A Self-contained Teleoperated Quadrotor: On-board State-Estimation and Indoor Obstacle Avoidance

Marcin Odelga[1], Paolo Stegagno[2], Nicholas Kochanek[3], and Heinrich H. Bülthoff[1]

***Abstract*— Indoor operation of unmanned aerial vehicles (UAVs) poses many challenges due to the lack of GPS signal and cramped spaces. The presence of obstacles in an unfamiliar environment requires reliable state estimation and active algorithms to prevent collisions. In this paper, we present a teleoperated quadrotor UAV platform equipped with an on-board miniature computer and a minimal set of sensors for this task. The platform is capable of highly accurate state-estimation, tracking of desired velocity commanded by the user and ensuring collision-free navigation. The robot estimates its linear velocity through a Kalman filter integration of inertial and optical flow (OF) readings with corresponding distance measurements. An RGB-D camera serves the purpose of providing visual feedback to the operator and depth measurements to build a probabilistic, robo-centric obstacle model, allowing the robot to avoid collisions. The platform is thoroughly validated in experiments in an obstacle rich environment.**

I. Introduction

Multirotor unmanned aerial vehicles (UAVs) constitute an attractive platform for many robotics applications because of their ability to hover and their smooth operation at a wide range of velocities. The large interest in these platforms is reflected in the high number of publications in the field and the wide variety of available commercial products. The main limitations of UAVs are their short battery life and limited payload capacity, which limit the computational power and sensors that can be embedded on a platform and thus, the autonomy that is achievable. The ongoing development of miniature lightweight computers and sensors, however, has helped to overcome this issue and extend the complexity of possible applications.

Although human supervision is often required due to legal and safety reasons, operating an UAV can be greatly simplified, and performance similarly improved by endowing the UAV with autonomy [1], [2]. For example, during a teleoperation task, the operator may have limited situational awareness due to the reduced feedback provided by the system. Autonomous obstacle avoidance, especially when operating in tight and unstructured GPS-denied environments, can enhance visual inspection or search and rescue missions. Thus allowing for operation closer to objects of interest, and in more cluttered environments, leaving the operator free to focus on higher level goals [3], [4], [5].

[1]M. Odelga and H. H. Bülthoff are with the Max Planck Institute for Biological Cybernetics, Department of Human Perception Cognition and Action, Tübingen, Germany, {modelga, hhb}@tuebingen.mpg.de

[2]P. Stegagno is with the University of Rhode Island, pstegagno@uri.edu

[3]N. Kochanek is an undergraduate student at Harvard University, nkochanek@college.harvard.edu

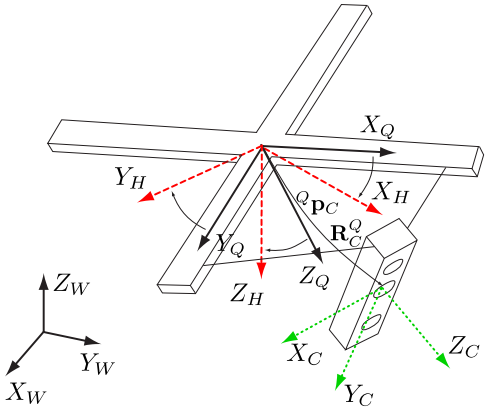

Fig. 1: Representation of the frames involved in the control and estimation of the quadrotor.

Depending on the objective and the amount of a priori knowledge of the system, we can distinguish several formulations of the navigation problem. In general, path planning addresses trajectory optimization, e.g., path length minimization or energy efficiency maximization, and usually under the assumption of full knowledge of the system and environment, including obstacles sizes and positions [6]. Teleoperation applications, however, consider mostly unknown and/or non-static environments. Moreover, when the UAV has only a generic goal (e.g., exploration of the given space), a path planning approach becomes ineffective.

To ensure collision-free navigation, obstacle avoidance is usually implemented as a reactive control law that adjusts the system's behavior given exteroceptive measurements. The most basic approach is the use of distance sensors emulating the behavior of mechanical bumpers that limit motion in the sensing directions [3], [7]. Gageik et al. [8] propose an array of complementary ultrasonic and infrared distance sensors to enhance their system's robustness. An evolution of this concept uses a 360° laser range finder to create a 2D occupancy grid map used to determine obstacle-free paths for a UAV [9]. A natural 3D extension can also be obtained by rotating the sensor via servomotors [3].

