Monocular Image Space Tracking on a Computationally Limited MAV

Kyel Ok[1], Dinesh Gamage[2], Tom Drummond[2], Frank Dellaert[3], and Nicholas Roy[1]

***Abstract*— We propose a method of monocular camera-inertial based navigation for computationally limited micro air vehicles (MAVs). Our approach is derived from the recent development of parallel tracking and mapping algorithms, but unlike previous results, we show how the tracking and mapping processes operate using different representations. The separation of representations allows us not only to move the computational load of full map inference to a ground station, but to further reduce the computational cost of on-board tracking for pose estimation. Our primary contribution is to show how the cost of tracking the vehicle pose on-board can be substantially reduced by estimating the camera motion directly in the image frame, rather than in the world co-ordinate frame. We demonstrate our method on an Ascending Technologies Pelican quad-rotor, and show that we can track the vehicle pose with reduced on-board computation but without compromised navigation accuracy.**

I. Introduction

We are interested in *monocular vision-based*, inexpensive, and potentially disposable MAVs that can be deployed in a large volume, e.g. a team of MAVs exploring a post-disaster site on a search and rescue mission. Apart from the obvious benefits of low cost, using a low-power, lightweight camera coupled with a small processing unit increases the flight time of a MAV and allows a smaller form factor.

Amongst the challenges in operating cost-effective MAVs, we are particularly interested in computationally efficient techniques for estimating the vehicle pose x_t, in order to control the pose and the velocity of the vehicle. On a monocular visual-inertial based MAV where we obtain camera images $I_{0:t}$, the vehicle pose is given by the maximum of $P(x_t|I_{0:t})$. However, there rarely is a way to compute a globally consistent pose estimate directly from image pixel values and an IMU. The conventional approach is to estimate the vehicle pose with respect to a set of landmarks $L = \{l_i\}_{i=1}^{n} \in \mathbb{R}^3$ in the world that can be sensed as features in the images. Without a prior map of the landmarks, we must then estimate the pose using the posterior $P(x_t, L|I_{0:t})$, essentially the landmark-based visual simultaneous localization and mapping (SLAM) problem.

Solving the complete joint posterior over x_t and L as a SLAM process can be computationally demanding, especially as the number of landmarks grows large. Klein and Murray [1] addressed the cost of solving for the entire joint posterior by decomposing the computation into two parallel processes: a full-rate camera tracking process that uses the best available information, and a global map optimization process using a subset of representative camera images, i.e. keyframes. This decoupling requires sharing a single map in memory where the camera tracking process assumes a fixed map, while in parallel, the map optimization process continuously (if slowly) improves the map. Although proven very successful, this technique still carries the burden of optimizing a growing global map; this difficulty is amplified on the kinds of low-power processors found in low-cost air vehicles that lack support for true parallelism.

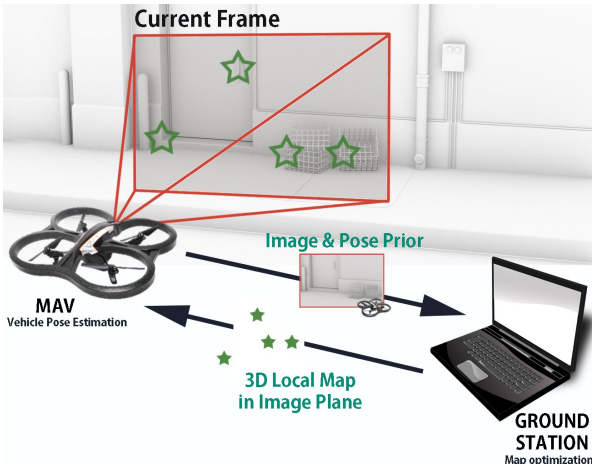

Fig. 1: The MAV occasionally sends the camera image and its own pose estimate to the ground station for map building. The ground station sends back a local map in the image space of the MAV for fast pose tracking.

An easy solution to reducing the cost of on-board computation is to move the entire pose computation to an off-board processor at a ground station. Compressed images or feature-descriptor sets could be sent to a ground station and a pose estimate could be received in return. However, the wireless communication channel to an untethered MAV will typically suffer from packet drops, limited bandwidth and large transmission delays that make time-critical dependence on the communication channel unreliable. A complete off-board scheme that includes an unreliable communication channel in its control loop is not suitable for a MAV.

To overcome the communication limits while still reducing the cost of on-board computation, only the costly map optimization can be moved to an off-board processor. A trivial solution is to send the entire map back and forth between

[1]The authors are with CSAIL, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA {kyelok, nicholas.roy}@csail.mit.edu

[2]The authors are with ARC Centre of Excellence for Robotic Vision, Monash University, Melbourne, Australia {dinesh.gamage, tom.drummond}@monash.edu

[2]Dellaert F. is with the Center for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, Georgia, USA dellaert@cc.gatech.edu

the MAV and ground station, keeping different copies of the same information. However, once the on-board process is restricted to tracking, this process can be reformulated in the image space, leading to an even greater reduction in computation. The mapping process can continue to run off-board, and provide asynchronous copies to the MAV of the updated map as the communication channel permits, where the updated map is projected into image space and bounded by visibility in the current frame, specifically for the purposes of fast, on-board pose estimation.

