2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Visual-inertial navigation algorithm development using photorealistic camera simulation in the loop

Thomas Sayre-McCord[1], Winter Guerra[1], Amado Antonini[1], Jasper Arneberg[1], Austin Brown[1], Guilherme Cavalheiro[1], Yajun Fang[2], Alex Gorodetsky[3], Dave McCoy[1], Sebastian Quilter[1], Fabian Riether[1], Ezra Tal[1], Yunus Terzioglu[1], Luca Carlone[1], and Sertac Karaman[1]

***Abstract*— The development of fast, agile micro Unmanned Aerial Vehicles (UAVs) has been limited by *(i)* on-board computing hardware restrictions, *(ii)* the lack of sophisticated vision-based perception and vision-in-the-loop control algorithms, and *(iii)* the absence of development environments where such systems and algorithms can be rapidly and easily designed, implemented, and validated. Here, we first present a new micro UAV platform that integrates high-rate cameras, inertial sensors, and an NVIDIA Jetson Tegra X1 system-on-chip compute module that boasts 256 GPU cores. The UAV mechanics and electronics were designed and built in house, and are described in detail. Second, we present a novel "virtual reality" development environment, in which photorealistically-rendered synthetic on-board camera images are generated in real time while the UAV is in flight. This development environment allows us to rapidly prototype computing and sensing hardware as well as perception and control algorithms, using real physics, real interoceptive sensor data (e.g., from the on-board inertial measurement unit), and synthetic exteroceptive sensor data (e.g., from synthetic cameras). Third, we demonstrate repeated agile maneuvering with closed-loop vision-based perception and control algorithms, which we have developed using this environment.**

I. Introduction

Unmanned Aerial Vehicle (UAV) systems have been demonstrated in many domains, ranging from agriculture to consumer utilization [1]. In most applications, human operators are still required for the safe and high-performance operation of the system. Although fully autonomous UAVs have long been available, their capabilities still do not match the operational speeds that can be easily achieved by minimally-trained UAV operators. While various algorithmic components, such as control, planning, or perception for agile flight have been demonstrated in isolation using off-the-shelf components, a system that integrates control and perception algorithms developed from the ground-up has not yet been designed, developed, and demonstrated.

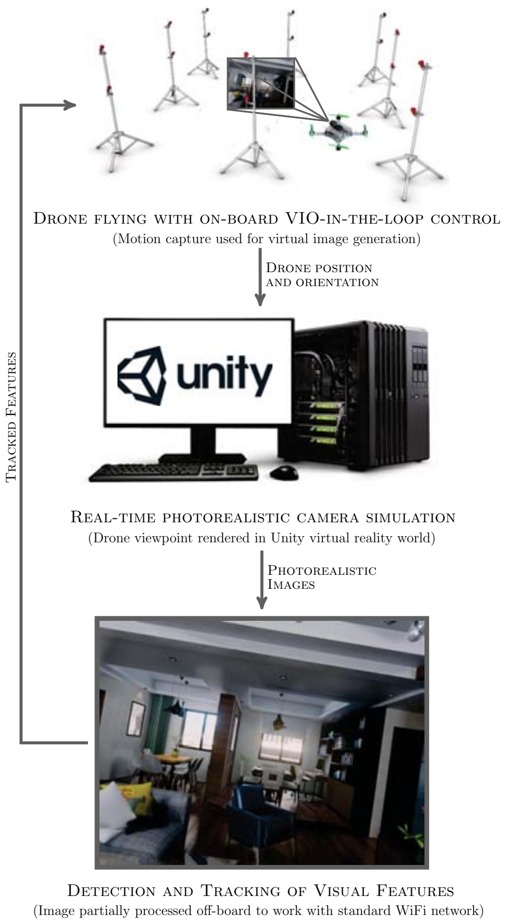

Drone flying with on-board VIO-in-the-loop control
(Motion capture used for virtual image generation)
Drone position and orientation
unity
Real-time photorealistic camera simulation
(Drone viewpoint rendered in Unity virtual reality world)
Photorealistic Images
Detection and Tracking of Visual Features
(Image partially processed off-board to work with standard WiFi network)
Tracked Features

Fig. 1: To enable algorithmic work in a wide range of visual conditions we have developed a system to replace the UAV's on-board camera with images from a virtual environment. While the UAV is in flight (top) the motion capture pose estimate of the UAV is sent to the Unity game engine running on a TitanX GPU (middle) which can generate the corresponding photorealistic image (bottom) for that pose from a virtual world which is processed and transmitted to the UAV in real time. The system runs fully in real time as if the sensors were on the UAV, allowing experiments and decision making in adverse conditions such as obstacle rich environments or in environments that are difficult to access such as cities.

An important opportunity that may help close the gap is the emergence of powerful embedded supercomputers that can process high-rate, high-resolution exteroceptive sensory data in an effective manner, in order to enable situational awareness and accurate state estimation for closed-loop agile control at high speeds. Developing the required algorithms, however, is a major challenge due to the lack of two critical

*This work was supported in part by the Office of Naval Research (ONR) through the ONR YIP program.

[1] Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA.

[2] Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, Cambridge, MA 02139, USA.

[3] Sandia National Laboratories, Albuquerque, NM 87185, USA.

Emails:{`rtsm, amadoa, jaspera, austinb, guivenca, yjfang, goroda, winterg, dmccoy, squilter, friether, eatal, terzi, lcarlone, sertac`}`@mit.edu`

978-1-5386-3081-5/18/$31.00 ©2018 IEEE

components: *(i)* powerful research platforms equipped with high-rate electronics and massively-parallel embedded computers and *(ii)* safe development environments that can help propel algorithm and software development.

