}. The measurement model can be derived for any type of landmark and their visual measurements, using any standard camera projection model. In Section VI-B we give a detailed measurement model for the specific landmarks we use. For comparison, the factor graph of the original PTAM is shown at the top of Fig. 3. It has no odometry factors between camera poses.

This problem can be solved by techniques like bundle adjustment [15], [12], smoothing and mapping [14], or incremental smoothing and mapping methods [16]. We use the state-of-the-art incremental smoothing and mapping algorithm iSAM2 [16] implemented in the GTSAM library[2] for the actual inference.

IV. The Tracker

While our mapper is fairly similar to PTAM's, our tracker differs significantly. PTAM's tracker can robustly localize the current camera pose x_t within the known map (received from the mapper), by relying on the visual measurements z_{tj} of *many* landmarks l_j visible in the current image t. To do so, it solves the well-known camera resectioning problem, i.e., computing the optimal camera pose from measurements of known landmarks:

$$\begin{aligned} x_t^* &= \operatorname*{argmax}_{x_t} p(x_t|\, \{z_{tj}, l_j\}_{j=1..m}) \\ &= \operatorname*{argmax}_{x_t} \prod_j p(z_{tj}|\, x_t, l_j) \end{aligned}$$

The corresponding factor graph is very simple: there is only a single variable node x_t, and a single (unary) resectioning factor for each of the visual measurements, parameterized by the *corresponding* known landmark. The top of Fig. 4 shows the factor graph of PTAM's tracker.

[2]https://borg.cc.gatech.edu/download

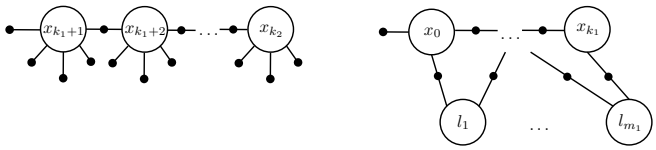
With the odometry measurements to compensate for the lack of visual measurements and landmarks in the environment, our tracker is more involved. We compute not only the current pose, but also all previous poses since the latest keyframe received from the mapper. To make things precise, let $X_{]k,t]} = \{x_i\}_{i=k+1}^{t}$ be those unknown camera poses, with k the index of the latest keyframe received. Also, let L^k be the set of known landmarks received from the mapper, $Z^k_{]k,t]}$ the visual measurements from frame $k+1$ to frame t, and $B_{[k,t[} = \{b_{i,i+1}\}_{i=k}^{t-1}$ the set of odometry measurements from frame k to t. Our tracker computes:

$$\begin{aligned} X^*_{]k,t]} &= \operatorname*{argmax}_{X_{]k,t]}} p(X_{]k,t]}|\, x_k, L^k, Z^k_{]k,t]}, B_{[k,t[}) \\ &= \operatorname*{argmax}_{X_{]k,t]}} \prod_i p(x_i|\, x_{i-1}, b_{i-1,i}) \prod_{i,j} p(z_{ij}|\, x_i, l_j) \end{aligned}$$

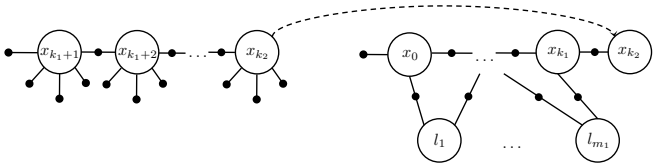
where $i \in]k,t]$ and where the index j ranges over landmarks in L^k observed in frame i. The corresponding factor graph is shown in the bottom of Fig. 4. We also use the GTSAM library mentioned above to optimize this graph.

At runtime, the tracker decides when to send a keyframe to the mapper, to request for new landmarks in the new parts of the scene. While waiting for the results from the mapper, it keeps on adding new camera poses and optimizing the graph as it receives new images and odometry measurements from the robot. In our experiments, the tracker simply sends a keyframe to the mapper after every 10 frames. Based on the speed of our robot and the field of view of its frontal camera, we found that this 10-frame sampling frequency is good enough for the mapper to cover the space, while preventing redundant overlap between them.