The main contribution of this paper is to show that if we reformulate the tracking and mapping problem so that the two processes are physically decoupled and no longer share the same copy of the map, the tracking process can be entirely in the image co-ordinate frame for fast computation. We introduce monocular image space tracking (MIST) and discuss landmark representation in the image space, data association using such landmarks, fast pose optimization using precomputed image space Jacobians, to update the landmarks frame-to-frame, and forward projection for compensating for the delay in asynchronous updates from the ground station.

Our novel approach of asymmetrically distributing SLAM onto separate devices results in fully scalable robust pose estimation on a computationally and bandwidth limited MAV, while the globally consistent map of arbitrary size can still be inferred on the ground. We demonstrate the improvements achieved with MIST using a monocular camera mounted on an Ascending Technologies Pelican quad-rotor.

II. Related Work

In the field of autonomous navigation and exploration, using planar laser scanners have shown success in achieving full autonomy for micro air vehicles [2], [3]. However, these methods do not adapt well to inexpensive platforms due to the weight, power requirement, and the cost of laser scanners. Recent work [4], [5] has demonstrated similar exploration capabilities using stereo vision as primary sensing means. Although abandoning the laser scanner reduces the weight and power requirements, stereo vision techniques are still computationally expensive. We would like to push towards the limits of minimal sensor suite and minimal processing efforts by utilizing a monocular camera.

Past work in using monocular camera on a MAV for on-board SLAM [6], [7] has used PTAM as a black-box pose estimation unit. However, originally developed for augmented reality applications in small workspaces where the explored map is assumed to be small, PTAM's computational requirements would still be a burden on computationally limited MAV platforms exploring a larger area.

Recent work by Forster et al. [8] reduced the processing requirements greatly, and demonstrated robust high frame-rate tracking using a small processing unit. Although their SVO method has low frame-to-frame computational requirements, it is not fully scalable due to the burden of storing a growing map in the MAV's memory. Also, SVO requires a two core processor for optimal speed for its parallel tracking and mapping design; our system only requires a single core on the MAV with all of the heavy computational load on the ground station. Lastly, SVO is currently engineered to use a downward camera. Thus, an algorithm that is as fast on a single core processor with small memory capacity and that could work with both downward and forward cameras would be much more desirable.

As opposed to on-board methods, some previous work [9], [10], [11] has streamed images from the MAV to a more capable ground station to off-board the computation. However, such strategy requires aggressive image compression and reduced frame rates, leading to overall poor image quality. Computing the pose of the MAV on a ground station, and streaming it back to the MAV would also introduce a large transmission delay in the pose updates needed by the on-board controllers. There are techniques for mitigating the controller errors that can result from a delay in the state estimate [10], but these solutions ultimately are not as robust as a high-rate on-board state estimation process.

Lastly, there is other previous work [12], [13], [14] that partitions the SLAM problems to meet different objectives, but our work is novel in dividing the problem onto two separate devices to meet the requirements of a computationally limited system.

III. Overview

We divide the full SLAM problem of computing $P(x_t, L|I_{0:t})$ into two processes: fast pose-tracking on the MAV and keyframe-based smoothing and mapping on the ground station. On the MAV, we compute the pose $x_t \in SE(3)$ given the current image I_t, and the transmitted local map $\bar{L}$ of landmarks $\{\bar{l}_i\}_{i=1}^{q}$ in the image space, i.e., we compute the posterior probability $P(x_t|I_t, \bar{L})$ of the pose.

Conversely, on the ground station, we take the smoothing and mapping (SAM) approach [15] of solving the SLAM problem. So, instead of computing $P(x_t, L|I_{0:t})$, we optimize over a selected subset of camera poses $\bar{X} = \{x_i\}_{i=1}^{r}$, i.e. *keyframe* poses that are far apart each other, and compute $P(\bar{X}, L|\bar{\mathcal{I}})$. We do this by occasionally receiving a subset of camera images $\bar{\mathcal{I}} \subset I_{0:t}$ from the MAV, and optimizing for the *maximum a posteriori* (MAP) estimates $\bar{X}^*$ and L^*,

$$\bar{X}^*, L^* = \underset{\bar{X}, L}{\operatorname{argmax}} P(\bar{X}, L|\bar{\mathcal{I}}). \tag{1}$$

This tracking and mapping approach is re-formulated into a monocular image space tracker (MIST) and a keyframe-based SAM as described in the following sections.

IV. Computation on the MAV

Using MIST requires observing landmarks in the image space, updating them frame-to-frame, rapidly calculating the vehicle pose by leveraging a special structure in the Jacobians, and finally forward projecting asynchronous landmark updates from the ground station to compensate for the communication delay. This image space tracking framework, as illustrated in Fig. 2, is discussed in this section.