In this paper, we present a powerful UAV system that is capable of high-speed agile navigation in GPS-denied environments using control closed on visual-inertial state estimation. For this purpose, we develop: *(i)* state-of-the-art mechanical and electronics hardware that integrate a powerful embedded supercomputer, the NVIDIA Jetson Tegra X1, with an inertial measurement unit and a camera, centered around the design of a custom NVIDIA Jetson carrier board; *(ii)* a unique virtual-reality UAV development environment, which we call *FlightGoggles*, that allows us to simulate camera images photorealistically while the UAV is in flight, providing the integration of simulated visual data with real inertial measurements; *(iii)* state-of-the-art visual-inertial state estimation algorithms that give an accurate state estimate for closed-loop navigation in complex environments, such as through doors and windows, in a robust and repeatable manner. The development and integration of these three systems are the main contributions of the present paper.

To the best of our knowledge, in this paper we present one of the most capable NVIDIA Jetson carrier boards designed specifically for the development of high-speed agile flight by integrating key peripherals that allow high-resolution, high-rate cameras along with precision inertial measurement units. Furthermore, we present the idea of a "virtual-reality" UAV development environment. While using simulation systems for UAV development has attracted a vast amount of attention, especially very recently with the introduction of AirSim by Microsoft [2], our system utilizes a motion capture environment to photorealistically render camera images which are then fed back to the UAV for active decision making and closed-loop flight control. In this system, the physics are real, the inertial measurements are real, but all exteroceptive measurements are simulated photorealistically in *real time* with the help of powerful desktop GPUs. We emphasize that this system does *not* simulate physics and interoceptive measurements. Hence, this system is best suited for applications involving complex physics, e.g., when aerodynamic effects are dominant, and complex electromechanical effects dominate propulsion forces. Because only a single element of the system is simulated, this environment allows us to rapidly develop agile UAVs and move into field deployments in a safe and scalable manner.

Finally, we develop visual-inertial navigation algorithms that integrate monocular camera images and inertial measurements to estimate the vehicle's state in real time onboard the drone for closed loop control. The visual-inertial odometry algorithm presented in this paper validates the idea of utilizing synthetic camera images generated in real time together with real inertial measurements in closed-loop flight. The image simulation system is used to develop the visual-inertial algorithms in a challenging scenario of flying through a window gap, which is subsequently verified in laboratory experiments with an on-board camera.

II. Related Work

UAV Systems. Multi-rotor UAVs for research in vision based algorithms have primarily used off-the-shelf flight platforms and/or autopilot systems that have been modified with additional sensors and computation power. The most popular off-the-shelf platforms have been the AscTec Hummingbird [3], [4], the AscTec Pelican [5], and the Parrot AR Drone [6]. A few custom platforms have also been built, typically augmented with an off-the-shelf autopilot board [7], [8]. Due to being lightweight and low-power the most popular sensor package for UAVs is a single camera (either forward or downward facing) and an IMU, although lightweight 2D laser scanners [4], stereo camera pairs [4], [9], and an RGB-D sensor [10] have also been used. The platform we present in this paper is built from the ground-up to house high-end inertial measurement units, high-rate high-resolution cameras, and state-of-the-art embedded GPU computing systems.

Synthetic Environments for Robotics. There has been a variety of work on the use of synthetic data sets and simulation in robotics and more generally computer vision. Synthetically generated data sets, such as those in [11], [12], have become of particular interest as the need for large labeled data sets for deep learning has become prevalent. Of particular note to the work presented here is the method of Richter *et al.* in [13] which uses pre-built video games to generate semantically mapped synthetic data sets. Kavena *et al.* use photorealistic renderings to evaluate the performance of different feature descriptors under various camera conditions [14]. Handa *et al.* provide a synthetic data set for the verification of SLAM algorithms against a known 3D model and trajectory [15]. In robotics, Gazebo [16] is the ubiquitous full simulation environment, with specific applications to UAVs in RotorS [17], which is studied in depth in [18]. Of primary relevance to this work is Microsoft Research's recent release of a developing project, AirSim, an Unreal Engine based simulation environment for UAVs [2]. AirSim is a plug-in to Unreal Engine providing a rendered viewpoint of a simulated (or possibly real) UAV location in the Unreal world. Early releases have an eye toward being able to generate large data sets for deep learning based off a simulated UAV model.

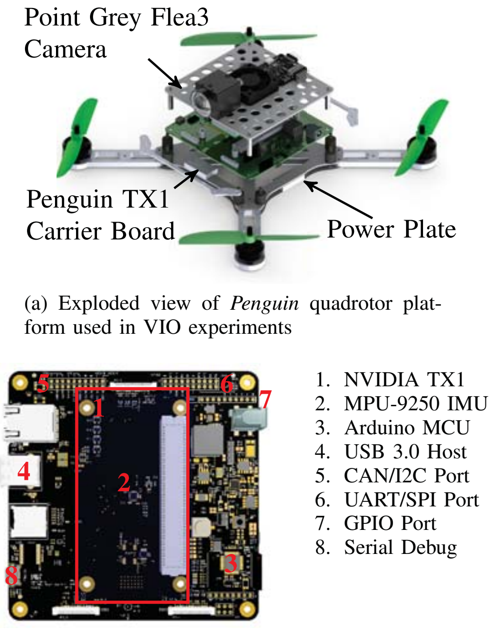
Visual Inertial Navigation. The literature on visual inertial navigation is vast, including approaches based on filtering, e.g., [19], [20], fixed-lag smoothing, e.g., [21], [22], and full smoothing [23]–[25]. We refer the reader to the recent survey by Forster *et al.* [24] for a comprehensive review. As the computational power that can be carried on a flying platform has increased, some visual-inertial navigation algorithms have begun to be run in real time on UAVs. Early implementations [6], [26], [27] focused on extending the full SLAM system of Klein and Murray (PTAM) [28] to work on aerial vehicles. More recent approaches have included using a cascading estimate of orientation and position with a low rate stereo camera [9], replacing the PTAM visual SLAM system with the semi-direct approach SVO [29] augmented by an IMU [8], using an off-the-shelf pose estimate from an RGB-D sensor (Google Tango) [7], and a factor graph based approach similar to our own [30].

