There's No Place Like Home: Visual Teach and Repeat for Emergency Return of Multirotor UAVs During GPS Failure

Michael Warren, Melissa Greeff, Bhavit Patel, Jack Collier, Angela P. Schoellig, and Timothy D. Barfoot

***Abstract*—Redundant navigation systems are critical for safe operation of UAVs in high-risk environments. Since most commercial UAVs almost wholly rely on GPS, jamming, interference, and multi-pathing are real concerns that usually limit their operations to low-risk environments and visual line-of-sight. This letter presents a vision-based route-following system for the autonomous, safe return of UAVs under primary navigation failure such as GPS jamming. Using a Visual Teach and Repeat framework to build a visual map of the environment during an outbound flight, we show the autonomous return of the UAV by visually localizing the live view to this map when a simulated GPS failure occurs, controlling the vehicle to follow the safe outbound path back to the launch point. Using gimbal-stabilized stereo vision and inertial sensing alone, without reliance on external infrastructure, Visual Odometry and localization are achieved at altitudes of 5–25 m and flight speeds up to 55 km/h. We examine the performance of the visual localization algorithm under a variety of conditions and also demonstrate closed-loop autonomy along a complicated 450 m path.**

***Index Terms*—Aerial systems: Perception and autonomy, visual-based navigation, sensor-based control.**

I. Introduction

SAFE beyond Visual Line-Of-Sight (VLOS) operations are critical to enhancing the utility of Unmanned Aerial Vehicles (UAVs) in large-scale, outdoor operations. Typically, reliance on Global Navigation Satellite Systems (GNSS) for navigation in most low-cost commercial UAVs mean the authorisation to do so from government regulators is rare. Jamming, interference and accuracy concerns mean that Global Positioning System (GPS) alone cannot be relied on in cases of close-proximity, safety-critical or high-value operations. In this letter, we present a complete vision-only route-following system for the autonomous navigation of UAVs, and demonstrate its use as a functional backup system for GPS-only navigation. Using this system allows the vehicle to navigate home visually in case of primary navigation system failure, without reliance on any external infrastructure, or inertial sensing for the vision-based components.

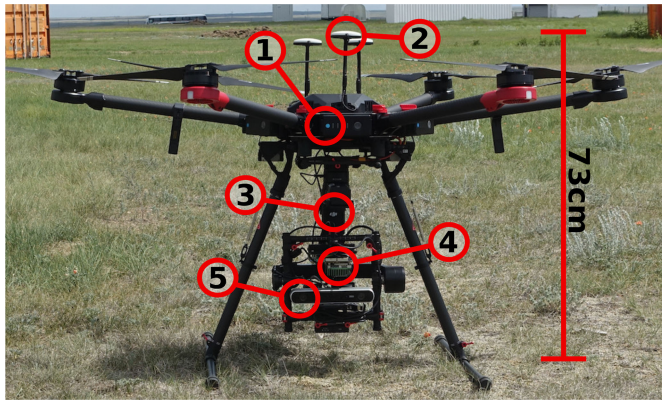

Fig. 1. The experimental setup for VT&R on our multirotor UAV: (1) DJI Matrice 600 Pro vehicle platform, (2) DJI A3 triple-redundant GPS module, (3) DJI Ronin-MX 3-axis gimbal, (4) NVIDIA Tegra TX2, (5) StereoLabs ZED camera.

Visual Teach and Repeat (VT&R) is a path-following algorithm capable of autonomously driving a robot by following a previously traversed route [1]. Using visual feature matches from a live view to a locally metric map of 3D points allows the robot to estimate a path offset and send corrections to a path-following controller [2]. Traditionally, VT&R is used on wheeled vehicles [3], with applications over constrained paths where external navigation infrastructure is unreliable or not available, e.g., factory floors, orchards, mines, urban road networks, and exploratory search-and-return missions. Using VT&R on aerial platforms has a number of unique use cases: just-in-time deliveries between warehouses, where flight paths are generally restricted to a few, high-frequency routes; monitoring of sensitive assets such as property borders or high-value infrastructure; and autonomous patrol in close-proximity environments, where poor sky view and jamming are notable concerns. Significantly, we want the vehicle to be able to

Manuscript received September 10, 2018; accepted November 5, 2018. Date of publication November 26, 2018; date of current version December 10, 2018. This letter was recommended for publication by Associate Editor E. Johnson and Editor J. Roberts upon evaluation of the reviewers' comments. This work was supported in part by the Smart Computing for Innovation Consortium (SOSCIP), in part by Defence Research and Development Canada (DRDC), in part by Drone Delivery Canada (DDC), in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and in part by the Centre for Aerial Robotics Research and Education (CARRE), University of Toronto. *(Corresponding author: Michael Warren.)*

M. Warren, M. Greeff, B. Patel, A. P. Schoellig, and T. D. Barfoot are with the University of Toronto Institute for Aerospace Studies, Toronto, ON M3H 5T6, Canada (e-mail: michaelwarren@robotics.utias.utoronto.ca; melissa.greeff@robotics.utias.utoronto.ca; bhavit.patel@robotics.utias.utoronto.ca; angela.schoellig@utoronto.ca; tim.barfoot@utoronto.ca).

J. Collier is with the Defence Research and Development Canada – Suffield Research Centre, Medicine Hat, AB T1A 8K6, Canada (e-mail: jack.collier@drdc-rddc.gc.ca).

Digital Object Identifier 10.1109/LRA.2018.2883408

autonomously and safely return to the take-off location at any time by using vision to localise to a map generated during the outbound path, all during a single flight.

In this letter, we adapt the traditional VT&R methodology to suit these target use cases and apply our VT&R 2.0 system [3] on-board a multirotor UAV (Fig. 1) to demonstrate closed-loop operation. We show results of live localisation at speeds up to 15 m/s (55 km/h) at low altitude (5–25 metres) in winds up to 8 m/s, and demonstrate vision-based path-following control for the return segment of a just-taught outbound path. The novel work of this letter includes 1) demonstration of the VT&R framework on a new platform, a UAV with gimbal-stabilised camera, 2) a thorough analysis of localisation performance in this new scenario and 3) presentation of a path-following controller for multirotor UAVs applied specifically in a VT&R framework, all in a wide range of outdoor test scenarios.