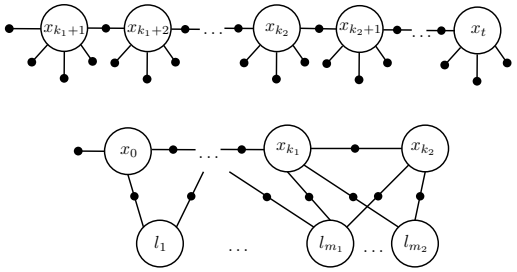
Upon receiving a new keyframe pose and landmarks back from the mapper, the tracker removes from the graph all the past frames before the latest keyframe received, including the keyframe itself. This process is shown in Fig. 5. Based on the new optimal keyframe pose and landmarks, it then updates the prediction, redoes data association, and re-optimizes all the remaining poses in the graph using the new prediction as the initial value. The data association step might not be necessary if the keyframe and landmark updates from the mapper are not very different from their values currently used in the tracker. However, this is not the case at loop closure, as we will discuss next.



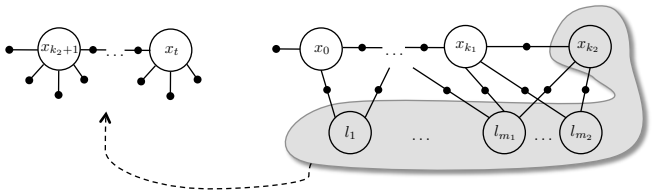
(a) Factor graphs of the tracker (left) and the mapper (right) before update



(b) The tracker sends a new keyframe (and its odometry measurement from the previous keyframe) to the mapper.



(c) While the mapper (top) is computing new landmarks and optimizing the graph, the tracker (bottom) keeps on adding new poses and measurements.



(d) After receiving the new map and the latest keyframe pose from the mapper, the tracker removes the past poses and updates the remaining graph.

Fig. 5: The communication and updating process between our tracker and mapper

V. Loop Closing

While PTAM cannot close the loop, as mentioned in its original paper [12], our use of odometry factors allows us to employ a simple loop closing mechanism. We implement the loop closure module in the mapper thread due to its high computational complexity.

Although feature-based loop closure detection schemes [17], [18] are available, we employ a simple yet effective loop-closure detection method that can work for typical indoor environments with straight orthogonal hallway segments. Our method is based on the Small-Blurry-Image (SBI) technique used in PTAM to relocalize the camera when the tracking is lost [19].

In particular, to detect a loop closure, we first determine if current hallway segment is potentially a pre-visited segment. When the robot makes a turn around a corner, based on a corner detection scheme [2], we know that it is entering a new segment. Thus, we can divide the trajectory into segments, and average the yaw angles of the robot during its stay on a segment and use it as the segment direction. If the current segment direction is similar to another segment in the past, a potential match is found.

To confirm a loop closure from a potential match, we find the keyframe in the old candidate segment that best matches with our current keyframe, and determine if their difference is small enough to consider them a loop closure. We do this by computing the SBIs, down-sampled and blurred images [19] for keyframes in all segments, and using the sum-of-square-differences (SSD) of pixels from the candidate keyframe's SBI and current SBI to measure the difference between them. In order to aid the matching, we also use image alignment technique to optimize for the relative camera rotation that best aligns the two SBIs before calculating the SSD. Finally, if the best match has differences less than a threshold, we consider it a good match, and a loop closure is found.

When a loop closure is found, we add a loop-closure factor constraining the relative pose between the latest keyframe and the best match keyframe to the mapper. Since keyframes in the mapper regularly sample the space as the drone moves at a constant speed, and the SBI match guarantees that the two frames are very close together, we simply predefine the values for this constrained relative pose measurement and use a reasonable large uncertainty as its noise model. We note that other traditional feature-based techniques, e.g. detecting and matching features between the two keyframes using fundamental matrix RANSAC, can also be used; however, their effectiveness is questionable with the lack of features and the regularity of indoor environments.