In monocular camera based approaches, instead of building a 3D model of the environment, researchers have tried to avoid obstacles based on optical flow methods [10], or by determining the size expansion ratios of obstacles from feature tracking in the field of view (FOV) of the sensor [11]. The perspective from a monocular camera, however, does not provide any metric correlations between the detected features and the robot other than the relative direction. Stereo cameras enable generation of disparity images based on

Fig. 2: Our quadrotor platform.

epipolar geometry and hence the distance estimation. In [12], a reactive avoidance is proposed based on local planning using a U-map, an accumulated histogram along the columns of the disparity image.

For most of the aforementioned works, the limiting factor was the onboard computational power. The transition from sensor arrays to 2D grid maps and frame-to-frame image analysis was possible not only thanks to the development of more advanced algorithms, but also because of to the recent progresses in mobile CPUs. In addition, the availability of lightweight sensors has extended the data gathering and processing abilities of UAVs with limited payload.

More recent works take advantage of smaller size and higher performance of miniature computers by employing more sophisticated methods not only in map building but also for obstacle avoidance. In [13], an octree-based 3D path planning algorithm using a state lattice concept is employed to find an optimal trajectory. Gohl et al. [14] equipped their platform with four stereo cameras allowing the robot to have an omnidirectional view of the environment. Detected obstacles are stored as points in a local, spherical coordinate frame that is transformed as the robot moves with the estimate of it's state.

The aforementioned works could also be categorized in terms of approaches they take to estimate their state. In general, the full knowledge of drift-free position is not critical for safe teleoperation [15], however, the robot must be able to reliably estimate its own velocity. Such an approach, usually based on inertial measurement unit (IMU) and optical flow integration ([4], [12]), requires fewer resources and quite often can be executed on an embedded microcontroller [16]. Although the more resource-intensive methods such as visual odometry ([5], [8]) or visual-inertial odometry ([14], [13]) can provide low drift position estimates they also require higher computational power or even dedicated hardware.

II. Motivation and Contributions

In our previous work [1] we presented initial results in the development of a system for automatic obstacle detection, tracking and avoidance with a quadrotor equipped with a single RGB-D sensor [1].

In that paper, the obstacle state, expressed in a local, robo-centric coordinate system, is updated not only with the new depth measurements but is also propagated using the estimated state of the robot (i.e., its own velocity and 3D orientation). That approach allows the robot to avoid obstacles that are not in the direct, limited FOV of the camera. Additionally, the 3D probabilistic representation of the obstacle state enables a complex, predictive based avoidance, not limited to the 2D plane. The predefined size of the obstacle state, in contrast to octree-based voxel grids that require complex ray casting, allows for rapid occupancy checks of any subregion.

In order to avoid collision, as our algorithm was designed for a teleoperation system, we took a passive approach with the robot only reacting to the commands of the operator and modifying them when necessary. In [1], we also validated our algorithm in experiments with single obstacles, simulating the estimate of the robot velocity through an external motion capture system.

As a consequential follow-up of that initial setup, in this work we have added an on-board state estimator that utilizes a cascade architecture to estimate the robot's orientation and velocity, using a complementary filter and a dual Kalman filter (KF) fed with the measurements of an IMU and OF sensor with a down-pointing sonar rangefinder, respectively. As the Kalman filter assumes zero-mean noise of the signals, we have improved its performance by introducing an extended set of filter's equations that estimates accelerometer drift in the pre-flight phase. This approach uses minimal computational resources, does not add significant mass and runs on the same on-board computational unit as our main algorithm and the teleoperation framework.

As compared to [1], the system presented in this paper is independent from external infrastructures and UAV can be flown in practically any indoor areas, while the limiting factor for outdoor obstacle detection is the sensing technology.

In order to achieve this result, we have tackled a number of practical issues, and adjustments to the system have been necessary. As expected, the on-board velocity estimation required improved calibration of the IMU, for which we detail two procedures in this work, an optimization based approach for general calibration using manually collected data and an automatic, pre-flight calibration for both the accelerometer and gyroscope. Another major improvement to account for the change from a reliable low-noise velocity estimate to a higher-noise estimation is the introduction of an active term in the obstacle avoidance that also compensates for small delays in the velocity control loop.