A. Feature Extraction and Data Association

We adopt a feature based approach, where we extract a set of feature locations $F_t = \{f_k^t\}_{k=1}^p$ in the 2D image space given the current image I_t, i.e., $F_t = f(I_t)$ where f is a function that selects a set of highly re-observable locations. At the feature locations, we extract an 8 by 8 patch as the descriptor $d_k^t \in [0, 255]^{64}$ for each feature f_k^t. We do not use any special descriptors [16], [17] to minimize the computation on the MAV. The descriptors are used to match the landmarks in the local map $\bar{L}$ to a feature, forming associations $J_t = \{j_k^t\}_{k=1}^m$ where $j_k \in m$ is index of the landmark correlation to a feature f_k subject to availability, i.e., $m \leq p$. Using this feature based approach, the tracking problem on the MAV can be written as

$$P(x_t|I_t, \bar{L}) = P(x_t|F_t, J_t, \bar{L}). \tag{2}$$

To do the feature-landmark association, we normally have to project all of the known landmarks L onto the current image I_t, according to the camera model at a predicted pose $\tilde{x}_t$; the predicted pose is obtained by applying an estimated rotation R_{t-1}^t between the frames, i.e., $\tilde{x}_t = R_{t-1}^t x_{t-1}$ where the inter-frame rotation is approximated by integrating the angular rates observed by a gyroscope. However, if we represent the landmarks directly in the image space, i.e., we parametrize the landmarks by $(u_j, v_j, Q_j)_t$ where $(u_j, v_j)_t$ is the projection p_j^t into image co-ordinates, and Q_j is the inverse distance [1], at frame t, we have the local map $\bar{L}_{t-1}$, and do not need to perform any projections. We are able to save computations by having the ground station do the initial projection for the MAV and creating the local map in the image space. The ground station also discards landmarks that are out of the view so that the matching on the MAV uses only a small local map $\bar{L} \subset L$ immediately usable in the near future. Note that at the end of the iteration, we do need to update the local map $\bar{L}_{t-1}$ to $\bar{L}_t$ to represent the local map in the image plane of the latest pose x_t.

Now, for every landmark l_j we have a predicted projection p_j^t on the current image I_t, so we can form a feature association j_j^t by enforcing a maximum distance in the image space, i.e., $\|p_j^t - f_k^t\| <$ maxdist, as well as matching each descriptor d_j^t with descriptor d_k^t of the feature f_k^t in the current image. To evaluate a descriptor match, we warp the 8 by 8 patch descriptor to the closest view using the predicted pose $\tilde{x}_t$ [1] and perform a sum of square differences (SSD).

In our implementation, we perform FAST [18], [19] corner extraction at 4 pyramid levels and store the features in a grid at each level. We use the grids to reduce the number of potential matches before comparing the actual descriptors or enforcing the maximum feature distance in the image space.

B. Forward Projection

As we allow the MAV to be autonomous, by the time it receives the local map $\bar{L}$, it could have proceeded a few frames further. The MAV can linearly project this received map into its current view by composing the kinematic chain it estimated while using the previous local map. This forward projection frees the MAV from time critically depending on the updates from the ground station and reduce the risk of crashing with temporary losses in the communication channel. We also do this forward projection after every iteration to represent the map in the latest frame, i.e., we update the projection p_j^{t-1} to p_j^t.

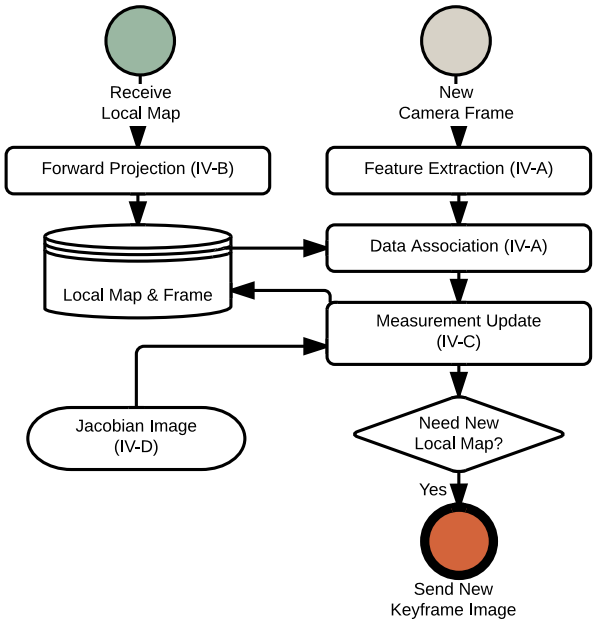

Fig. 2: Flow diagram of operations done on the MAV.

Let the current frame of the MAV be at the time step t and $\bar{t}(\leq t-1)$ be the time step where the initial projections were done for the local map $\bar{L}_{\bar{t}}$ from the ground station. If we let the landmark predictions being used for the current frame be the estimates made in the previous iteration, to bring the local map $\bar{L}_{\bar{t}}$ into a map $\bar{L}_{t-1}$ we have to project it through the kinematic chain $\xi_{\bar{t}+1} \dots \xi_{t-1}$, where each $\xi_i \in \mathbb{R}^6$ is the motion between the frame at time $t = i-1$ and $t = i$.

We can linearly do this forward projection, using the projection Jacobian $H_j = \frac{\partial h(x_t, l_j)}{\partial x_t}|_{x_t = x_{t-1}}$,

$$h(x_{\bar{t}}, l_j) \approx h(\tilde{x}_{\bar{t}-1}, l_j) + H_j(\xi_{\bar{t}+1} \dots \xi_{t-1}), \tag{3}$$

where h is the measurement function, x_t the state at t, l_j the landmark position. Since our landmarks are directly represented in the image space, i.e., $\bar{l}_j = (p_j, Q_j)$, we do not require a measurement function:

$$p_j \approx p_j^{\bar{t}-1} + H_j(\xi_{\bar{t}+1} \dots \xi_{t-1}). \tag{4}$$

This projection is fast due to a direct look-up of the Jacobian as discussed in a later section. In addition, if more computation is available, Runge-Kutta 4 can be used in a similar fashion to Eq. 4 with the fast Jacobian look-up.