At loop closure, after receiving the latest keyframe and landmarks from the mapper, it is important for the tracker to redo the data associations for all visual measurements of previous frames in its graph and re-compute the initial values of those poses for optimization. This is because when closing the loop, the latest keyframe can be shifted very far back to an old place in the map, moving the attached frames together with it due to their odometry constraints. Hence, the visual measurements need to be re-associated with these old landmarks, and the initial values need to be recomputed to make sure the nonlinear optimization process converges at the correct local minima.

VI. Experiments

A. Platforms

We evaluate our system using a dataset captured from an AR.Drone[3] 2.0 quad-rotor, flying twice around a square hallway, shown in Fig. 7, at 1 meter above the ground.

The quad-rotor is equipped with a 30fps HD frontal camera, a 60fps QVGA bottom camera, an ultrasound height sensor, and an IMU system with a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer. An on-board server program on the quad-rotor filters measurements from the bottom camera, the height sensor, and the IMU system to estimate the pose of the quad-rotor [20].

[3]http://ardrone2.parrot.com/

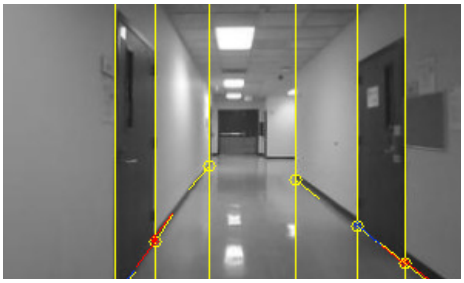


Fig. 6: Wall-floor Intersection Features. More details in [2].

We developed an additional on-board program to obtain images from the frontal camera and stream JPEG compressed 320×180 grayscale images to the tracker running on a MacbookPro 6.2 laptop with a 2.4 GHz Intel Core i5 CPU and 4GB DDR3 RAM. Our on-board program also receives estimated robot poses from the on-board server and creates odometry measurements between the frames, using their relative pose. However, as shown in Fig. 7, these odometry measurements are unreliable. They systematically underestimate the traveled distance, perhaps due to the lack of features on the floor corrupting the motion estimation scheme that largely depends on the features from the bottom-facing camera [20].

B. Wall-floor Intersection Features

To overcome the aforementioned problems of corner-type features in typical indoor environments (Fig. 1), we use Wall-floor Intersection Features (Fig. 6) in our experiments. As detailed in [2], these features lie on the intersection of a vertical line on the wall and the intersecting floor plane. We represent each landmark as an element of the Lie-group $\mathbb{SE}(2)$, encoding its 2D position and direction on the floor. The projection of a landmark onto an image given the camera pose is also an element of $\mathbb{SE}(2)$ encoding its projected point and projected direction in the image.

C. PTAM results

We first run PTAM with our dataset to analyze its performance in feature-poor environments. We initialize the system as required [12] using the last frame when the quad-rotor is on the floor and the first frame when it starts stabilizing in the air. Since the quad-rotor is flying at about a 1 meter height, we set the initial scale of the system, determined by the baseline between these two frames, as 1 meter.

Fig. 7 shows the trajectory of the quad-rotor estimated by PTAM. As we predicted, PTAM cannot estimate the motion and shows that the quad-rotor does not move. This is because there are not enough corner features in the scene, and the few that are available are mostly at the end of the hallway, as shown in Fig. 1, providing little information about the robot's forward movement; these features have almost no motion parallax as they are close to the focus-of-expansion point on the image. PTAM also breaks down when the quad-rotor starts to turn around the first corner, failing to initialize new landmarks with no motion parallax.