The main contributions of this work are therefore (i) an improved version of our previous algorithm which is suitable for higher-noise velocity estimates and delays in the control loop, (ii) a detailed implementation of a dual KF state estimator with on-line, pre-flight bias estimation, (iii) the implementation of a portable teleoperation setup with full on-board sensing and computation, and (iv) a comprehensive experimental validation of obstacle avoidance and state estimation in teleoperation.

III. System Setup

In this work we consider a multirotor UAV equipped with an IMU, an optical flow sensor with a dedicated

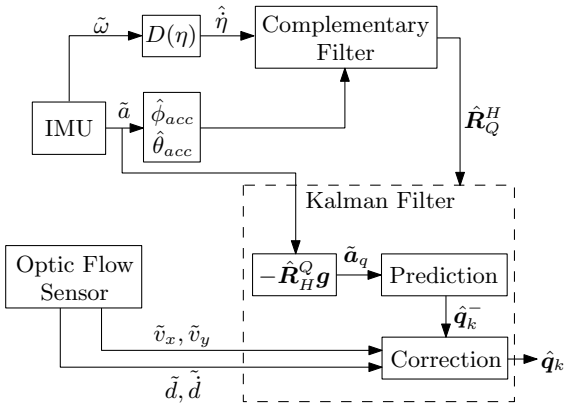

Fig. 3: Block diagram of the estimation scheme.

rangefinder and an RGB-D camera for environment sensing. The relevant quantities are represented in the reference frames shown in Fig. 1. The quadcopter frame of reference $Q : \{O_Q, X_Q, Y_Q, Z_Q\}$, represented in the common aerospace North-East-Down (NED) notation is used to express the robot's position and orientation in an arbitrary, North-West-Up (NWU), world reference frame $W : \{O_W, X_W, Y_W, Z_W\}$. Q is attached to the middle point of the robot, ideally its center of mass, the X_Q axis defines the front direction and the Z_Q axis points downward.

The robot's orientation is represented in the *roll-pitch-yaw* notation, where ϕ, θ, ψ denote the corresponding angles and

$$\mathbf{R}_Q^W = \mathbf{R}_x(\pi)\mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \in SO(3), \qquad (1)$$

is the rotation matrix, that defines the orientation of Q in W. $\mathbf{R}_x(\cdot)$, $\mathbf{R}_y(\cdot)$, $\mathbf{R}_z(\cdot)$ are the basic rotation matrices, which represent the elemental rotations around the X, Y and Z axes respectively. The $\mathbf{R}_x(\pi)$ matrix describes the transformation from NED to NWU notation.

The state of the system, and commands sent to the robot are expressed in a local, robo-centric, horizontal frame $H : \{O_H, X_H, Y_H, Z_H\}$ in which the $X_H Y_H$ plane is parallel to the world $X_W Y_W$ plane. It is defined such that $O_H \equiv O_Q$ and its orientation differs from $\mathbf{R}_Q^W$ only by the yaw angle and the NED-NWU conversion, $\mathbf{R}_H^Q = \mathbf{R}_y(\theta)\mathbf{R}_x(\phi)$. The position of the quadrotor and its yaw angle expressed in such a frame, ${}^H\mathbf{p}_Q$ and ${}^H\psi_Q$, respectively, are equal to zero.

Lastly, we represent the camera frame in which the sensor captures images as $C : \{O_C, X_C, Y_C, Z_C\}$. The position and orientation of C in Q, i.e. ${}^Q\mathbf{p}_C$ and $\mathbf{R}_C^Q$, are constant extrinsic parameters of the camera, obtained in an off-line calibration.

IV. State Estimation

The state of the platform is defined as a set of quantities that need to be estimated as they are required for the purpose of control. In the context of teleoperation, a robo-centric approach has been proven convenient [15] to express the state of the system and the operator's commands (i.e., the desired velocity). Therefore, the state of the system, expressed in the horizontal frame H,

$${}^H\mathbf{q} = \begin{bmatrix} {}^H\dot{x}_Q & {}^H\dot{y}_Q & {}^H\dot{z}_Q & \phi & \theta & \dot{\psi} \end{bmatrix}^T, \qquad (2)$$

includes the linear velocity of the robot, the roll and pitch angles and the angular velocity around the Z_H axis.