III. System

A UAV test platform and a development environment was built for the testing of on-board estimation and control algorithms while performing agile maneuvers. To integrate the electronics on the UAV, a custom carrier board for the NVIDIA TX module was designed, providing the interfaces necessary for sensing and control, while minimizing size and weight. A mechanical frame was designed and built around this carrier board. The UAV is fully controlled by an on-board NVIDIA TX1 module with a modular software framework, enabling rapid testing of new algorithms and sensors. A real-time visual simulation environment runs using a motion capture system and the Unity game engine, allowing for rapid prototyping of visual algorithms.

UAV Mechanics. The mechanical layout of the UAV consists of a series of three stacked plates carrying the power, control, and sensors (see Fig. 2a). At the bottom is the power plate carrying the electronic speed controllers (ESCs), power distribution board, batteries, and the four quadrotor arms with motors and propellers. The middle board is the TX module carrier board, and the top plate serves as a utility plate for mounting the camera(s), external IMU, NAZE flight controller (safety mechanism only), and the WiFi antennas. To reduce the vibration on the sensors, the bottom "dirty" plate is separated from the top two "clean" plates by four mechanical dampeners. For maximum agility, the motors are placed as close as possible to the center of the board.

UAV Electronics. The *Penguin* Carrier Board (Fig. 2b) carried by *Penguin* was designed in-house to integrate the TX1 module with the rest of the vehicle. The board is designed to minimize size and weight while providing seamless integration of the essential capabilities. Elements such as an extra microcontroller (Atmel328P MCU) to control the ESCs and high speed data lanes for IMU and camera data were integrated to provide autonomous flight. A single USB 3.0 Point Grey Flea3 monochrome camera with a resolution of 1024x1280 and an external Xsens MTi-3 IMU provide the visual and inertial sensor package for the board. The Point Grey camera uses a Sunex DSL219 fisheye lens; to avoid the high distortion at the edges of the lens only the center of the image was used for VIO algorithms, leading to an effective resolution of 512x640.



(a) Exploded view of *Penguin* quadrotor platform used in VIO experiments

(b) *Penguin* Carrier Board

Fig. 2: Mechanical and electronic designs for our agile quadrotor platform *Penguin*. The quadrotor platform is CNC routed out of Garolite G10 laminate for a strong and lightweight housing of the drone electronics. Custom carrier boards were designed in house to provide all of the essential elements for flight (TX module, IMU, camera(s), motor control) while minimizing the weight and footprint of the electronics.

UAV Software Setup. The UAV is controlled through an on-board software setup that provides complete end-to-end operation of the UAV from raw sensor data to the signals sent to each ESC. The system uses Lightweight Communications and Marshaling (LCM) [31] for communication between on-board modules, giving a lightweight and flexible framework. Each on-board module (controller, VIO estimation, motor control) remains agnostic to its data source, allowing for easy switching between methods and data sources (e.g. moving from motion capture to VIO state estimation). All processing occurs on-board the CPU and GPU of the TX1.

Development Environment for Photorealistic Sensor Simulation in the Loop (PiL). Simulation of images through a Unity engine is enabled via a ground station computer featuring an NVIDIA TitanX GPU for rapid rendering of images based on the motion capture location of the UAV. The simulation of imagery is performed by creating an environment in Unity that contains a virtual world we wish to fly in, and one or more camera objects which are attached to a TCP socket. Over TCP, the various parameters of the camera may be set, most importantly the camera pose can be set in real time based on the motion capture position of the UAV. For each pose of the camera received, the Unity camera object returns a timestamped image of the virtual reality environment as it would be seen from that pose (see Figure 1). Because of the networking limitations of sending full images wirelessly to the UAV, for our VIO state estimation experiments using simulated imagery the vision front-end is executed on the ground station computer at speed that the on-board GPU can execute, and only feature data is sent wirelessly to the UAV for state estimation. The total delay in receiving visual data on the UAV (rendering and wireless transmission) is primarily Gaussian around 37 ± 8 ms with 1.3% outliers above two standard deviations due to wireless network bottlenecks. For comparison, the time from image acquisition to processed data with our live camera is 15 ± 5 ms.

Although our system is intended for UAV flight experi-

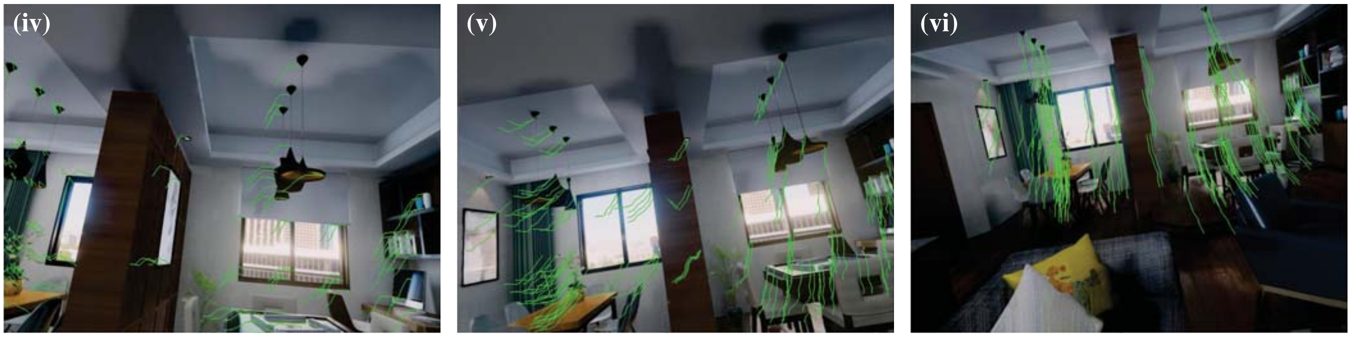


Fig. 3: Three photorealistic images generated and streamed to the drone in real time for VIO state estimation during a VIO experiment. Green lines show 0.3 seconds of history for visual features detected and tracked by the vision front end. Our simulation system generates photorealistic images at 60 Hz based on motion capture pose estimate, which are processed by the VIO system, and used in-the-loop for controlling the drone's flight.

ments with Photorealistic (exteroceptive) sensor simulation in the Loop (PiL), it can also be used for replicating the traditional Hardware in the Loop (HiL) simulations in which the UAV physics is also simulated in real time along with the sensors, while the computing hardware is in the loop.