[1] In our image space tracking, we do not update the inverse distance Q_j. This update is not crucial since the inverse distance does not change significantly between a few frames. We choose the inverse distance representation over the inverse depth counterpart for this reason.

C. Measurement Update

Given the deterministic data association J that maps each observed landmark l_j to a feature f_k^t, i.e., forms associated landmarks l_{jk}, we can calculate the pose x_t by maximizing the posterior probability

$$P(x_t|F, J, \bar{L}) \propto P(x_t|I_{t-1}, \bar{L}) \prod_k P(f_k^t|x_t, l_{jk}), \quad (5)$$

where we applied Bayes' law. Assuming a Gaussian prior on the pose x_t and Gaussian measurement noise, where $\mathcal{R}$ is the covariance matrix, this is equivalent to minimizing the negative log-likelihood

$$\arg\min_{x_t} \parallel x_t - \tilde{x}_t \parallel_\Sigma^2 + \sum_{j=1}^{K} \parallel z_k^t - h(x_t, l_{jk}) \parallel_\mathcal{R}^2, \quad (6)$$

where $h(x_t, l_{jk})$ is the measurement function, and z_k^t is the measurement of a landmark, i.e., associated feature location.

We parametrize the current pose x_t as an incremental change with respect to $\tilde{x}_{t-1}$, defined by $x_t = \exp(\hat{\xi}) \oplus \tilde{x}_{t-1}$ [20] where $\oplus$ denotes pose composition in $SE(3)$, $\hat{\xi} \in \mathfrak{g}$ represents the Lie algebra element corresponding to the vector $\xi \in \mathbb{R}^6$, and the exp operator, $\exp : \mathfrak{se}(3) \to SE(3)$ maps an incremental twist $\hat{\xi}$ in the Lie algebra $\mathfrak{se}(3)$ to its corresponding pose in the Lie group $SE(3)$. Hence, with linear approximation, the observation model becomes

$$h(x_t, l_j) = h(\exp(\hat{\xi}) \oplus \tilde{x}_{t-1}, l_j) \approx h(\tilde{x}_{t-1}, l_j) + H_j \xi, \quad (7)$$

where H_j is the 2×6 Jacobian matrix defined by:

$$H_j = \frac{\partial h(x_t, l_j)}{\partial x_t}|_{x_t = x_{t-1}} = \frac{\partial h(\exp(\hat{\xi}) \oplus \tilde{x}_{t-1}, l_j)}{\partial \xi}|_{\xi=0}. \quad (8)$$

In the MIST framework, we represent the landmarks in image space, i.e., $\bar{l}_j = (p_j, Q_j)$ for constant Q_j, so that we do not require the measurement function h. Therefore, our tracking problem can be posed as a *linear* optimization:

$$x_t^* = \arg\min_{\xi} \parallel x_t - \tilde{x}_t \parallel_\Sigma^2 + \sum_{j=1}^{K} \parallel (z_k - p_j - H_j \xi) \parallel_\mathcal{R}^2 . \quad (9)$$

Since there is no perfect data association in practice, we iterate the computation a few times by re-weighting $\mathcal{R}$ based on the residuals. This iteratively re-weighted least squares (IRLS) framework [21] serves two purposes: 1) it reduces the effect of outliers and 2) the final weights can be used to evaluate the quality of pose tracking. This quality assessment along with the percentage of successful data associations J between the current frame and the local map are used to judge whether a new frame should be sent to the ground station for a subsequent local map update.

D. Jacobian Image

The Jacobian matrix H_j used in the measurement update has a special structure identifiable as

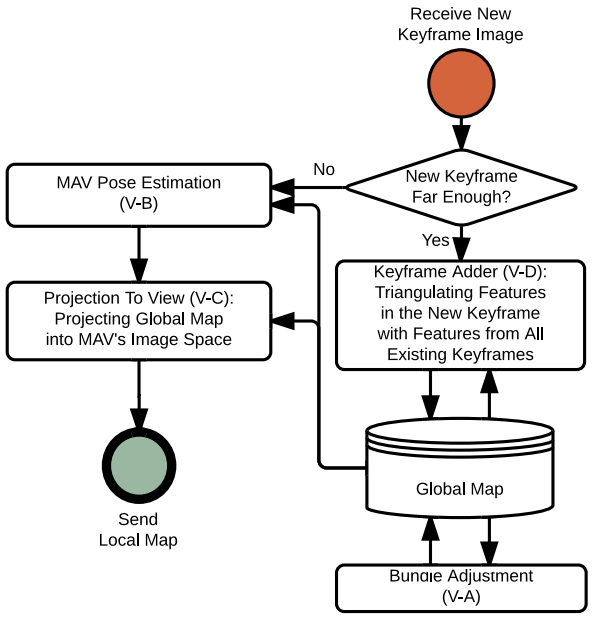

Fig. 3: Framework showing the ground's operations in relation to the MAV's.

$$H_j = \left[\frac{\partial I}{\partial p}\right] \begin{bmatrix} Q & 0 & -QU & -UV & 1+U^2 & -V \\ 0 & Q & -QV & -1-V^2 & UV & U \end{bmatrix}. \quad (10)$$

The first three columns of H_j correspond to translation parameters and the second three columns to the rotation parameters of ξ, where $I = (u, v)$ are the pixel coordinates and $p = (U, V)$ are normalized camera coordinates $(x/z, y/z)$. Thus, the first term, $\frac{\partial I}{\partial p} = \begin{bmatrix} \frac{\partial u}{\partial U} & \frac{\partial u}{\partial V} \\ \frac{\partial v}{\partial U} & \frac{\partial v}{\partial V} \end{bmatrix}$, depends on the calibration model of the camera.