A. Inertial Sensor Model

The on-board IMU provides measurements of the linear acceleration and the angular velocity in Q. In principle, the IMU can also include a magnetometer to collect absolute heading measurements. However, in indoor settings, due to disturbances caused by power lines, ferromagnetic structures, and robotic motors, the measurements provided by digital compasses are unreliable [17]. Therefore, in most indoor mobile robotics research, e.g., [18], the heading is considered as an unknown quantity that requires estimation using localization filters.

Considering sensor imperfections and measurement errors, we model the measurements from the accelerometer as

$$\widetilde{\boldsymbol{a}} = \boldsymbol{b}_{as}(\boldsymbol{a} + \boldsymbol{\nu}_a) + \boldsymbol{b}_{ao}, \qquad (3)$$

and the measurements from the gyroscope as

$$\widetilde{\boldsymbol{\omega}} = \boldsymbol{b}_{\omega s}(\boldsymbol{\omega} + \boldsymbol{\nu}_\omega) + \boldsymbol{b}_{\omega o}, \qquad (4)$$

where, for $\boldsymbol{i} = \{\boldsymbol{a}, \boldsymbol{\omega}\}$, $\widetilde{\boldsymbol{i}}$ denotes the measured value, $\boldsymbol{i}$ the real value of acceleration and angular rate respectively, $\boldsymbol{b}_{is}$ and $\boldsymbol{b}_{io}$ represent, respectively, scaling and offset biases, and $\boldsymbol{\nu}_i$ is a zero-mean random error.

Although the zero-mean error $\boldsymbol{\nu}_i$ can be filtered out in the estimation process, the offset and scaling biases must be estimated in order to decrease their negative effect on the state estimate (2).

For the purpose of this work, we have employed the ellipsoid fitting method [19] to calibrate the accelerometer. This process requires samples of a known acceleration, e.g., gravity, in at least six widely-spread orientations and was performed off-line. The obtained calibration results are presented in Table I. In principle, the same method could be applied to estimate the gyroscope bias. That would require, however, to rotate the sensor with a known angular velocity around different axes.

TABLE I: Estimated accelerometer biases

	$\boldsymbol{b}_{as}$	$\boldsymbol{b}_{ao}$
x	0.9963	0.0028
y	1.0053	0.0096
z	1.0098	0.0517

Our experiments showed that the factory calibration of the gyroscope is accurate enough to neglect the scaling factor (i.e., $\boldsymbol{b}_{\omega s} \simeq 1$). The gyroscope's offset $\boldsymbol{b}_{\omega o}$ can be estimated by averaging steady measurements over a sampling period τ_s, which for the platform presented in this work, is performed every time during the on-ground phase with motors switched off. The sampling period is set to 2 s, which corresponds to 1000 samples at 500 Hz.

We introduce a set of bias-corrected measurements, $\bar{\boldsymbol{a}}$ and $\bar{\boldsymbol{\omega}}$, for the acceleration and angular velocity, respectively.

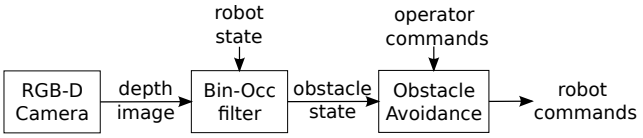

Fig. 4: Block diagram of the navigation system.

Hence, these new quantities, assuming an accurate bias estimation, should be affected only by the random, zero-mean noise:

$$\bar{\boldsymbol{a}} = \frac{\widetilde{\boldsymbol{a}} - \boldsymbol{b}_{ao}}{\boldsymbol{b}_{as}} = \boldsymbol{a} + \boldsymbol{\nu}_a, \tag{5a}$$

$$\bar{\boldsymbol{\omega}} = \frac{\widetilde{\boldsymbol{\omega}} - \boldsymbol{b}_{\omega o}}{\boldsymbol{b}_{\omega s}} = \boldsymbol{\omega} + \boldsymbol{\nu}_\omega. \tag{5b}$$

In addition, considering the non-inertial character of the body frame Q, the measured acceleration consists not only of the quadrotor's self-acceleration $\boldsymbol{a}_q$ but also the gravity term $\boldsymbol{g}$. Thus, the following relation completes the IMU model:

$$\boldsymbol{a} = \boldsymbol{a}_q - \boldsymbol{g} = \boldsymbol{a}_q - \boldsymbol{R}_W^Q \boldsymbol{g}_W, \tag{6}$$

where $\boldsymbol{g}_W = [0,\ 0,\ g]^T$.