The rest of this letter is outlined as follows: Section II examines similar work in visual route following for ground vehicles and UAVs, and explores recent work in autonomous vision-based navigation of UAVs. Section III describes the VT&R methodology for application on our target UAV, including the VT&R framework, localisation algorithm and gimbal and vehicle controllers. Section IV describes the experimental setup to test the airborne VT&R framework, as well as description of datasets, field tests and results. The letter is concluded in Section V.

II. Previous Work

VT&R and similar route-based navigation algorithms have a rich history on ground platforms [1], [2], [4], [5], with the most recent extension adapted to include multiple experiences, increasing the autonomous performance time from a few days to several months [3]. On UAVs, there are now several demonstrations of teach-and-repeat style algorithms from the authors of this letter and others [6]–[9].

Our previous work, demonstrating the localisation performance of VT&R on fixed-wing UAVs [7] and integration of a gimbal-stabilised camera on a ground vehicle [10], is the lead-up to this work. While there are few examples using a gimbal-stabilised camera on ground vehicles, a number of examples exist in demonstrations on UAVs [11]–[15]. This discrepency can most likely be attributed to the larger dynamic motions of UAVs, where the utility of a gimbal is highly justified to ensure smooth sensor motion. In all the above cases, however, only two-axis gimbals are utilised. In our setup, we use an off-the-shelf three-axis gimbal to attenuate motion in all three rotational axes.

The approach that is closest conceptually to our work, with specific application on UAVs, is [9]. Despite not being framed as a 'teach-and-repeat' technique, this system presents a demonstration of such a method on a UAV using a visual-inertial framework with weak GPS priors to assist initialisation of localisation and inform loop closures. Our work differs in that it requires no offline map building (the map is built on-board in real time) and does not require inertial sensors or external infrastructure such as GPS for the perception component of the system.

Beyond the VT&R paradigm, there is a rich demonstration of vision-based navigation on UAVs in recent years [16]. While most older demonstrations incorporate stereo camera systems for scale, they suffer from poor (i.e., small) baseline-to-depth ratios at higher altitudes. Recent advances in Inertial Measurement Unit (IMU) technology have allowed the use of loosely [17]–[19] and tightly coupled [20]–[22] visual-inertial systems using both monocular and stereo cameras [23], and with impressive demonstrations of dynamic manoeuvres at high speed [24] in indoor, small-scale setups. The majority of large scale demonstrations using these systems, however, often exist as a full Simultaneous Localisation and Mapping (SLAM) framework [25], [26], incorporating exploration and globally metric 3D maps as a method of accurate survey. In contrast, VT&R takes a locally metric approach for map building, and leverages a human operator for the initial 'demonstration' task, circumventing the difficult tasks of autonomous exploration and loop closures.

III. Methodology

In this letter, we use our well-established VT&R 2.0 software system as presented in [3], including the extension to a gimbal-stabilised camera [10]. However, we adapt this system for use on a multirotor UAV specifically for the purposes of emergency return. Instead of *teach* and *repeat* phases, we implement functionally similar *learn* and *return* phases.

In the *learn* phase, the UAV flies using autonomous GPS waypoint following or human operator control. During this phase, the VT&R algorithm performs passive Visual Odometry (VO), inserting the visual observations from this 'privileged' experience into a relative map of pose and scene structure, effectively 'learning' the route. Following a primary navigation systems failure, the UAV should enter the *return* phase, and, without reliance on GPS or other external sensing, autonomously re-follow the route home in the reverse direction. In addition to performing the same VO as in *learn*, it performs a localisation using a local segment from the learnt path. The vehicle follows the learnt path by sending high-frequency localisation updates (relative position and orientation with respect to the map) to a path-following controller. Once the vehicle returns to the start point, it hovers until taken over by a human controller. To be clear, in this letter we only simulate GPS failures by manually commanding the vehicle to enter the return phase during flight.

In the following sections, we describe our VT&R system, including the architecture of the system, the visual navigation algorithm, and gimbal and path-following controllers.

A. System Overview

The architecture of the VT&R system for the multirotor UAV is shown in Fig. 2. All processing, including visual navigation, localisation, planning and control occurs on-board the UAV on the primary computer (Fig. 1). This computer directly interfaces with the on-board camera via Universal Serial Bus (USB) 3.0, which provides greyscale stereo images for visual navigation. This computer also interfaces with the on-board autopilot via a serial Transistor-Transistor Logic (TTL) connection, which

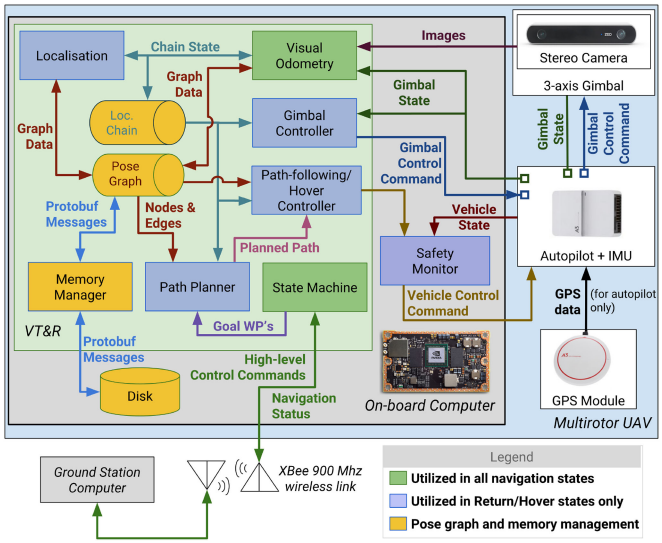

Fig. 2. The architecture of the VT&R system for multirotor UAVs.

provides vehicle data (gimbal state, autopilot state, etc.) and the interface for sending control commands. A long-range, low-bandwidth XBee 900 Mhz wireless link is used to communicate with the primary on-board computer from a ground station. The ground station computer is utilised only for status monitoring and sending of high-level control commands. These commands consist of manual state transition requests (switching from *learn* to *return*), obtaining flight control authority from the autopilot, and initiating GPS waypoint missions.