IV. Algorithms

A. State Estimation

State estimation is based on an Extended Kalman Filter (EKF) powered by Visual-Inertial Odometry (VIO). The VIO algorithms estimate the motion of a device from visual and inertial cues. Our VIO approach is based on the work by Forster *et al.* [24] with modifications made to allow for real time state estimation on the limited computation of a TX1 module. In the following, we discuss the different components of our VIO pipeline, made up of the low-level signal processing (*vision and IMU front-end*), the inference engine used for accuracy (*estimation back-end*), and the high-rate visual-inertial filter used in the control loop (*visual-inertial EKF*).

The Vision Front-end. Our vision front-end includes *feature detection*, *tracking*, and *geometric verification*. The state estimation system uses a *keyframe* based scheme where computationally intensive tasks (feature detection, MAP estimation, geometric verification) only occur at keyframes, while computationally cheap tasks (feature tracking, EKF estimation) occur at full camera frame rate. A camera frame is declared to be a keyframe if one of three situations occurs: after a maximum amount of time has elapsed, after the smoother has finished processing the previous keyframes, or if the number of tracked features drops below a threshold. The feature detector, triggered at each keyframe, extracts Shi-Tomasi corners [32]. Between keyframes, given the pixel locations of the features in the $(k-1)$-th frame, we use the Lucas-Kanade feature tracking method for finding the location of these features in the k-th frame. We use OpenCV's GPU implementations for these tasks.

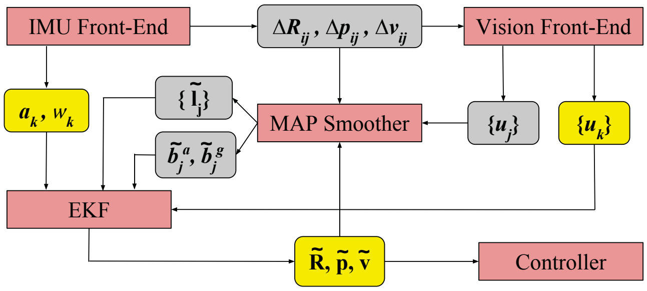

IMU Front-End
$\Delta R_{ij}, \Delta p_{ij}, \Delta v_{ij}$
Vision Front-End
a_k, w_k
$\{\tilde{l}_j\}$
MAP Smoother
$\{u_j\}$
$\{u_k\}$
$\tilde{b}_j^a, \tilde{b}_j^g$
EKF
$\tilde{R}, \tilde{p}, \tilde{v}$
Controller

Fig. 4: Diagram of the full VIO state estimation system. Gray items are transmitted at keyframe rate (3-10 Hz) and yellow items are transmitted at frame rate (60 Hz). Subscripts i and j denote subsequent keyframes, while subscript k denotes the current frame.

Verification of the tracked features is performed using 2-pt RANSAC [33] (implemented in OpenGV [34]) to determine the largest set of tracked features that could be described by a rigid body transformation given the rotation estimated from Euler integration of the on-board gyroscope.

The IMU Front-end. The IMU front-end is responsible for the preintegration of IMU measurements, which amounts to compressing the set of IMU measurements collected between two consecutive keyframes into a single preintegrated measurement and its corresponding covariance matrix.

The on-manifold preintegrated IMU model used in this system is described briefly below, a more detailed derivation can be found in [24]. Let us denote the accelerometer and gyroscope measurements acquired at time k by $a_k \in \mathbb{R}^3$ and $\omega_k \in \mathbb{R}^3$, and denote the estimated accelerometer and gyroscope bias at time k by $\tilde{b}_k^a \in \mathbb{R}^3$ and $\tilde{b}_k^g \in \mathbb{R}^3$. We wish to determine the relation between the state, x, of the UAV at two consecutive keyframes, where the state is made up of the attitude R, position p, velocity v and IMU biases b^a, b^g.

Considering two consecutive keyframes at time i and j, the IMU preintegration performs integration of the IMU measurements (a_k, ω_k) for all sampling times $k = i, \ldots, j$, with time spacing Δt, to produce a relative rotation $\Delta\tilde{R}_{ij}$, a psuedo-relative velocity $\Delta\tilde{v}_{ij}$, and a psuedo-relative position

$\Delta\tilde{p}_{ij}$ in the local frame at time i, as shown below [24]:

$$
\begin{aligned}
\Delta\tilde{R}_{ij} &= \prod_{k=i}^{j-1} \mathrm{Exp}((\omega_k - \tilde{b}_k^g)\Delta t) \\
\Delta\tilde{v}_{ij} &= \sum_{k=i}^{j-1} \Delta\tilde{R}_{ik}(a_k - \tilde{b}_k^a)\Delta t \\
\Delta\tilde{p}_{ij} &= \sum_{k=i}^{j-1} \left[\Delta\tilde{v}_{ik}\Delta t + \frac{1}{2}\Delta\tilde{R}_{ij}(a_k - \tilde{b}_k^a)\Delta t^2 \right],
\end{aligned}
\tag{1}
$$