B. Complementary Filter

The robot's state is estimated in a cascade scheme shown in Fig. 3 with two main components for simultaneous estimation of the platform's orientation (attitude) and linear velocity.

In order to estimate the platform's orientation, i.e., the roll and pitch angles ϕ, θ, we have used a complementary filter [20], whose principle we summarize here for completeness. This algorithm has the form of a low-pass filter, in which the angles are integrated using a proper projection of the gyro readings and corrected using the orientation of the gravity vector, estimated from the filtered accelerometer readings. First, the gyro readings, angular velocity in the sensor (body) frame $\bar{\boldsymbol{\omega}}$, must be properly transformed to obtain the roll, pitch and yaw rates $\dot{\boldsymbol{\eta}} = [\dot{\phi},\ \dot{\theta},\ \dot{\psi}]^T$:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{D}(\boldsymbol{\eta})\bar{\boldsymbol{\omega}}, \tag{7}$$

where $\boldsymbol{D}(\boldsymbol{\eta})$, the transformation matrix, has the following form:

$$\boldsymbol{D}(\boldsymbol{\eta}) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix}. \tag{8}$$

The notation s_δ, c_δ and t_δ indicates the $\sin$, $\cos$ and $\tan$ of a generic angle δ. Then, using the accelerometer measurement model (6):

$$\bar{\boldsymbol{a}} = \boldsymbol{R}_W^Q \boldsymbol{g}_W + \boldsymbol{a}_q = \begin{bmatrix} s_\theta \\ -c_\theta s_\phi \\ -c_\theta c_\phi \end{bmatrix} \boldsymbol{g}_W + \boldsymbol{a}_q, \tag{9}$$

and assuming that the acceleration of the platform $\boldsymbol{a}_q$ is zero (or negligible with respect to the gravity acceleration $\boldsymbol{g}_W$), the attitude can be estimated as:

$$\begin{aligned} \hat{\phi}_{acc} &= \operatorname{atan}_2(-\bar{a}_y, -\bar{a}_z), \\ \hat{\theta}_{acc} &= \operatorname{atan}_2\left(\bar{a}_x, \sqrt{\bar{a}_y^2 + \bar{a}_z^2}\right). \end{aligned} \tag{10}$$

Thus, the complementary filter equations with gain α are:

$$\begin{aligned} \hat{\phi}_k &= (1-\alpha)(\hat{\phi}_{k-1} + \dot{\phi}_k \Delta t) + \alpha \hat{\phi}_{acc}, \\ \hat{\theta}_k &= (1-\alpha)(\hat{\theta}_{k-1} + \dot{\theta}_k \Delta t) + \alpha \hat{\theta}_{acc}. \end{aligned} \tag{11}$$

C. Kalman Filter

The second component of the estimation system is a Kalman filter for the platform's linear velocity in the horizontal frame

$${}^H\dot{\boldsymbol{p}}_Q = \begin{bmatrix} {}^H\dot{x} & {}^H\dot{y}_Q & {}^H\dot{z}_Q \end{bmatrix}^T = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & d \end{bmatrix}^T. \tag{12}$$

Since the optical flow sensor is also equipped with an echo sonar sensor to provide the metric reference for the flow based velocity estimate, we have extended the filter's state to also include the distance to the ground d. Hence, the state vector of our Kalman filter is

$$\boldsymbol{q}_k = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & d \end{bmatrix}^T. \tag{13}$$

The filter's state equation, representing the integration of accelerometer measurements, has the form

$$\hat{\boldsymbol{q}}_k = \boldsymbol{A}_k \hat{\boldsymbol{q}}_{k-1} + \boldsymbol{B}_k \boldsymbol{u}_k, \tag{14}$$

where $\hat{\cdot}$ indicates an estimate, and, at instant k,

$$\boldsymbol{A}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \Delta t & 1 \end{bmatrix}, \boldsymbol{B}_k = \begin{bmatrix} \Delta t \hat{\boldsymbol{R}}_{Q_k}^{H_k} \\ 0 \quad 0 \quad 0 \end{bmatrix}$$
$$\boldsymbol{u}_k = \bar{\boldsymbol{a}}_k - (\hat{\boldsymbol{R}}_{Q_k}^{H_k})^T \boldsymbol{g}_W. \tag{15}$$