The VT&R software system consists of several interacting components (Fig. 2): 1) VO (including separate feature extraction, pose estimation and windowed refinement threads) 2) visual localisation, 3) a state machine, 4) a path planner, 5) gimbal and path-following controllers and 6) a safety monitor. Each system operates in a separate thread or process, interacting through the transfer of data caches (a packet of new and derived data, including images, processed features and estimated transforms) and through the use of a Google Protobuf back-end for disk storage. Memory managers ensure that stale data is written to disk to reduce Random Access Memory (RAM) utilisation, and is pre-emptively re-loaded during the return phase to ensure localisation can proceed without waiting for disk access. The Robot Operating System (ROS) is used to run the safety monitor and interface to the autopilot and camera. The adapted VT&R state machine for multirotor UAVs controls the high-level state that the system is in (usually *learn* or *return*).

A safety monitor runs as an independent process to ensure safe operation of the vehicle in case of system failure. It performs a sanity check on control and localisation data, in addition to a watchdog functionality on the control commands and state data both from VT&R and the autopilot. Any monitored command or state data that is delayed by more than a preconfigured timeout triggers a safety failure, forcing the vehicle to release autonomous control and revert to manual pilot control.

In the following sections, the visual system, path-following and gimbal controllers are described in more detail.

B. Visual System

The visual system consists of separate threads for feature extraction, pose estimation (VO), refinement and localisation, using images captured by a stereo camera to estimate both pose updates and localisation to the path during the return.

1) Visual Odometry: During both the *learn* and *return* phases, image pairs are captured by a calibrated stereo camera at a frame rate of ∼15 Hz, while the gimbal state (read as roll-, pitch- and yaw-axis angular positions) is captured at 10 Hz. The gimbal state gives the pose of the camera in the vehicle frame by compounding the captured gimbal angles through a series of transforms with known translations extracted from a hand calibration. We denote the vehicle-to-sensor (camera) transform at time τ as $\mathbf{T}_{sv}^{\tau}$.

For each stereo image pair captured at time t, Speeded-Up Robust Features (SURF) features[1] are extracted, descriptors generated and landmarks triangulated. Landmarks are triangulated from both the stereo pair and from motion to account for both close proximity and extremely large depths depending on altitude, similar to [22]. Each feature in this latest frame-pair is matched to the last keyframe via SURF descriptor matching on the Graphics Processing Unit (GPU). The raw matches are then passed through a Maximum Likelihood Estimation SAmple Consensus (MLESAC) robust estimator to find the relative transform to the last keyframe. Finally, this transform is optimised using our Simultaneous Trajectory Estimation And Mapping (STEAM) bundle adjustment engine [27], keeping landmarks fixed.

After this process, if the number of inliers drops below a minimum count or the motion (translation or rotation) exceeds a threshold, the frame is set as a keyframe and the features, new landmarks, and vehicle-to-sensor transform at that time are stored in a vertex in a pose graph for future retrieval. The relative transform is stored as an edge to the previous vertex. Windowed bundle adjustment (termed *windowed refinement*) is then performed on the last 5–10 vertices. This VO plus bundle adjustment process generates a dead-reckoned set of linked poses that represent the path. During the *learn* phase, this set of poses and edges is marked as 'privileged', as they belong to the known, safe path generated from either human or GPS control. Naturally, incremental translational and rotational errors compound during this process, causing the global map to be distorted. However, VT&R depends on the graph being only *locally* metric in the region to which the vehicle is localized. For a more thorough explanation of this component, we direct the reader to our previous work [3].

2) Visual Localisation: During the *return* phase, while the vehicle flies in reverse, an additional thread performs visual matching to the local map of 3D points in the graph to estimate the path-following error (Fig. 3), which is used by a path-following controller to keep the vehicle on the path.

To enable this process, the *localisation chain* is used to keep track of important vertices in the graph and their respective

[1]Alternative feature/descriptor types such as ORB were tested, but were demonstrably inferior in both speed and reliability to SURF for both VO and localisation in our experimentation on this platform.

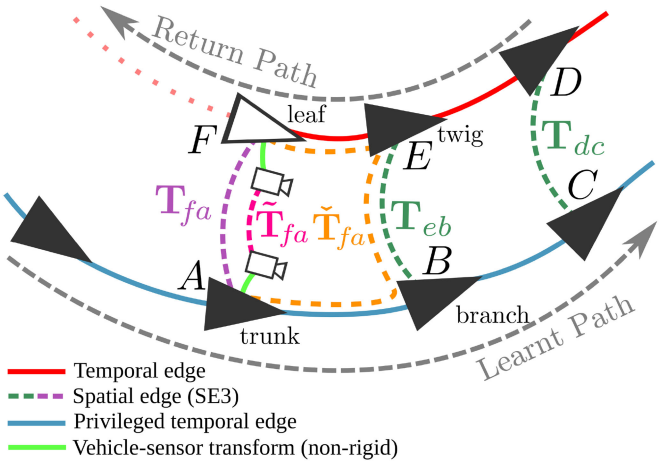

Fig. 3. During the return phase, the vehicle follows the learned route in reverse. The localisation chain updates the estimated localisation transform $\tilde{\mathbf{T}}_{fa}$ at each VO update. Upon creation of a new vertex F, visual localisation inserts the new edge $\mathbf{T}_{fa}$. The gimbal controller minimises orientation error of $\tilde{\mathbf{T}}_{fa}$, which includes vehicle-to-sensor transform, $\mathbf{T}_{sv}$, and $\check{\mathbf{T}}_{fa}$. The uncertainties and some estimated transforms are omitted here for clarity.

transforms. We use a 'tree' model to name vertices in the chain, going from the *trunk* vertex (defined as the closest vertex spatially on the privileged path), through the *branch* (the closest vertex on the privileged path with a successfully MLESAC estimated transform to the current path), *twig* (the corresponding vertex to the *branch* on the current path) and *leaf* (latest live vertex) vertices. These can be seen in Fig. 3. We use the notation $\mathfrak{t}$, $\mathfrak{b}$, $\mathfrak{w}$ and $\mathfrak{l}$ to refer to the *trunk*, *branch*, *twig* and *leaf* vertices, respectively.

At every step of VO (i.e., on every successfully estimated frame, not just keyframes) the localisation chain is updated with the estimated transform from *trunk* to *leaf*, (or $\check{\mathbf{T}}_{\mathfrak{lt}} = \check{\mathbf{T}}_{fa} = \mathbf{T}_{fe}\mathbf{T}_{eb}\mathbf{T}_{ba}$ in Fig. 3). The leaf is updated every step and, if necessary, the *trunk* vertex is updated to the closest estimated privileged vertex to the *leaf*.