These same values can be computed directly as functions of the keyframe states and noise values $\delta\phi_{ij}, \delta v_{ij}, \delta p_{ij} \in \mathbb{R}^3$ in Eqn. 2. The decoupling of measurement integration and keyframe states significantly saves computation by allowing for adjusting state estimates during optimization without reintegrating IMU measurements.

$$
\begin{aligned}
\Delta\tilde{R}_{ij} &= R_i^{\mathsf{T}} R_j \,\mathrm{Exp}(\delta\phi_{ij}) \\
\Delta\tilde{v}_{ij} &= R_i^{\mathsf{T}}(v_j - v_i - g\Delta t_{ij}) + \delta v_{ij} \\
\Delta\tilde{p}_{ij} &= R_i^{\mathsf{T}}(p_j - p_i - v_i\Delta t_{ij} - \frac{1}{2}g\Delta t_{ij}^2) + \delta p_{ij}
\end{aligned}
\tag{2}
$$

The Maximum a Posteriori (MAP) Back-end. To enable on-board processing, the optimization back-end performs fixed-lag smoothing and computes the MAP estimate of the most recent keyframe states within a given time window, using the measurements produced by the front-end. Older measurements are marginalized out of the factor graph, producing a faster computation at the expense of accuracy.

The *vision measurements* (produced by the vision front-end) represent the pixel observations of a landmark in a keyframe. The vision measurements are included in the MAP problem as structureless vision factors which treat the unknown landmark location l_m as a direct function of the measurements of the landmark and the state estimate, rather than as an unknown variable in the MAP estimation [24]. The *IMU measurements* (produced by the IMU front-end) are the preintegrated measurements $(\Delta\tilde{R}_{ij}, \Delta\tilde{v}_{ij}, \Delta\tilde{p}_{ij})$ in (1).

The MAP estimator is a nonlinear least squares optimization problem, whose minimum is the MAP estimate:

$$
\begin{aligned}
\min_x \sum_{(i,j)\in\mathcal{F}} \|r_{\mathrm{IMU}}(x_i, x_j, \Delta\tilde{R}_{ij}, \Delta\tilde{v}_{ij}, \Delta\tilde{p}_{ij})\|^2 + \\
\sum_{m\in\mathcal{L}} \sum_{i\in\mathcal{F}_m} \|r_{\mathrm{CAM}}(x_i, u_{im})\|^2 + \|r_{\mathrm{PRIOR}}(x)\|^2
\end{aligned}
\tag{3}
$$

where the elements of Eqn. (3) are the negative log-likelihood of the IMU measurements, vision measurements, and the priors, respectively. In Eqn (3), $\mathcal{F}$ is the set of consecutive keyframes indices, $\mathcal{F}_m$ is the set of keyframes in which landmark m has been observed, and $\mathcal{L}$ is the set of landmarks observed during the time horizon. The functions $r_{\mathrm{IMU}}(\cdot)$, $r_{\mathrm{CAM}}(\cdot)$, $r_{\mathrm{PRIOR}}(\cdot)$ are often called *residual errors* in that they quantify the mismatch between a given state estimate and the available measurements and priors. The optimization problem in Eqn. (3) is solved using the incremental smoothing algorithm iSAM2 [35] implemented in the GTSAM 4.0 toolbox [36].

Symbol	Property	Value
m	mass	1.05 kg
I_x	inertia around x axis	4.9×10^{-3} $kg\,m^2$
I_y	inertia around y axis	4.9×10^{-3} $kg\,m^2$
I_z	inertia around z axis	6.9×10^{-3} $kg\,m^2$
d	torque coefficient	2.6×10^{-8} $kg\,m^2$
b	thrust coefficient	1.89×10^{-6} $kg\,m$
d	thrust lever w.r.t x and y	0.158 m

TABLE I: Measured model parameters of the UAV

Visual-Inertial Extended Kalman Filter (EKF) Estimator. Because of the computation limitations on-board the UAV, the MAP estimation back-end runs with a keyframe rate of only 3-10 Hz, which is insufficient for use in closed loop control. Unlike standard techniques which use IMU data only to bridge the gaps in MAP estimates, we take advantage of the frame rate camera data in a visual-inertial EKF. The EKF follows the standard two step EKF process with a prediction step provided by IMU integration and an update step provided by comparing frame rate feature measurements against the estimated 3D landmark locations generated by the MAP estimator. The IMU state prediction is given by standard euler integration of the acceleration and angular velocity measurements, with the unknown bias terms $\tilde{b}_k^a$ and $\tilde{b}_k^g$ updated at each keyframe by the MAP estimator.

The camera update step runs at the frame rate of the camera (rather than the keyframe rate used for the MAP estimate) using the error between tracked pixel locations u_m of the m^{th} landmark and the back projection of the MAP estimated 3D location $\tilde{l}_m$ onto the camera.

By using the IMU bias and the 3D landmark locations from the MAP estimate, the EKF maintains similar accuracy with the MAP estimate, while still running at a rate suitable for closed-loop control.

The full estimation system, from processing IMU and camera data in the IMU/Vision front-ends, to a high rate state estimate from the EKF is shown in Figure 4.

B. Control

A backstepping controller based on the work of Bouabdallah and Siegwart [37] was implemented to perform trajectory tracking on the UAV. The controller uses an outer loop position controller and an inner loop orientation controller.