Now, in Eq. 10, if we divide the first 3 columns of H_j by Q, the result only depends on the pixel location (where U, V are functions of u, v). Therefore, we can pre-calculate this matrix at every pixel location and store them in an image of 2×6 matrices that can be used at run-time to reconstitute the Jacobian quickly from a pixel coordinate p_j^t for any landmark by retrieving the 2×6 matrix at the location and multiplying the first 3 columns by Q_j^t. Note that the Jacobian depends on changing u, v. We track in image space but only keep the constant inverse distance for the purpose of retrieving the image space Jacobian.

V. Computation on the Ground

A computationally powerful ground station is used to build and maintain a global map while occasionally providing a local map to the MAV for the short-term tracking. We describe the process (illustrated in Fig. 3) in this section.

A. Bundle Adjustment

The ground station computes a global map of landmarks L and keyframe camera poses $\bar{X}$, given visual measurements

by recovering the *maximum a posteriori* (MAP) estimate

$$\bar{X}^*, L^* = \underset{\bar{X},L}{\text{argmax}}\, P(\bar{X}, L|Z)$$
$$= \underset{\bar{X},L}{\text{argmax}} \prod_i P(x_i) \prod_{i,j} P(z_j^i|\, x_i, l_j). \quad (11)$$

This map-building problem can be posed as inference on a factor graph [15]. The variable nodes are camera poses x_i and the landmarks l_j while the factor nodes are the prior densities $P(x_i)$ on the variable nodes, and the measurement likelihoods $P(z_j^i|\, x_i, l_j)$ constraining a pose x_i and a landmark l_j, given the corresponding visual measurement z_j^i. This measurement likelihood is equivalent to the observation model used on the MAV, described in Eq. 6. By eliminating the factor graph, we can solve for all the camera poses and the landmarks. We omit the details of this process, since we use standard inference techniques for the camera poses and landmarks in the world co-ordinate frame, rather than inference in the image frame as we do on-board the vehicle.

B. MAV Pose Estimation

Parallel to the process of bundle adjustment, the ground station periodically receives images from the MAV. We perform a similar pose estimation done on the MAV, in the standard world co-ordinate frame to estimate the pose at the image frame. This pose optimization is the well-known camera re-sectioning problem, i.e., computing the optimal camera pose x_t^* given measurements z_j^t of *known* landmarks:

$$x_t^* = \underset{x_t}{\text{argmax}}\, P(x_t|\, \{z_j^t, l_j\}_{j=1..m})$$
$$= \underset{x_t}{\text{argmax}}\, P(x_t) \prod_j P(z_j^t|\, x_t, l_j), \quad (12)$$

where we use the MAV's pose estimate as a prior $P(x_t)$.

One thing to note is that the MAV and the ground station can use different corner features and feature descriptors since the pose tracking on the MAV is repeated on the ground station. Accurate but computationally expensive methods such as SIFT [22] can be used on the ground station in place of the lightweight algorithms on the MAV.

C. Projection to View

The ground station can apply the MAV's camera model to obtain estimated image co-ordinates p_k in MAV's image space for each probably-visible landmark and communicate these rather than the metric locations. Furthermore, at any given time step, the MAV only needs to know about the landmarks that it is likely to observe, and so the ground station only needs to transmit a map of these landmarks.

Thus, using the optimized MAV pose, we first project the landmarks L into the calibrated coordinates (U_i, V_i);

$$(U_i, V_i) = K[R_w^c|t_w^c]l_i^w, \quad (13)$$

and then project to in-image *pixel coordinates* $(u_i, v_i) = f(U_i, V_i)$ using f the fish-eye lens model [23]. Out of these projections $p_i = (u_i, v_i)$ in pixel coordinates, the ones matched with corner features within the image boundaries are included as a local map and transmitted to the MAV.

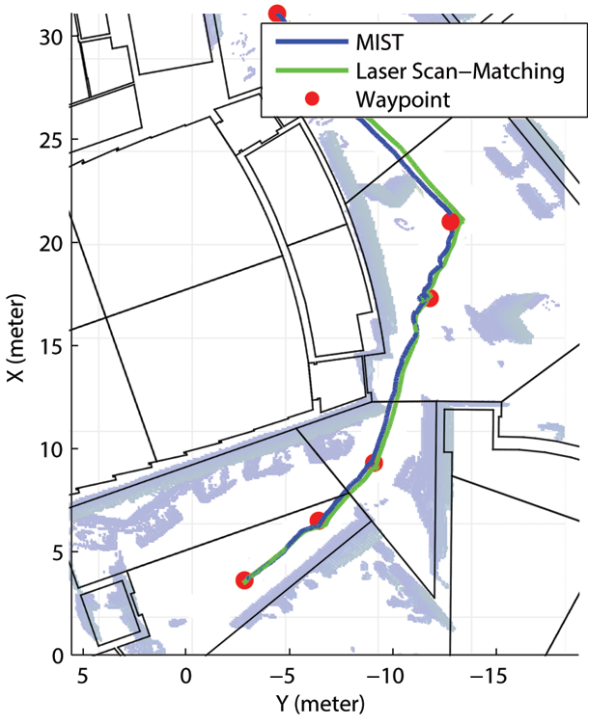

Fig. 4: Way-points were manually selected to generate the trajectory. MIST was used as the primary pose estimator while laser scan-matching was performed in parallel to provide a ground-truth trajectory and a pseudo-scale input for alignment in the map for planning purposes. The occupancy grid built with the laser scan-matching is displayed in light blue and overlaid with the building floor plan.