Upon insertion of a new VO keyframe as a vertex in the graph, the localisation thread attempts to estimate a new transform from *branch* to *twig*. This process follows four separate stages: i) landmark migration, ii) landmark matching, iii) pose estimation, iv) optimisation. First, the nearest privileged vertex (the *trunk*) is used as the base vertex to generate a local window of privileged vertices that contain potentially matchable landmarks. Using the transforms on the privileged edges, the landmarks in this window are transformed to the *trunk* to generate a locally metric set of 3D points with a common origin.[2]

Following a similar process to VO, features in the latest non-privileged vertex (the *leaf*) are matched using their SURF descriptors to *all* descriptors of the migrated landmarks, which are then passed through a MLESAC robust estimator to estimate the relative transform from *trunk* to *leaf*, $\mathbf{T}_{\mathfrak{lt}}$ ($\mathbf{T}_{fa}$ in Fig. 3). Finally, the transform is optimised while leaving all landmarks fixed. The localisation chain is then updated to reflect this fresh transform estimate, and the new *branch* to *twig* is set $\mathbf{T}^{*}_{\mathfrak{wb}} \leftarrow \mathbf{T}_{fa}$. The path-following controller can query the localisation chain at any time to get the best estimate of $\mathbf{T}_{\mathfrak{lt}}$, facilitating control at high speed even with significant delays from visual localisation.

[2]While our previous work incorporates features and points from multiple experiences (i.e., multiple traverses), the learn-return framework by definition only uses a single experience: the privileged one.

C. Gimbal Controller

Use of a gimbal decouples the visual perspective from the orientation changes of multirotor UAVs due to their underactuated control behaviour. This significantly improves the robustness of VT&R in the air by adding extra degrees of actuation to the visual servoing problem. During fast, dynamic manoeuvres, a gimbal-stabilised camera system will be able to outperform a static camera system by decoupling the aircraft motion from the camera view. In addition, maintaining a consistent roll ensures that generally unstable point features are tracked more consistently.

The gimbal is hand-calibrated to estimate the translations between each actuated axis, and this proves sufficient for most applications. This calibration is used in conjunction with axis encoder values provided by the autopilot to estimate the sensor-to-vehicle transform $\mathbf{T}_{sv}$, i.e., from the camera frame to the vehicle frame through the gimbal. Gimbal encoder data is provided at 10 Hz with an unspecified latency. In practice, the latency is less than 100 ms.

During the *learn* phase, gimbal control is not performed by VT&R, but left open-loop such that the gimbal internal controller performs stabilisation of roll and pitch, and smoothes yaw that follows the vehicle yaw. The value of this sensor-to-vehicle transform $\mathbf{T}^{\tau}_{sv}$ (Fig. 3) is recorded at each new vertex, corresponding to time τ. During the return, the gimbal is actively controlled by VT&R for the pitch and yaw axes. The gimbal is commanded to reduce orientation error between the current (*leaf*) view and nearest privileged (*trunk*) view, ($\tilde{\mathbf{T}}_{fa}$ in Fig. 3), as knowledge of the transform between the current and the privileged poses is known via the localisation chain such that:

$$\tilde{\mathbf{T}}_{fa} = \mathbf{T}^{\mathfrak{l}}_{sv}\mathbf{T}_{\mathfrak{lt}}{\mathbf{T}^{\mathfrak{t}}_{sv}}^{-1} \tag{1}$$

using the sensor-to-vehicle transforms captured at vertices $\mathfrak{t}$ and $\mathfrak{l}$. $\check{\mathbf{T}}_{fa}$ is updated in the localisation chain at every frame.

D. Path-Following Controller

A path-following controller is implemented for vehicle control during the *return* phase to keep the vehicle as close as possible to the outbound path while maintaining a suitable target velocity.

To enhance robustness to environmental disturbances and system delays, we consider a path-following approach, which, in contrast to trajectory tracking, prioritizes spatial error over temporal error [28]. By extending the approach in [28] to a VT&R framework, we achieve simple multirotor path-following. This is done by converting a standard P-D tracking control to select the spatially closest reference point on the path at each control time step (50 Hz). The localization chain provides an estimated transform from *trunk* to *leaf* $\check{\mathbf{T}}_{\mathfrak{lt}}$ from which we can extract the

current position $\mathbf{p}_{\mathrm{t}}^{\mathrm{lt}} = (x, y, z)$ and orientation $\mathbf{C}_{\mathrm{tl}}$ using:

$$\check{\mathbf{T}}_{\mathrm{tl}} = \begin{bmatrix} \mathbf{C}_{\mathrm{tl}} & \mathbf{p}_{\mathrm{t}}^{\mathrm{lt}} \\ \mathbf{0}^T & 1 \end{bmatrix} = \check{\mathbf{T}}_{\mathrm{lt}}^{-1}.$$

We obtain a translational velocity estimate $\mathbf{v}_{\mathrm{t}}^{\mathrm{lt}} = (\dot{x}, \dot{y}, \dot{z})$ using STEAM trajectory generation [27], which fits a constant velocity trajectory through the previous path vertexes.

1) VT&R Path-Following Reference: We generate a path by connecting a straight-line through successive privileged vertices. To do this, we use the localization chain to obtain a transform from the *next* privileged vertex to the *trunk* $\mathbf{T}_{\mathrm{tn}}$. From this we can extract the position $\mathbf{p}_{\mathrm{t}}^{\mathrm{nt}}$ of the *next* privileged vertex with respect to the *trunk* using

$$\mathbf{T}_{\mathrm{tn}} = \begin{bmatrix} \mathbf{C}_{\mathrm{tn}} & \mathbf{p}_{\mathrm{t}}^{\mathrm{nt}} \\ \mathbf{0}^T & 1 \end{bmatrix}.$$