The estimate resulting from (14) is inherently encumbered with an accumulating error and noise resulting from inaccuracy in the estimation of system's orientation $\hat{\boldsymbol{R}}_{Q_k}^{H_k}$. The estimate's precision and accuracy can be increased using additional measurements, i.e., the optical flow and the distance to the ground [21], [22].

The optical flow measurement

$$\widetilde{\boldsymbol{z}}_{flow} = \begin{bmatrix} \tilde{v}_x \\ \tilde{v}_y \end{bmatrix} \tag{16}$$

where $\tilde{\mathbf{v}}$ and f represent the optical flow measurement (with rotation compensation [23]) and the sensor's focal length, respectively, has the following measurement model,

$$\widetilde{\boldsymbol{z}}_{flow} = \begin{bmatrix} \tilde{v}_x \\ \tilde{v}_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \boldsymbol{q}_k = \boldsymbol{C}_{flow} \boldsymbol{q}_k. \tag{17}$$

To correct the estimates of the horizontal velocity $\dot{z}$ and distance to the ground d, we use measurements from the echo sonar with the corresponding measurement model,

$$\widetilde{\boldsymbol{z}}_{dist} = \begin{bmatrix} \tilde{d}_k \\ \frac{\tilde{d}_k - \tilde{d}_{k-1}}{\Delta t} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \boldsymbol{q}_k = \boldsymbol{C}_{dist} \boldsymbol{q}_k, \tag{18}$$

It is important to note that the OF velocity measurement (16) is computed from the flow value $\tilde{\mathbf{v}}$ using the estimate of the ground distance $\hat{d}$ that, thanks to the filtering process, has lower covariance than the direct measurement from the echo sonar $\tilde{d}$.

D. On-ground Bias Estimation

Throughout our experiments, we noticed that the accelerometer biases (3) tend to drift slightly between consecutive tests. The Open/LibrePilot community[1] confirmed our observations claiming that the IMU chip is sensitive to mechanical stress applied to the controller board mounted on a flying platform. Therefore, the biases can change due to vibrations and impacts during landings.

To mitigate this issue, we have employed an additional on-ground bias estimation based on an extension to our Kalman filter's state model (14):

$$\boldsymbol{q}_k^\star = \begin{bmatrix} \boldsymbol{q}_k^T & \boldsymbol{b}_{ao}^T \end{bmatrix} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & d & b_x & b_y & b_z \end{bmatrix}^T, \quad (19a)$$

$$\boldsymbol{A}^\star = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & & & \\ 0 & 1 & 0 & 0 & & -\Delta t \hat{\boldsymbol{R}}_{Q_k}^{H_k} & \\ 0 & 0 & 1 & 0 & & & \\ 0 & 0 & \Delta t & 1 & 0 & 0 & 0 \\ \hline & \boldsymbol{0}_{3\times 4} & & & & \boldsymbol{I}_{3\times 3} & \end{array}\right], \quad (19b)$$

$$\boldsymbol{B}^\star = \begin{bmatrix} \Delta t \hat{\boldsymbol{R}}_{Q_k}^{H_k} \\ \boldsymbol{0}_{4\times 3} \end{bmatrix}. \quad (19c)$$

This bias estimation utilizes the fact that when the platform is on the ground its state is constant (in particular, its velocity is equal to zero) to correct the biases with assumed zero measurements in (16) and (18).

Once the robot is commanded to lift-off, the estimation system switches from (19) to the reduced form (15).

V. Navigation System Architecture

The obstacle detection and tracking part of our algorithm serves primarily as a means to extend the limited field of view of the camera by including regions that are not instantly visible. Using filtering, we can extend the knowledge of obstacles in the vicinity of the robot and reduce measurement noise as well.

The block diagram of the navigation system is shown in Fig. 4; the principle is that the operator's commands are being altered by the *Obstacle Avoidance* block given the knowledge of the estimated obstacle state by the *Bin-Occupancy* filter [24].