D. Keyframe Adder

The two initial frames are created in a separate initialization process using homography with a locally planar assumption as done in [1]. During the initialization stage, all of the frames are transmitted from the MAV to track a trail of features on the ground station. After initialization, the ground station waits for the MAV to send a new frame, while optimizing the global map in parallel. Once a new frame is received, the ground station prepares a new local map in the image plane of the frame and sends it back to the MAV. Then the frame's distance to all the keyframes in the global map are calculated to judge whether it is far enough to other keyframes to qualify as a new keyframe. The ground station then searches through all of the known keyframes to make data associations with the newly received frame. This dense association stage is critical to building a globally salient map.

VI. Experiments

We autonomously flew an Ascending Technologies Pelican quad-rotor, shown in Fig. 5, in an unknown indoor environment using a 30 fps PointGrey Chameleon camera (640x480 images), a Microstrain IMU, a Hokuyo laser scanner, and

a Gigabyte dual-core i7[2] to evaluate MIST as a visual pose estimation module on a MAV. We then analyzed the accuracy of our pose estimates by comparing them to the pose estimates generated using PTAM. We also compared the time to pose estimates for MIST, PTAM running on-board, PTAM running on-board with mapping process turned off, and PTAM running off-board with image streaming. The data-set used for this benchmarking was collected by carrying the quad-rotor around an indoor environment, and saving camera images and other sensor data using LCM [24]. We also show performance from flight data using the heading to control the vehicle. The saved data was played back at the original intervals on the quad-rotor, to simulate the MAV flying while providing the exact same input to different algorithms used in comparison. For the ground-station, we used a quad-core i7 laptop.

A. Autonomous Flight using MIST

During the autonomous flight, we ran a laser-scan matching algorithm [2] in parallel. The pose estimates and the occupancy grid from the laser scan-matcher were treated as ground truth; we obtained metric pseudo-scale input from the pose estimates, and used the occupancy grid to plan a collision-free trajectory as shown in Fig. 4. The use of the laser-scanner was for these purposes only, and our algorithm does not require the laser-scanner to estimate its pose.

We formed the flight trajectory by selecting way-points and using a polynomial trajectory generator [25] to smoothly connect them within the laser-built occupancy grid. We controlled the quad-rotor using a nonlinear controller [26] and increased the frequency of the pose estimation by relying on an EKF to fuse our MIST estimates with a 100 Hz IMU.

In order to align the laser-based poses with our MIST pose estimates, we transformed the MIST pose updates in the camera frame to updates in the robot body frame, and continuously composed to an initial laser pose estimate. Then to align the MIST poses with the laser-built map for planning purposes, MIST poses and the laser-based poses were collected between two consecutive local map updates and the difference in translation was used as a pseudo-scale input to re-scale the map for the following frames.

As shown in Fig. 4 and 6, while the MAV could reliably follow the trajectory, continuously composing the MIST pose updates on a single initial laser pose resulted in the trajectory drifting away from the laser-estimated trajectory as errors accumulate. One source of systematic error between the vision-based estimates and the laser-based estimates is the approximate transformation between the camera and the laser frame, caused by an approximate hand-alignment of the camera, the IMU, and the laser on the quad-rotor. Another source of approximation error is the linear interpolation performed on the laser pose estimates when finding the pose chronologically closest to a camera timestamp. A final source of error is the heavy dependence on the pseudo-scale inputs, which would corrupt the translation if estimated poorly.

[2]Not a low-cost PC but allowed us to benchmark to the heavier load of PTAM, and support laser scan-matching in parallel during experiments.

(a) Ascending Technologies Pelican quadrotor

(b) Camera frame and tracked features

Fig. 5: The vehicle and the camera used for the flight experiment are shown. The MAV uses the monocular camera and tracks landmarks in the image space to estimate its pose. Video of the flight is available at `https://www.youtube.com/watch?v=VWWvjSHZCNo`

B. MIST Tracking Accuracy

To evaluate the accuracy of MIST in the presence of external systematic errors, we collected camera images and sensor data and played them back as an identical input to PTAM and MIST. To demonstrated the ability to come back to a known location and update the global map, we selected a 50 meter trajectory with loops as shown in Fig. 6. Our pose estimates were approximately as good as the trajectory estimated using PTAM.

We compared the error in rotation and translation of frame-to-frame updates in the robot body frame for PTAM and MIST, with the laser scan-matching as the ground-truth. It can be seen in Fig. 7 that our system performed similar to PTAM at an average absolute rotation error of 0.34 degrees and an average absolute translation error of 2.97 cm. In rotation, PTAM performed similarly while in translation the error for PTAM was slightly higher at 4.51 cm.