At each time step, we determine the reference position $\mathbf{p}_{\mathrm{ref}} = (x_{\mathrm{ref}}, y_{\mathrm{ref}}, z_{\mathrm{ref}})$ by projecting our current multi-rotor position $\mathbf{p}_{\mathrm{t}}^{\mathrm{lt}}$ onto the straight-line segment connecting the *trunk* to the *next* privileged vertex using:

$$\mathbf{p}_{\mathrm{ref}} = \mathbf{p}_{\mathrm{t}}^{\mathrm{lt}} \cdot \mathbf{p}_{\mathrm{t}}^{\mathrm{nt}} \frac{\mathbf{p}_{\mathrm{t}}^{\mathrm{nt}}}{|\mathbf{p}_{\mathrm{t}}^{\mathrm{nt}}|}.$$

We obtain a reference velocity $\mathbf{v}_{ref} = (\dot{x}_{ref}, \dot{y}_{ref}, \dot{z}_{ref})$, where the magnitude is a user-selected parameter v_{des}, in the direction of the *next* privileged vertex using:

$$\mathbf{v}_{\mathrm{ref}} = v_{des} \frac{\mathbf{p}_{\mathrm{t}}^{\mathrm{nt}}}{|\mathbf{p}_{\mathrm{t}}^{\mathrm{nt}}|}.$$

2) Control Design: Our path-following control is designed to send commands $(\dot{z}_{\mathrm{cmd}}, \dot{\psi}_{\mathrm{cmd}}, \theta_{\mathrm{cmd}}, \phi_{\mathrm{cmd}})$ where $\dot{z}_{\mathrm{cmd}}$ is a commanded z-velocity, $\dot{\psi}_{cmd}$ is a command yaw rate, and θ_{cmd} and ϕ_{cmd} are commanded pitch and roll, respectively. The z-velocity command is designed using a P-D controller:

$$\dot{z}_{\mathrm{cmd}} = \frac{2\zeta_z}{\tau_z}(z_{\mathrm{ref}} - z) + \frac{1}{\tau_z^2}(\dot{z}_{\mathrm{ref}} - \dot{z}), \tag{2}$$

where ζ_z and τ_z are tuned damping ratio and time constant. The current yaw, ψ, with respect to the *trunk* is determined from the rotation matrix, $\mathbf{C}_{\mathrm{tl}}$. As seen in (3), a P-controller (with tuned time constant τ_ψ) is used to correct for any yaw-mismatch between the *leaf* and the *trunk*:

$$\dot{\psi}_{cmd} = -\frac{1}{\tau_\psi}\psi. \tag{3}$$

As in [29], lateral-motion control commands are determined by first designing translational acceleration commands using P-D control:

$$a_x = \frac{2\zeta_\theta}{\tau_\theta}(x_{ref} - x) + \frac{1}{\tau_\theta^2}(\dot{x}_{ref} - \dot{x}), \tag{4a}$$

$$a_y = \frac{2\zeta_\theta}{\tau_\theta}(y_{ref} - y) + \frac{1}{\tau_\theta^2}(\dot{y}_{ref} - \dot{y}), \tag{4b}$$

where ζ_θ and τ_θ are tuned damping ratio and time constant. Assuming small lateral acceleration ($\ddot{x} \approx \ddot{y} \approx 0$) and using standard feedback linearization, these linear acceleration commands are transformed into pitch and roll commands:

$$\theta_{cmd} = \arcsin\left(\frac{a_x}{g}\cos\psi + \frac{a_y}{g}\sin\psi\right), \tag{5a}$$

$$\phi_{cmd} = -\arcsin\left(-\frac{a_x}{g}\sin\psi + \frac{a_y}{g}\cos\psi\right), \tag{5b}$$

where g is the gravitational constant.

In general, this controller can follow any arbitrary path, as long as vertices are placed sufficiently close together to accurately reflect the true path. However, path-curvature and large direction changes are not accounted for by this controller, the results of which will be examined in Section IV.

IV. Experiments

To evaluate the performance of the airborne VT&R algorithm, a number of outdoor experiments were performed on-board the target UAV.

In the first experiment, we evaluate the performance of the localisation algorithm under GPS control using the described gimbal controller. Specifically, we test the performance of the localisation algorithm and gimbal controller under deliberately challenging conditions, including high-speed, dynamic flight and high *learn* vs. *return* positional error. For this experiment, we deliberately exclude the vehicle controller to isolate the performance of the subcomponents of the algorithm. In the second experiment, we perform closed-loop control with the aforementioned path-following controller developed for full 6-DOF vehicle motion. This system is evaluated over several runs, showing the full system operating. The experimental setup is described in the following subsection.

A. Experimental Setup

For these experiments, we use a DJI Matrice 600 Pro, with attached Ronin-MX gimbal (Fig. 1). This system has a take-off weight of approximately 10 kg, and maximum span rotor-tip-to-tip of 1.64 m. Control is provided by a DJI A3 triple redundant autopilot. On-board this system is an NVIDIA Tegra TX2 module (6 ARM cores + 256 core Pascal GPU) and StereoLabs ZED stereo camera connected via USB3, both mounted in the stabilised platform of the Ronin-MX gimbal. The Matrice 600 Pro provides state information to VT&R running on-board the TX2, including gimbal encoder positions and GPS status, while the ZED camera provides greyscale imagery with resolution 672×376 at 15 Hz. The Tegra TX2 runs NVIDIA L4T v28.2, a variant of Ubuntu 16.04 for ARM architectures.

The primary location used for the experiments in this letter is a simulated village at the Defence Research & Development Canada (DRDC) Suffield Research Centre in southern Alberta, Canada. The Suffield location consists of a number of shipping containers placed to emulate buildings and narrow alleys in flat grassland, suited to a simulated patrol scenario.

B. Localization Performance Evaluation

In these experiments, we evaluate the combined performance of the localisation algorithm and gimbal controller to

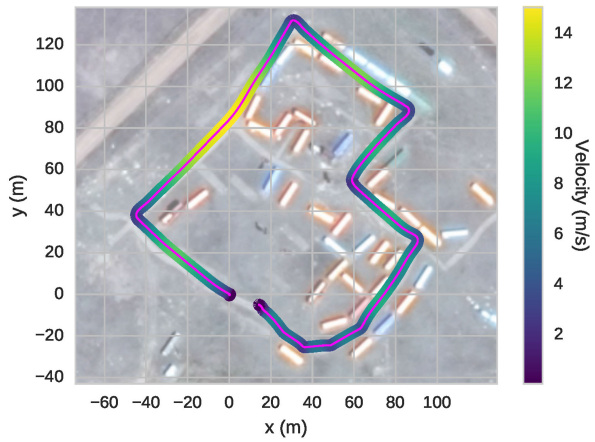

Fig. 4. Overview of the trajectory flown at the DRDC Suffield Research Centre, shown in magenta. Velocity profile for a target 15 m/s commanded speed overlaid.

successfully localise the vehicle under increasingly difficult operational conditions. We test this in two ways: increasing target velocity of the vehicle, and deliberately offset altitudes on the outbound and return paths. The first test shows the performance under increasingly dynamic manoeuvres of the vehicle, inducing rapid perspective change and poor path tracking, which must be attenuated by the gimbal controller. The second test shows the performance of the localisation algorithm with intentionally poor perspective. We deliberately do not use the vehicle controller in these tests to decouple and isolate the performance of the localisation algorithm and gimbal controller.