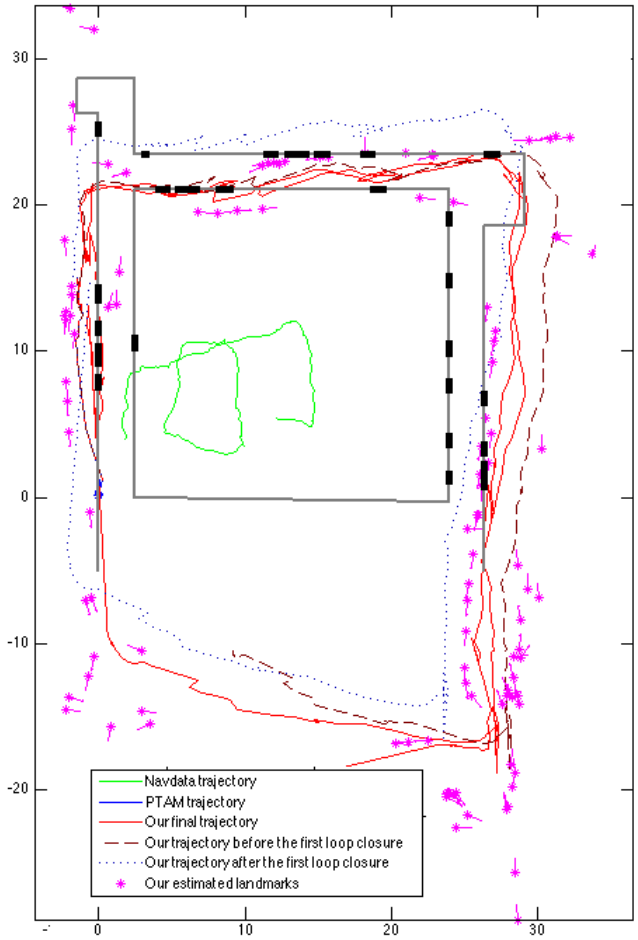


Fig. 7: Comparison between the ground-truth map of the environment and the estimated trajectories from the quad-rotor's on-board program, PTAM, and our system. Our map is manually-aligned to fit with the ground-truth map. The Wall-Floor Intersection Features are shown in magenta and the ground-truth floor layout is in grey (walls) and black (doors). Our estimate roughly matches with the ground-truth map even with an unreliable odometry input and a small number of visual features.

D. Our results

Using the same dataset, we show the results of our system in Fig. 7, where the estimated trajectory is in red and the landmarks in magenta. Due to drift and unreliability in sensor readings during AR.Drone's take-off sequence [20], we only start our system once the drone stabilizes in the air. Since the entire map depends on the first robot pose at the system start, which is arbitrary due to the drift, we manually rotate our map to match the ground-truth map orientation.

Our system successfully finishes the sequence with a proper loop closure. With the aid of the odometry measurements, our system does not fail when the robot yaws around the corner, or when it cannot detect any visual features in the environment. For example, during the last hallway segment before the loop closure at the bottom of Fig. 7, it relies

solely on the incorrect odometry measurements obtained from the quad-rotor's on-board program and underestimates the amount of its forward motion. This is evident in the estimated trajectory before the first loop closure, shown in dashed lines in Fig. 7. Due to the underestimated forward motion, the estimation results show that it is at the middle of the bottom hallway, when it has already turned around the corner. However, with the loop closure in our system, this is corrected, as shown in the trajectory after the first loop closure in Fig. 7. Our final map and trajectory roughly match the ground-truth map of the environment, even with the odometry input that drifts significantly.

In terms of speed, our unoptimized tracker runs favorably at 10-12fps, while the mapper runs at 1-2fps with the majority of time is spent on waiting for the new keyframe and odometry measurements from the tracker. Comparing to iSAM2 [16], which is also very fast due to its incremental computation nature, our framework allows the extra resource in the mapper to be utilized for other time-consuming wall inference tasks without affecting the tracker speed.

VII. Conclusion and Future Work

We have shown that PTAM is not suitable for robot navigation with a frontal camera, because (1) features in front of the robot do not provide enough information for localization when the robot moves forward, and (2) there is not enough motion parallax to triangulate new landmarks in the scene when the robot rotates to change its heading. In fact, motions typically performed by robots, mainly moving forward and purely rotating to change direction, are challenging for any monocular visual SLAM system in general. The lack of visual landmarks in typical indoor environments add to the challenge for feature-based systems.