A. Obstacle Detection and Tracking

The algorithm detects obstacles using the depth images provided by the camera and represents them in a quantized, and bounded, surveillance region S, in the cylindrical coordinate frame $M : \{O_M, P_M, \Psi_M, Z_M\}$, defined such that $O_M \equiv O_H$ and $Z_M \equiv Z_H$. The P_M, Ψ_M coordinates are, respectively, the radial and the azimuth distances.

The surveillance region S is defined as a set of bins b_i, with constant size in M, $\Delta\rho \times \Delta\psi \times \Delta z$, and the obstacle state is represented as the probability of bins $b_i \in S$ being occupied. In order to extend the knowledge of the obstacles to the region of S outside the field of view of the camera, the obstacle state is propagated with the estimated velocity of the robot.

[1] https://forum.librepilot.org/

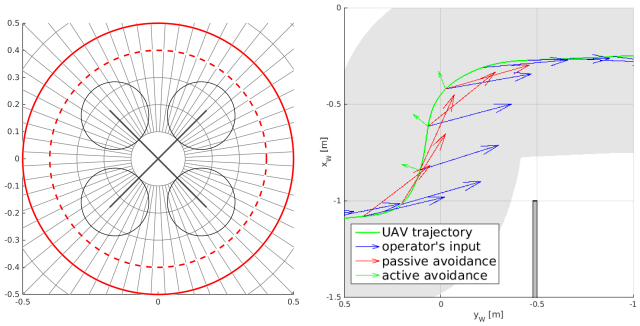

Fig. 5: Top view (left) of the bin representation and, in red, the restricted area; a top view (right) of a part of robot's trajectory, in gray, projection of the restricted area.

The choice of the cylindrical coordinate system to express the obstacle state was motivated by the fact that in such a system, the volume of the bins increases with the radial distance. This in turn corresponds to the change of the area that each pixel of the depth images represents. The pre-fixed, limited, size of S with a constant number of bins ensures finite computational time and memory demand.

B. Obstacle Avoidance

To ensure collision-free navigation, the obstacle avoidance part of the algorithm ought to prevent obstacles from entering the region of S occupied by the robot. We define the probability of collision as a joint probability of occupancy over a subset of bins $b_i \in S_R \subset S$, where S_R represents the part of the surveillance region associated to the robot. The boundary of the restricted area S_R is represented in the right part of Fig. 5 with the red continuous circle.

Assuming that collisions with obstacles in different bins are independent, we define the probability of collision with obstacles in S_R at instant k as

$$1 - \prod_{i \in S_R} \Big(1 - p(U_k(i))\Big), \quad (20)$$

where $p(U_k(i)) = p_i$ is the probability of bin i being occupied.

In general, the two main sources of possible collision (i.e., an obstacle moving into S_R) are:

a) the operator driving the robot towards an occupied area,
b) uncommanded drift of the robot.

In our previous work [1], we specifically addressed the first case.

The algorithm described there and employed also in this work foresees possible collisions based on the user input and the current obstacle state. The reference velocity for the robot is obtained by solving an optimization problem over

N steps:

$$\begin{aligned}\underset{\mathbf{u}_t}{\operatorname{argmin}} J = &\sum_{t=1}^{N} w_t \left(1 - \prod_{i \in S_R} \Big(1 - p(U_{k+t}(i)|\mathbf{u}_t)\Big)\right) \\ &+ w_\psi \cdot |u_\psi| \\ &+ w_z \cdot |u_z|,\end{aligned} \tag{21}$$

where $\mathbf{u}_t = (u_\rho, u_\psi, u_z)^T$ is the control input expressed in the cylindrical coordinates M and represents a transformation of the obstacle state, and $p(U_{k+t}(i)|\mathbf{u}_t)$ is the probability of occupancy given the input $\mathbf{u}_t$. Parameters w_t, w_ψ and w_z are the weights that correspond to the probability of collision at step N, change of direction, and change of elevation, respectively. Thus, the algorithm produces a set of N optimal control inputs $\mathbf{u}_t^\star$ that ensure obstacle avoidance and minimize the change of user input's magnitude, direction, and altitude of the robot. The command is then tracked using the controller described in [1].

It is a *passive* approach (i.e., no action is performed in absence of an operator command), and, as proven in the previous experiments, it is valid under the assumption of an accurate state estimation and fast control.