1) Increasing Target Velocity: For this experiment, the aircraft is autonomously flown at 12 m Above Ground Level (AGL) along the path depicted in Fig. 4 in a clockwise direction. VT&R is placed into *learn* mode, before the outbound route is flown under autonomous control, by uploading a waypoint mission to the Matrice 600 autopilot. Once the vehicle reaches the end of the loop, VT&R is switched to *return* mode, and the aircraft is again autonomously commanded to return along the same path by following the waypoints in reverse. During this return stage, the gimbal is actively controlled by VT&R to reduce orientation errors caused by path-following discrepancies generated by the GPS-based controller.

The route is flown at increasingly fast target speeds, 3, 7, 8, 10, 12 and 15 m/s, on both the learn and return stages. While the vehicle reaches this speed during only parts of this path, the average speed also increases with each pass. A typical speed profile for the path at 15 m/s target speed is shown in Fig. 4.

Fig. 5 shows the median and variance of the localisation inliers recorded along the return path for each of the target speeds. This figure shows that even at the highest commanded speed (15 m/s), localization performance remains similar to those examples at lower speeds. At 15 m/s some localisation failures occur, but the majority of these can be attributed to failures of the hardware gimbal controller during a segment of the path.

2) Increasing Height Error: For this experiment, the aircraft is again autonomously flown at 12 m AGL during the *learn* stage along the path depicted in Fig. 4 in a clockwise direction at a nominal speed of 7 m/s. The total length of the path is approx. 450 m. Once the vehicle reaches the end of the loop, VT&R is

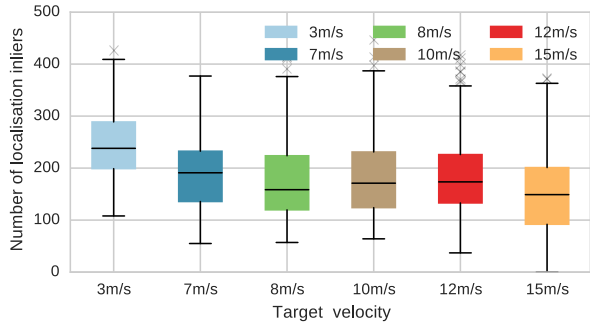

Fig. 5. Localisation performance is comparable with increasing target (and average) velocity, where *learn* and *return* phases are conducted at the same speed.

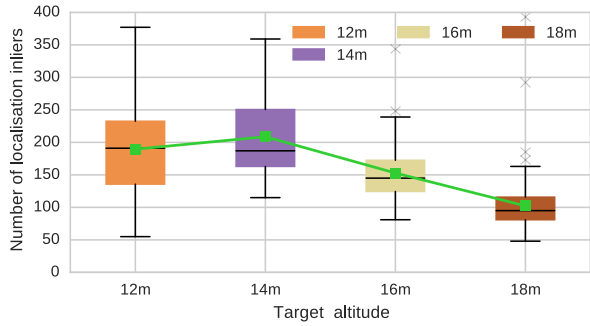

Fig. 6. Successful localisation occurs with significantly increasing altitude difference between *learn* (12 m) and *return* phases (tested at 12, 14, 16 and 18 m), but the average (green) shows decline at more extreme (50%) differences.

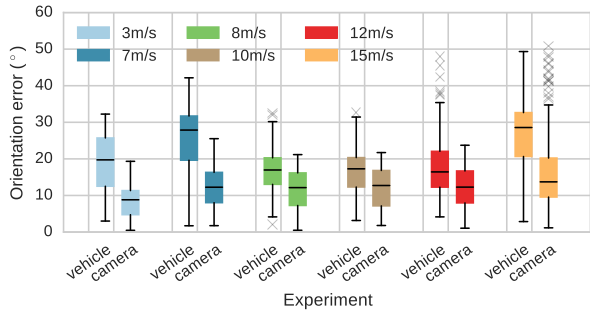

Fig. 7. While vehicle attitude error increases in both median and variance with increasing target velocity, the gimbal controller maintains a consistent camera orientation between *learn* and *return* regardless of target speed.

switched to *return* mode, and the aircraft is again autonomously commanded to return along the same path by following the waypoints in reverse at the same 7 m/s target speed. In this case, however, we vary the altitude at which the aircraft returns, to test the robustness of the gimbal controller and ability of the algorithm with large positional offsets. In these experiments, we show the localisation inliers along the path with target return heights of 12, 14, 16 and 18 m, respectively (Fig. 6).

In Fig. 6, localisation performance is still high, with an average of 100 inliers per keyframe, even at altitude differences of 6 m, or 50%. While some of this performance can be attributed to perspective due to the altitude, a significant component can be attributed to the gimbal compensating for the reduced image overlap that would be present on a static camera. Importantly, however, the average inliers does drop significantly, and more

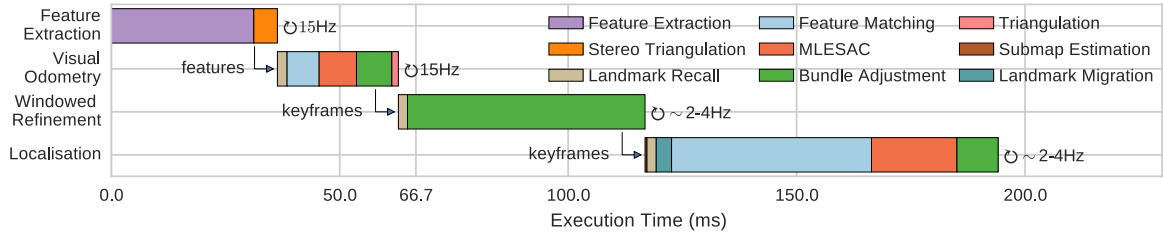

Fig. 8. Average execution times for the visual pipelines on the TX2 during a mission, separated by thread of execution. Once one thread is finished processing, it is able to process the next image and its data products.

interestingly, reduces in variance. This is likely due to the enhanced viewpoint overlap (of the *learnt* path) at higher altitudes, meaning positional errors have less effect on maintaining observability of all landmarks during localisation.