Position controller. The total thrust, U_z, is defined by an altitude controller according to:

$$
U_z = \frac{m}{\cos\phi\cos\theta}\left(e_z + g - \alpha_z\left(e_{\dot{z}} + \alpha_z e_z\right) - \alpha_{\dot{z}} e_{\dot{z}}\right) \tag{4}
$$

where ϕ (roll), θ (pitch), ψ (yaw) are the Euler angles that rotate XYZ (global frame) to xyz (body frame), $e_\eta = \eta^d - \eta$ and $e_{\dot{\eta}} = \dot{\eta} - \dot{\eta}^d - \alpha_\eta\left(\eta^d - \eta\right)$ are offsets from the desired value η^d of state η, and α_η are control parameters. The x and y positions are controlled by adjusting the associated projections of U_z onto the XY axes, that is, $u_X U_z$ and $u_Y U_z$. The desired values of u_X and u_Y are specified by another set of backstepping controllers:

$$
\begin{aligned}
u_X^d &= (m/U_z)\left(e_x - \alpha_x\left(e_{\dot{x}} + \alpha_x e_x\right) - \alpha_{\dot{x}} e_{\dot{x}}\right) \\
u_Y^d &= (m/U_z)\left(e_y - \alpha_y\left(e_{\dot{y}} + \alpha_y e_y\right) - \alpha_{\dot{y}} e_{\dot{y}}\right)
\end{aligned}
\tag{5}
$$

Orientation controller. Since the systems thrust is assumed to be directly along the z axis of the body, u_X^d and u_Y^d prescribe desired pitch and roll for the inner loop:

$$\phi^d = -\sin^{-1}\left(u_Y^d \cos\psi - u_X^d \sin\psi\right) \tag{6}$$

$$\theta^d = \sin^{-1}\left(\frac{u_X^d \cos\psi + u_Y^d \sin\psi}{\sqrt{1-\left(u_Y^d \cos\psi - u_X^d \sin\psi\right)^2}}\right) \tag{7}$$

These values, alongside the desired yaw inputs, are then controlled by specifying the torques in their associated directions:

$$U_\phi \ = \frac{I_x}{l}\left(e_\phi - \frac{(I_y - I_z)}{I_x}\dot{\theta}\dot{\psi} - \alpha_\phi\left(e_{\dot{\phi}} + \alpha_\phi e_\phi\right) - \alpha_{\dot{\phi}} e_{\dot{\phi}}\right) \tag{8}$$

$$U_\theta \ = \frac{I_y}{l}\left(e_\theta - \frac{(I_z - I_x)}{I_y}\dot{\phi}\dot{\psi} - \alpha_\theta\left(e_{\dot{\theta}} + \alpha_\theta e_\theta\right) - \alpha_{\dot{\theta}} e_{\dot{\theta}}\right) \tag{9}$$

$$U_\psi \ = \frac{I_z}{l}\left(e_\psi - \frac{(I_x - I_y)}{I_z}\dot{\phi}\dot{\theta} - \alpha_\psi\left(e_{\dot{\psi}} + \alpha_\psi e_\psi\right) - \alpha_{\dot{\psi}} e_{\dot{\psi}}\right) \tag{10}$$

Finally the desired forces and torques are transformed into appropriate propeller speeds using the thrust and torque coefficients assuming a quadratic relationship with motor speed.

V. Experiments

Experimental Setup. Experiments were performed in an approximately rectangular $6\,m$ x $4\,m$ environment. A set of 6 OptiTrack Prime 17W cameras provide a ground truth pose estimate in the enclosed area, running at 120Hz, which is used for photorealistic camera image generation. Three sets of experiments were performed:

1) Visual state estimation and control in a baseline scenario involving an indoor environment
2) Visual state estimation and control in a challenging scenario involving flying through a window
3) Camera parameter sweep to investigate estimation accuracy for various camera parameters

The first two experiments were conducted both in simulated environments (using FlightGoggles) and in real environments, whereas the last experiment was performed in simulated environments. To keep the UAV within the flight cage drift is corrected by shifting the global desired trajectory to match the visual-inertial odometry (VIO) local frame after each loop.

Visual Navigation in Open Space. In total 42 experiments were performed in open space, 21 using the on-board camera on the UAV and 21 using our photorealistic image generation system to simulate a camera in real time. In all 21 experiments using the on-board camera and in 19 out of 21 experiments using the simulated camera the UAV traced out the desired trajectory with the VIO state estimate in the control loop for the full life of the battery (2-3 minutes). In the two simulated experiments that had to be ended early, Wifi network dropouts caused visual data to not reach the UAV, and the experiment was ended for safety. The reference trajectory flown for these experiments is an oval of length $2.8\ m$ and width $1.6\ m$, with a period of $3.5-3.8\ s$, for an average speed of ~2 m/s and a maximum speed of ~3 m/s on the long sides of the oval.

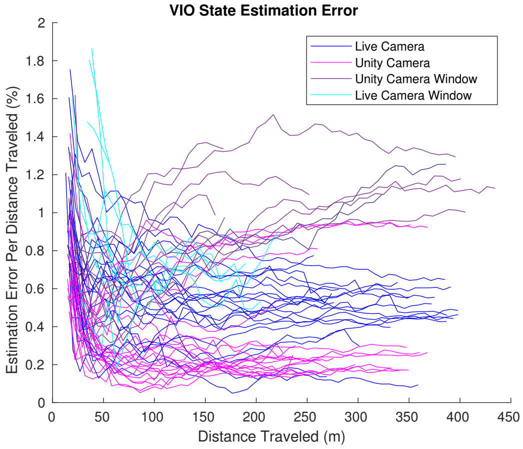


Fig. 5: Error in VIO state estimate of the UAV's position as a percentage of the distance traveled by the UAV. Flights flown with the on-board camera are shown in blue (no window) and cyan (window) while flights flown using camera images rendered in Unity are shown in magenta (no window) and purple (window); a total of 21 flights were flown with each style of camera without a window, and 8 and 10 flights with the real and simulated cameras through a window.