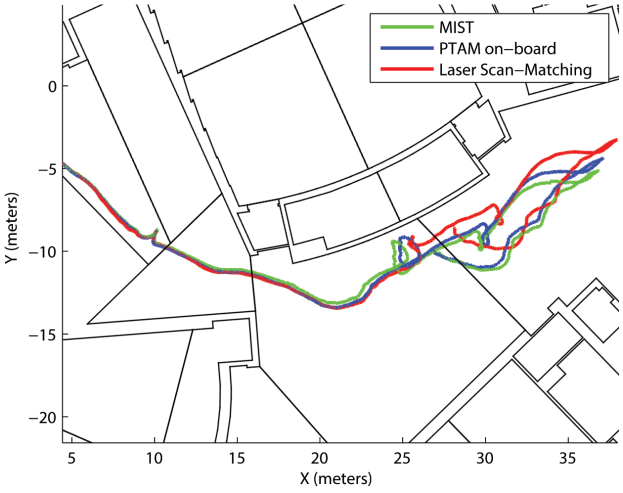

Fig. 6: The laser-based trajectory estimate in red, MIST pose estimates in green and PTAM estimates in blue. During the total distance of approximately 50 meters, the vision-based trajectory drifts away from the laser estimates. However, our method is approximately as accurate as PTAM.

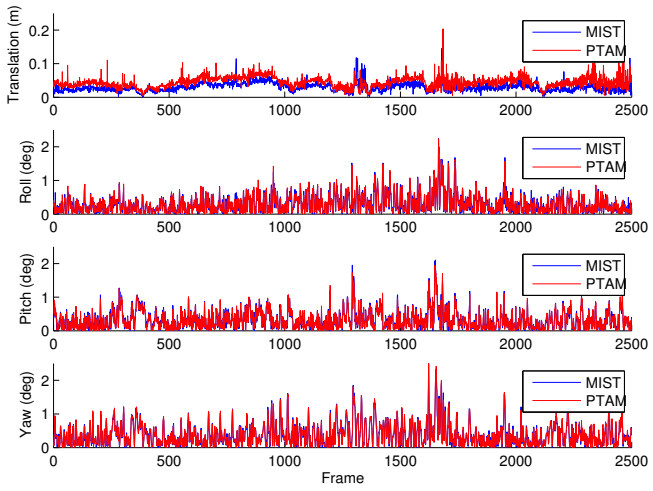

Fig. 7: Errors in translation and rotation for PTAM and MIST, compared to the laser scan-matching counterpart. It can be seen that MIST is on par with the accuracy of PTAM.

C. Timing Comparisons

We compare the time to pose estimates between MIST, PTAM running on-board, PTAM with raw streamed images, and PTAM with JPEG-compressed streamed images. Pre-recorded data were played back on the MAV to simulate the vehicle flying while providing identical sensor data to different methods. We did not perform this analysis in flight since the loss of a pose estimate due to network latency would cause a loss of control. In the case of streaming methods, the time to pose estimate includes the round-trip transmission time over Wi-Fi as well as the pose computation

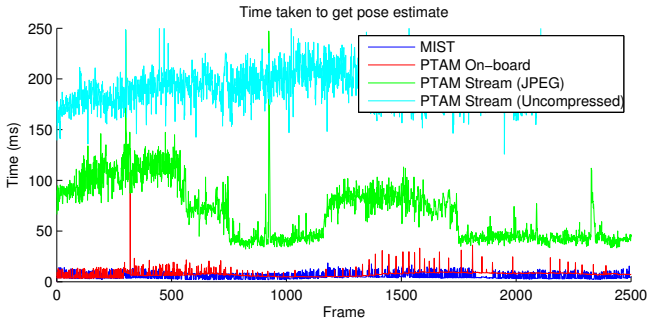

Fig. 8: Comparing the time to pose estimate for MIST, PTAM running on-board, PTAM with streamed images. It can be observed that it was infeasible to track using raw streaming method, while the JPEG-compressed images still take more than a full frame to arrive at the MAV.

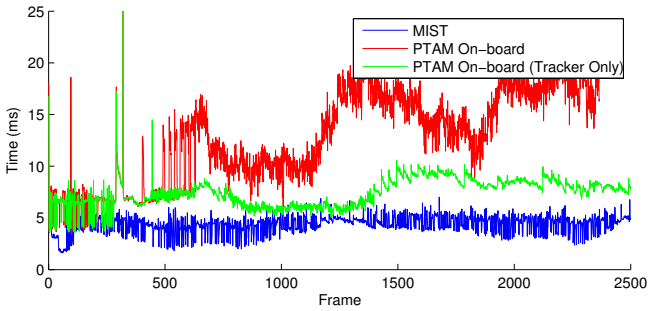

Fig. 9: A single-core computationally limited system was emulated to better highlight the difference between PTAM running on-board, PTAM tracker running on-board, and MIST. It can be observed that the computation time for MIST remained constant, while PTAM gradually took longer due to the mapping process. Comparing against only the tracking process in PTAM, MIST was nearly twice as fast.

time on the ground station. For MIST and PTAM running on-board, this time is only the computation time taken since the camera image was available.