Finally, Fig. 7 shows the utility of the gimbal in minimising perspective error caused by differing vehicle attitudes between *learn* and *return*. Due to the obvious underactuated nature of multirotor systems, accelerations and decelerations cause the vehicle attitude to differ between these two passes of the path. For a static camera system, these differences can cause performance degradation due to poor image overlap. Using a gimbal with active control to attenuate camera orientation error can minimise this effect. Fig. 7 shows the magnitude of orientation error for two separate localisation transforms at each speed profile (read as a pair) taken from the estimated localisation chain as estimated from the visual pipeline: in the vehicle frame ($\mathbf{T}_{\mathfrak{l}\mathfrak{t}}$, or $\mathbf{T}_{fa}$ in Fig. 3) on the left, and in the camera frame ($\tilde{\mathbf{T}}_{fa}$ in Fig. 3) on the right. As can be seen at all speed profiles, the gimbal succeeds in minimising the orientation of the localisation transform, and this performance is relatively consistent with increasing speed. In this scenario, the target speeds of the *learn* and *return* phases are the same for each speed profile, meaning there will be some consistency in orientation in both phases. With differing speed profiles, we would expect the observed utility of the gimbal-stabilised camera to increase further.

3) Execution Time: Fig. 8 shows the average execution time for the separate processes in the VT&R algorithm on-board the Tegra TX2. While *feature extraction* and *VO* process every image pair at an approximate speed of 66 ms (∼15 Hz), *windowed refinement* only runs on generation of a keyframe, and *localisation* runs after this process is complete. Feature extraction and matching are all performed on the GPU. Using this threaded setup allows VT&R to run online.

C. Full VT&R Evaluation

In this experiment, we evaluate the performance of the full closed-loop VT&R system, using GPS navigation during the *learn* phase, and switching to the presented path-following controller for the *return* phase. Over three separate trials, each consisting of a single flight, we traverse the path shown in Fig. 4 in a clockwise direction at an altitude of 12 m AGL, before returning in an anticlockwise direction at the same altitude (attempting to minimise all positional errors) at a target speed of 3 m/s.

In all three trials, VT&R was able to complete the return phase of flight under path-following control over an approximately 2.5 minute period. Fig. 9 shows the path for these trials. The

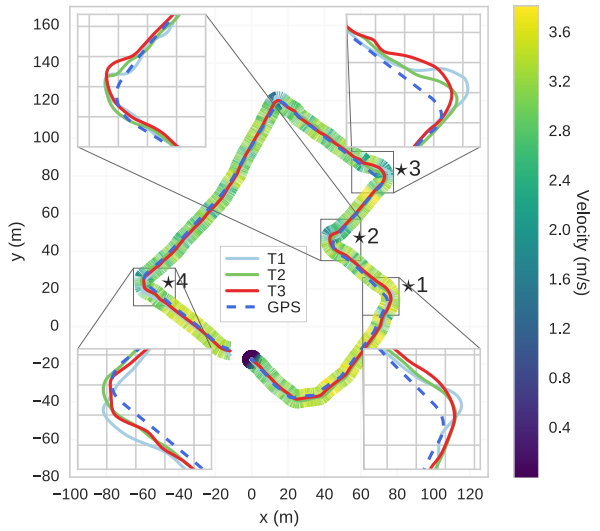

Fig. 9. The outbound (GPS, *dashed blue*) and return (controller, *light blue, green, red*) paths during for three separate trials. Some offset is seen on sharp turns. Velocity profile for trial T3 overlaid.

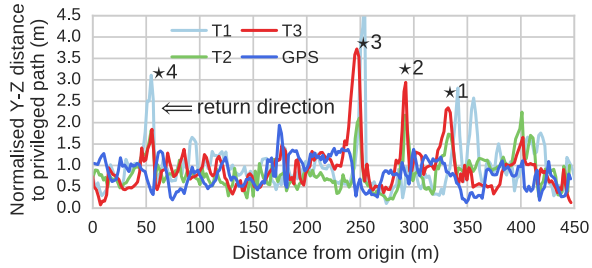

Fig. 10. Path-following error is of a similar order to that for GPS-based control over the majority of the path using our controller, according to the localisation transform estimated by VT&R.

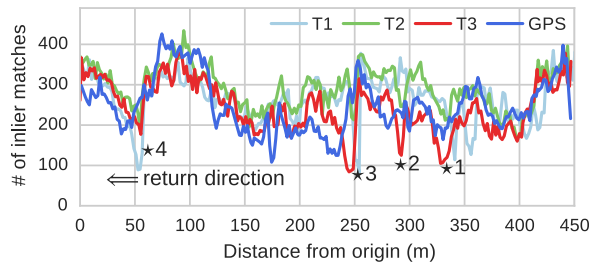

Fig. 11. The number of localization inliers while using our path-following controller is of a similar order to that for GPS-based control over the majority of the path.

outbound path under GPS control is shown in dashed blue, while the return path under path-following control is shown in light blue, green and red for trials T1, T2 and T3, respectively. Figs. 10 and 11 show the normalised cross-track error (in Y and Z, using the vision-based estimate) and number of inlier matches respectively for each trial in comparison to a return flight under GPS control. Specific segments of the path are highlighted in the inset figures of Fig. 9 and annotated with numbers that correlate to those in Figs. 10–11.

The positional error is less than 1.5 m over most of the path using the path-following controller (Fig. 10), and is comparable to a return trajectory under GPS control, showing the strong performance of a simple vision-based path-following controller compared to this primary sensor. In specific sections such as corners, however, cross-track error increases to a maximum of 4.5 m. This can be attributed to the simplicity of the controller, as curvature of the path is not accounted for, and velocity error is weighted higher than cross-track error.