The estimation error as a function of distance traveled for all 42 experiments is shown in Figure 5. Since this system has no loop closures the initial estimation error during take-off cannot be recovered, resulting in the higher error percentages at the beginning of the flight when little distance has been traveled. Once the UAV starts flying its trajectory the estimation error remains below 1% (1 cm error for every 1 m flown) in all experiments. Note that the tracking of features is intentionally limited to 3 seconds, both to maintain low computation costs and to better mimic flying through an ever changing environment where no features can be seen continuously. The VIO state estimate was continuously in the control loop without assistance from motion capture for all 42 flights, demonstrating a stable and accurate state estimate.

Visual Navigation through a Window. Our second set of experiments involves flying through a window ($0.90\,m$ x $0.60\,m$, approximately twice the size of the UAV) with the VIO state estimate in the loop. Flying through windows presents a challenging problem for monocular VIO systems with forward facing cameras, as the visual element of the VIO system relies on motion to triangulate features. When flying through a window the only visual data linking state estimates on one side of the window to the other are those seen through the window, for which there is little tangential motion making triangulation inaccurate.

To the best of our knowledge the only two demonstrations of on-board, vision-based navigation through window openings come from Loianno *et al.* [38] and Falanga *et al.* [39]. In both cases the focus was on trajectory generation and control under uncertainty, and state estimation was only maintained for a single traversal of the window before landing. A key concern is the ability to continue flight after passing

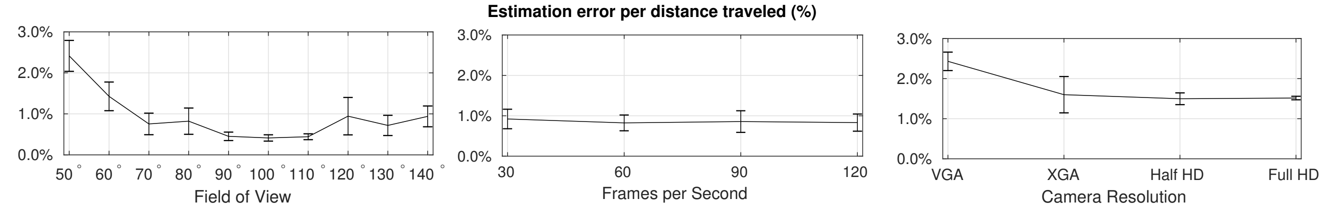
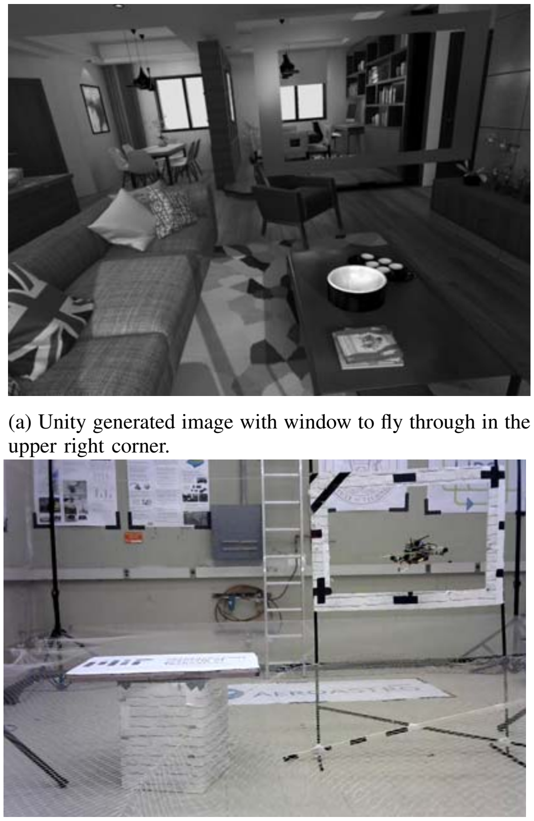


Fig. 6: Data from camera parameter testing using real-time photorealistic image simulation generated from logged flight data to assess VIO estimation performance for different camera types. Eight trials were performed for each sensor type. All trials were run in realtime using our simulation pipeline and an attached Jetson TX1. FOV trials were conducted with XGA (1024x768) resolution at 60 FPS. FPS trials were run at VGA (640x480) resolution and 80° field of view. Camera resolution trials were conducted at 50 FPS and 60° FOV.



(a) Unity generated image with window to fly through in the upper right corner.

(b) Image of UAV flying through a physical window using VIO in the loop based on its onboard camera and IMU.

Fig. 7: Images from VIO experiments, showing an image from a virtual on-board camera (top) and of our UAV flying through a window gap under VIO control (bottom)

through a window; therefore we sought to repeat the baseline experiment with a window in the path of the oval trajectory, although at a slower speed (average speeds of ~1.7 m/s and maximum speed of ~2.3 m/s). At each loop a simulated window detection occurred to set a new flight trajectory through the window, however this window detection was not used for state estimation.

Our photorealistic sensor simulation system provides the platform to develop our algorithms for window navigation. Developing algorithms with a physical window hazards numerous crashes as the system is developed, and the development of the system without a window or without visual navigation algorithms in the loop does not provide an accurate description of the performance of the UAV powered by visual navigation algorithms. Instead, our development environment allows for the visual effect of flying through a window, real dynamics, and real inertial measurements, without crashing on failure.

A total of 10 flights were performed with a simulated camera, constituting 361 traversals of the window, with 3 traversals resulting in a “crash” with the virtual window (crashes detected from the motion capture position of the UAV). The estimation error across the flights is shown in cyan in Figure 5.

Through experimentation with simulated imagery we found that a high keyframe rate with less feature data is necessary to both consistently bridge the gap created by flying through the window, and to quickly re-establish an accurate state estimate on the other side of the window. Based on these lessons, we performed the same experiment with a real window and the on-board camera. A total of 8 flights were performed, constituting 119 traversals of the window with 6 crashes/pilot take overs due to estimation divergence. Based on the analyzed data the lower success rate when using a real camera was due to a combination of noisier visual data than provided by the simulation system, and the additional computational load on the on-board computer of image acquisition slowing down the optimization rate. The noisier visual data can come from a combination of lower quality features in the world, motion blur of the live camera, and imperfections in the estimated camera model.