As shown in Fig. 8, sending uncompressed images took 191.17 ms on average, causing the pose estimates to arrive 8 frames later. On the other hand, streaming JPEG images was relative fast, with the estimates lagging only two frames behind. While the transmission delay may seem manageable on slow moving vehicles, the inconsistency (over twice at times) and the potential danger of dropped packets could cause the vehicle to crash at any point in time.

It can be observed that the time taken for PTAM and MIST was not very different. The reason is that on the dual-core i7, the computation done by PTAM's mapping process was parallel to the tracking thread, and did not add to the computation time to pose estimate. While this shows the strength of the parallel design, we emulated a more computationally limited platform typically found on low-cost MAVs by enabling only a single core. We also quadrupled

	Approach	Time to pose estimate (ms)
Single-core, 4x speed	MIST	4.3858
	PTAM On-board (Tracker)	7.6195
	PTAM On-board	12.7133
Dual-core, 1x speed	PTAM Stream (JPEG)	62.9230
	PTAM Stream (Uncomp.)	191.1747

	Roll (deg)	Pitch (deg)	Yaw (deg)	Translation (m)
MIST	0.2979	0.3266	0.3896	0.0297
PTAM	0.3037	0.3209	0.3896	0.0451

TABLE I: Computation time and mean absolute error in incremental updates for MIST and PTAM-based methods.

the playback speed of the sensor data and produced camera images at 120 Hz, and IMU outputs at 400 Hz.

As shown in Figure 9, on an emulated single-core machine, MIST still retained a constant time to pose estimates. However, for PTAM running on-board, as the processor jumped from the tracking process to the mapping process, the time to pose estimate grew with the growing map, due to increasing difficulty in bundle adjustment. We also disabled the entire mapping process of PTAM, and ran only its tracker to compare against MIST. The average computation time for MIST was 4.39 ms, capable of estimating the pose at over 200 Hz, while the average for PTAM tracker was 12.71 ms. It can be seen that the computation time for PTAM tracker also grew slowly since it had to project the growing map into its measurement space. A summary of the computation time and the average error in frame-to-frame pose updates for the algorithms are shown in Table I.

VII. Conclusion

We have divided the full SLAM problem into a fast monocular image space tracking (MIST) on the MAV and a keyframe-based smoothing and mapping on the ground station. Using our fully decoupled tracker and mapper design and fast image space tracking, we are able to compute the pose estimates on the MAV in constant time at 4.39 ms while building the growing global map on the ground station. The quality of this global map is as accurate as PTAM when compared to the laser scan-matcher ground-truth.

For future work, we can explore utilizing MIST in a multi-robot scenario where a team of disposable MAVs cooperate to explore a post-disaster scene. Having a single ground-station supporting multiple low-cost MAVs while building a single globally consistent map may be a trivial solution to creating a centralized multi-robot system.

VIII. Acknowledgement

This research was funded by the ONR under MURI N00014-09-1-0641 and the ARO MAST CTA. Their support is gratefully acknowledged.

References

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225–234, 2007.

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

[3] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 20–25, 2011.

[4] 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*, Citeseer, 2013.

[5] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor MAV," in *Intelligent Robots and Systems (IROS), 2012 IEEE International Conference on*, pp. 4557–4564, 2012.

[6] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM–based navigation for autonomous micro helicopters in GPS-denied environments," vol. 28, no. 6, pp. 854–874, 2011.

[7] 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.

[8] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, 2014.

[9] 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.

[10] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems*, 2014.

[11] D.-N. Ta, K. Ok, and F. Dellaert, "Vistas and parallel tracking and mapping with wall–floor features: Enabling autonomous flight in man-made environments," *Robotics and Autonomous Systems*, 2014.

[12] K. Ni, D. Steedly, and F. Dellaert, "Tectonic SAM: Exact, out-of-core, submap-based SLAM," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1678–1685, IEEE, 2007.

[13] U. Frese, "Treemap: An O (log n) algorithm for indoor simultaneous localization and mapping," *Autonomous Robots*, vol. 21, no. 2, pp. 103–122, 2006.

[14] S. B. Williams, G. Dissanayake, and H. Durrant-Whyte, "An efficient approach to the simultaneous localisation and mapping problem," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1, pp. 406–411, IEEE, 2002.

[15] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[16] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2564–2571, IEEE, 2011.

[17] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: binary robust independent elementary features," in *Computer Vision–ECCV 2010*, pp. 778–792, Springer, 2010.

[18] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*, pp. 430–443, 2006.

[19] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 1, pp. 105–119, 2010.

[20] J. Humphreys, *Introduction to Lie algebras and representation theory*, vol. 9. Springer, 1972.

[21] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics-Theory and Methods*, vol. 6, no. 9, pp. 813–827, 1977.

[22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[23] F. Devernay and O. Faugeras, "Straight lines have to be straight," *Machine vision and applications*, vol. 13, no. 1, pp. 14–24, 2001.

[24] A. S. Huang, E. Olson, and D. C. Moore, "LCM: Lightweight communications and marshalling," in *Intelligent robots and systems (IROS), 2010 IEEE/RSJ international conference on*, pp. 4057–4062, IEEE, 2010.

[25] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for quadrotor flight," in *International Conference on Robotics and Automation*, 2013.

[26] T. Lee, M. Leok, and N. H. McClamroch, "Control of complex maneuvers for a quadrotor UAV using geometric methods on SE (3)," *arXiv preprint arXiv:1003.2005*, 2010.