Despite these path-following errors, localisation performance is strong over the full trajectory (Fig. 11), with no localisation failures, even at the highlighted corner points. The average performance over the trajectory is again comparable to a return phase under GPS control.

V. Conclusions

In this letter, a full VT&R system for emergency return of a multirotor UAV has been presented. Using imagery from a gimbal-stabilised stereo camera to build a map online during a commanded *learn* phase, autonomous return of the vehicle in the same flight has been demonstrated by matching live landmarks to the map for autonomous path-following control. In addition, we have demonstrated the robustness of the gimbal-stabilised system to high-speeds and large positional errors.

Future work will focus on a more advanced path-tracking controller to minimise cross-track errors, and testing in a multi-experience framework over a long-term experiment.

Acknowledgment

An accompanying video for this paper is available at: tiny.cc/noplacelikehome.

References

[1] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *J. Field Robot.*, vol. 27, no. 5, pp. 534–560, 2010.

[2] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments," in *Proc. Int. Conf. Intell. Robots Syst.*, IEEE, 2013, pp. 176–181.

[3] M. Paton, K. Mactavish, M. Warren, and T. D. Barfoot, "Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat," in *Proc. Int. Conf. Intell. Robots Syst.*, 2016, pp. 1918–1925.

[4] T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, and L. Peučil, "Simple yet stable bearing-only navigation," *J. Field Robot.*, vol. 27, no. 5, pp. 511–533, 2010.

[5] M. Paton, F. Pomerleau, and T. D. Barfoot, "Eyes in the back of your head: Robust visual teach & repeat using multiple stereo cameras," in *Proc. 12th Can. Conf. Comput. Robot Vis.*, 2015, pp. 46–53.

[6] A. Pfrunder, A. P. Schoellig, and T. D. Barfoot, "A proof-of-concept demonstration of visual teach and repeat on a Quadrocopter using an altitude sensor and a monocular camera," in *Proc. Can. Conf. Comput. Robot Vis.*, 2014, pp. 238–245.

[7] M. Warren, M. Paton, K. MacTavish, A. P. Schoellig, and T. D. Barfoot, "Towards Visual teach & repeat for GPS-denied flight of a fixed-wing UAV," in *Proc. 11th Int. Conf. Field Serv. Robot., Results,* 2017, pp. 481–498.

[8] A. G. Toudeshki, F. Shamshirdar, and R. Vaughan, "UAV visual teach and repeat using only semantic object features," in *Proc. Can. Conf. Comput. Robot Vis.*, 2018.

[9] J. Surber, L. Teixeira, and M. Chli, "Robust visual-inertial localization with weak GPS priors for repetitive UAV flights," in *Proc. Int. Conf. Robot. Automat.*, 2017, pp. 6300–6306.

[10] M. Warren, A. P. Schoellig, and T. D. Barfoot, "Level-headed: Evaluating Gimbal-stabilised visual teach and repeat for improved localisation performance," in *Proc. Int. Conf. Robot. Automat.*, Brisbane, 2018, pp. 7239–7246.

[11] N. Playle, *"Improving the performance of monocular visual simultaneous localisation and mapping through the use of a Gimballed camera*," M.S. thesis, Graduate Dept. Aerosp. Sci. Eng.University of Toronto, Toronto, ON, Canada, 2009.

[12] C. S. Sharp, O. Shakernia, and S. Sastry, "A vision system for landing an unmanned aerial vehicle," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2001, pp. 1720–1727.

[13] A. Borowczyk, D.-T. Nguyen, A. P.-V. Nguyen, D. Q. Nguyen, D. Saussie, and J. L. Ny, "Autonomous landing of a Quadcopter on a high-speed ground vehicle," *J. Guid., Control, Dyn.*, vol. 40, no. 9, pp. 2378–2385, 2017.

[14] C. E. Lin, "Camera Gimbal Tracking from UAV Flight Control," in *Proc. CACS Int. Autom. Control Conf.*, pp. 319–322, 2014.

[15] C. L. Choi, J. Rebello, L. Koppel, P. Ganti, A. Das, and S. L. Waslander, "Encoderless Gimbal calibration of dynamic multi-camera clusters," in *Proc. Int. Conf. Robotics Automat.*, Brisbane, 2018, pp. 2126–2133.

[16] A. Giusti *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 661–667, Jul. 2016.

[17] M. W. Achtelik, M. C. Achtelik, S. M. Weiss, and R. Siegwart, "Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments," in *Proc. Int. Conf. Robotics Automat.*, IEEE, May 2011, pp. 3056–3063.

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

[19] S. M. Weiss *et al.*, "Monocular vision for longterm micro aerial vehicle state estimation: A compendium," *J. Field Robot.*, vol. 30, no. 5, pp. 803–831, 2013.

[20] T. Hinzmann *et al.*, "Monocular visual-inertial SLAM for fixed-wing UAVs using sliding window based nonlinear optimization," in *Proc. Int. Symp. Vis. Comput.*, Springer International Publishing, 2016, pp. 569–581.

[21] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera Systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2017.

[22] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[23] K. Sun *et al.*, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 965–972, 2017.

[24] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small Quadrotor with a single camera and IMU," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 404–411, Apr. 2017.

[25] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV," in *Proc. Int. Conf. Robot. Automat.*, 2014, pp. 4974–4981.

[26] T. Qin, P. Li, and S. Shen, "Relocalization, global optimization and map merging for monocular visual-inertial SLAM," in *Proc. Int. Conf. Robot. Automat.*, Brisbane, 2018, pp. 1197–1204.

[27] S. Anderson and T. D. Barfoot, "Full STEAM ahead: Exactly sparse Gaussian process regression for batch continuous-time trajectory estimation on SE(3)," in *Proc. Int. Conf. Intell. Robots Syst.*, Sep. 2015, pp. 157–164.

[28] J. E. Hauser and R. Hindman, "Maneuver regulation from trajectory tracking: Feedback linearizable systems," in *Proc. 3rd IFAC Symp. Nonlinear Control Syst. Des.*, 1995, vol. 28, no. 14, pp. 638–643.

[29] S. Spedicato, A. Franchi, and G. Notarstefano, "From tracking to robust Maneuver regulation : An easy-to-design approach for VTOL aerial robots," in *Proc. Int. Conf. Robotics Automat.*, 2016, pp. 2965–2970.