Camera Parameter Tests. In addition to allowing for testing in a variety of visual environments, our development environment also allows for rapidly evaluating sensor properties and configurations. For instance, we took a 70 second pre-recorded flight of our UAV flying an oval trajectory under motion capture and tested the VIO’s performance in real-time using real-world IMU measurements against a set of camera parameters spanning Field of View, Camera Resolution, and Frame Rate. See Figure 6 for a selection of results. These measurements are not meant as a declaration of the best camera to use for visual inertial navigation, but rather to show the capabilities of the system for rapid system prototyping to fit new challenges.

The ease of experimenting with this simulation system makes testing out sensor configurations in new flight scenarios easy and cost effective. While we have focused on a few parameters of the camera sensor itself, a wide range of other effects such as camera blur, scene lighting, feature richness, and accuracy of the camera model can easily be investigated.

VI. Conclusion

The work presented here demonstrates the capabilities of our fully integrated drone platform, its on-board visual inertial odometry system, and our novel real time visual simulation environment. Using the VIO system we can fuse inertial data and visual data (either real or simulated) in real time to perform closed loop control through agile maneuvers. We demonstrate this capability in an open room, and use the visual simulation system to safely develop our capabilities in the challenging scenario of flying through a window opening. The combined drone and visual simulation system provides a platform for rapid development of vision based algorithms in increasingly complex scenarios.

References

[1] J. Dong, L. Carlone, G. C. Rains, T. Coolong, and F. Dellaert, "4D mapping of fields using autonomous ground and aerial vehicles," in *2014 ASABE and CSBE/SCGAB Annual International Meeting*, 2014.

[2] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017.

[3] J. How, C. Fraser, K. Kulling, L. Bertuccelli, O. Toupet, L. Brunet, A. Bachrach, and N. Roy, "Increasing autonomy of UAVs," *IEEE Robotics and Automation Magazine*, vol. 16, no. 2, 2009.

[4] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments," in *Intl. Soc. Opt. Eng. (SPIE)*, 2009.

[5] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, may 2015, pp. 5303–5310.

[6] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, vol. 320, p. 240, 2012.

[7] G. Loianno and V. Kumar, "Vision-based fast navigation of micro aerial vehicles," in *Proc. SPIE, Micro- and Nanotechnology Sensors, Systems, and Applications*, 2016.

[8] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle." *J. of Field Robotics*, vol. 33, no. 4, pp. 431–450, 2016.

[9] 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.

[10] G. Loianno, G. Cross, C. Qu, Y. Mulgaonkar, J. Hesch, and V. Kumar, "Flying smartphones: Automated flight enabled by consumer electronics," *IEEE Robotics and Automation Magazine*, vol. 22, no. 2, pp. 24–32, 2015.

[11] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3234–3243.

[12] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4340–4349.

[13] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, *Playing for Data: Ground Truth from Computer Games*. Springer International Publishing, 2016, pp. 102–118.

[14] B. Kaneva, A. Torralba, and W. T. Freeman, "Evaluation of image features using a photorealistic virtual world," in *Intl. Conf. on Computer Vision (ICCV)*. IEEE, 2011, pp. 2282–2289.

[15] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, 2014.

[16] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154.

[17] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS—a modular Gazebo MAV simulator framework," in *Robot Operating System (ROS)*. Springer, 2016, pp. 595–625.

[18] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and Gazebo," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2012, pp. 400–411.

[19] D. G. Kottas, J. A. Hesch, S. L. Bowman, and S. I. Roumeliotis, "On the consistency of vision-aided inertial navigation," in *Intl. Sym. on Experimental Robotics (ISER)*, 2012.

[20] J. Hesch, D. Kottas, S. Bowman, and S. Roumeliotis, "Camera-IMU-based localization: Observability analysis and consistency improvement," *Intl. J. of Robotics Research*, vol. 33, no. 1, pp. 182–201, 2014.

[21] A. Mourikis and S. Roumeliotis, "A dual-layer estimator architecture for long-term localization," in *Proc. of the Workshop on Visual Localization for Mobile Platforms at CVPR*, June 2008.

[22] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial SLAM using nonlinear optimization," *Intl. J. of Robotics Research*, 2015.

[23] M. Bryson, M. Johnson-Roberson, and S. Sukkarieh, "Airborne smoothing and mapping using vision and inertial sensors," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2009, pp. 3143–3148.

[24] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry," *IEEE Transactions on Robotics*, pp. 1–20, 2016.

[25] R. Mur-Artal, J. Montiel, and J. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[26] M. Bloesch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2010, pp. 21–28.

[27] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, "Monocular vision for long-term micro aerial vehicle state estimation: A compendium," *J. of Field Robotics*, vol. 30, no. 5, pp. 803–831, 2013.

[28] 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.

[29] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-Direct Monocular Visual Odometry," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[30] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *J. of Field Robotics*, vol. 00, pp. 1–29, 2017.

[31] A. S. Huang, E. Olson, and D. C. Moore, "LCM: Lightweight communications and marshalling," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010, pp. 4057–4062.

[32] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1994.

[33] L. Kneip, M. Chli, R. Siegwart, *et al.*, "Robust real-time visual odometry with a single camera and an IMU," in *BMVC*, 2011.

[34] L. Kneip, S. Weiss, and R. Siegwart, "Deterministic initialization of metric state estimation filters for loosely-coupled monocular vision-inertial systems," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011, pp. 2235–2241.

[35] 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.

[36] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Institute of Technology, Tech. Rep. GT-RIM-CP&R-2012-002, September 2012.

[37] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2005, pp. 2247–2252.

[38] 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 Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2017.

[39] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017.