We have presented a new parallel tracking and mapping framework that fuses visual information with odometry measurements for robot navigation. We have shown that by fusing the odometry measurements, our framework can overcome the key difficulties in frontal monocular SLAM for robot navigation. Odometry information helps to prevent breakdowns when the visual features do not provide enough information to localize the camera. Our new framework also enables a simple loop-closing mechanism to correct the robot trajectory and the map.

Our future work is to further leverage the benefits of the parallel tracking and mapping framework for robotics applications. While the slow-paced mapping thread allows room for complex structure inference and planning tasks, the fast tracking thread guarantees fast responses to changes in the environment and provides real-time information for control. In our current implementation, the tracker and mapper communicate with each other over TCP/IP and we plan to implement the tracker on the quad-rotor itself so that its results are immediately accessible to the on-board controller. The mapper can also be improved to utilize other visual information in the environment, using state-of-the-art structure inference methods.

Acknowledgements

This work is supported by an ARO MURI grant, award number W911NF-11-1-0046.

References

[1] C. Bills, J. Chen, and A. Saxena, "Autonomous MAV flight in indoor environments using single image perspective cues," in *Robotics and Automation, 2011. ICRA 2011. Proceedings of the 2011 IEEE International Conference on*, 2011.

[2] K. Ok, D.-N. Ta, and F. Dellaert, "Vistas and wall-floor intersection features - enabling autonomous flight in man-made environments," in *Workshop on Visual Control of Mobile Robots ViCoMoR*, 2012. [Online]. Available: http://frank.dellaert.com/pubs/Ok12vicomor.pdf

[3] A. Bachrach, R. He, and N. Roy, "Autonomous flight in unknown indoor environments," *International Journal of Micro Air Vehicles*, vol. 1, no. 4, pp. 217–228, 2009.

[4] A. Bachrach, S. Prentice, R. He, and N. Roy, "Range–robust autonomous navigation in gps-denied environments," *J. of Field Robotics*, vol. 28, no. 5, pp. 644–666, 2011.

[5] S. Shen, N. Michael, and K. V., "Autonomous navigation in confined indoor environments with a micro-aerial vehicle," *IEEE Robotics & Automation Magazine*, Jan 2012.

[6] A. Bry, A. Bachrach, and N. Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 1–8.

[7] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor," in *Robotics: Science and Systems (RSS)*, 2013.

[8] K. Schmid, M. Suppa, and D. Burschka, "Towards autonomous mav exploration in cluttered indoor and outdoor environments," in *RSS 2013 Workshop on Resource-Efficient Integration of Perception, Control and Navigation for Micro Air Vehicles (MAVs)*, Berlin, Germany, June 2013.

[9] J. Montiel, J. Civera, and A. Davison, "Unified inverse depth parametrization for monocular SLAM," in *Robotics: Science and Systems (RSS)*, Aug 2006.

[10] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 4531–4537.

[11] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard imu and monocular vision based control for mavs in unknown in- and outdoor environments," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[12] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, Nara, Japan, Nov 2007, pp. 225–234.

[13] S. H., M. J. M. M., and D. A. J., "Real-time monocular SLAM: Why filter?" in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2010, pp. 2657–2664.

[14] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, Dec 2006.

[15] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment – a modern synthesis," in *Vision Algorithms: Theory and Practice*, ser. LNCS, W. Triggs, A. Zisserman, and R. Szeliski, Eds. Springer Verlag, 2000, vol. 1883, pp. 298–372.

[16] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Intl. J. of Robotics Research*, vol. 31, pp. 217–236, Feb 2012.

[17] A. Angeli, S. Doncieux, J.-A. Meyer, and D. Filliat, "Real-time visual loop-closure detection," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2008, pp. 1842–1847.

[18] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. D. Tardos, "An image-to-map loop closing method for monocular slam," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

[19] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *European Conf. on Computer Vision (ECCV)*, Marseille, France, 2008.

[20] P. Bristeau, F. Callou, D. Vissière, and N. Petit, "The navigation and control technology inside the ar. drone micro uav," in *World Congress*, vol. 18, no. 1, 2011, pp. 1477–1484.