C. Active Avoidance

In the current on-board implementation, two main factors contribute to the second cause of possible collision related to the presumably lower quality of the on-board velocity estimate with respect to the estimate provided by an external motion capture system. First, the on-board estimate is affected by larger noise and may be biased for some time. As a result the quadrotor will naturally drift in the direction of the error.

The second source of uncommanded drift is related to the delay in the velocity control loop. As a lower quality estimate of the velocity is now available, we need to reduce the gains of the velocity controller, making the system slower to respond to the reference velocity.

In order to tackle this issue we have individualized a subset of S_R at its boundary S_{Rb} (between the solid and dashed red line in the right part of Fig. 5), thus, we can extract the occupied bins in S_{Rb} as: $S_O = \{b_i \in S_{Rb} : p(U_k(i) > 0)\}$. The algorithm actively checks this region of the obstacle state for possible obstacles and in a case of detection performs a repulsive avoidance action.

The active avoidance velocity command $\boldsymbol{u}_A$ is computed in the opposite direction than the detected obstacles, proportionally to the possibility of collision according to:

$$\boldsymbol{u}_A = \begin{bmatrix} u_{Ax} \\ u_{Ay} \end{bmatrix} = \sum_{b_i \in S_O} \frac{-a \cdot p_i}{1 - \prod_{b_i \in S_O} (1 - p_i)} \cdot \frac{\boldsymbol{x}_{b_i}}{|\boldsymbol{x}_{b_i}|}, \tag{22}$$

where $\boldsymbol{x}_{b_i}$ is the (x, y) coordinate of bin i in H and a is a parameter that defines the magnitude of the repulsive velocity.

The vector $\boldsymbol{u}_A$ is then added to the reference command obtained from (21) and the new command is sent to the robot. In the worst case, when there is no motion that clears S_R from the obstacles, the produced command minimizes the possibility of collision (20).

VI. Experimental Setup

Mechanically, the platform is based on a MK-Quadro quadcopter from MikroKopter, which consists of a frame with four 10 inch propellers powered by brushless motors with motor controllers. We have retrofitted the robot with a low-level flight controller based on a CopterControl3D board, developed in the OpenPilot project, equipped with a 16-bit digital IMU, i.e., an accelerometer and gyroscope, whose measurements are used for the state estimation. The flight controller board runs our original software with the Near-Hovering controller [25] to track the desired attitude (the roll and pitch angles, and the yaw rate) and thrust commands driving the brushless motor controllers over a standard I^2C bus.

The main computational unit is a smartphone-grade single-board computer Odroid-XU3 running the ROS-based teleoperation framework, TeleKyb [2]. It communicates wirelessly with a ground station PC, transmitting the visual feedback for the operator (color images from the camera) and receiving joypad and Omega.6 haptic device inputs. Using the joypad buttons, the operator can easily switch between different modes of operation (turn on the motors, initiate lift-off, switch to the haptic control mode, etc.) and give the desired velocity with the haptic device while receiving force feedback as described in more detail in [1].

The main on-board computer communicates with the flight controller over a serial connection with two software channels. On one channel, it receives the IMU data with a 500 Hz sample rate, while the second one is used to send the commands, receive low-rate status data (e.g., battery voltage level) and change some low-level controller settings (e.g., the controller gains).

The platform has two optical sensors, an RGB-D camera (Asus Xtion Pro) mounted horizontally in the forward direction, which is used as the source of data for obstacle detection and to provide RGB visual feedback to the operator, and an optical flow sensor with a built-in echo sonar (PX4Flow, [23]) oriented downward. The complete platform depicted in Fig. 2 weights approximately 1.3 kg.

VII. Evaluation

We validated the performance of our approach to obstacle avoidance with on-board state estimation in different setups, including both, horizontal and vertical obstacles. We performed more than 50 experiments, in which the operator was able to guide the robot along the full preplanned path with a success rate of over 94%. Excessive motion of the platform, due to drift in the state estimation, caused early termination of two experiments, which were aborted as safety measure to prevent crashes. Although the robot drifted, it merely touched an obstacle, as our algorithm was still able to limit the platform's velocity and prevent major collisions.

In this section, we present selected results from our experiments with the following setting of the algorithm's

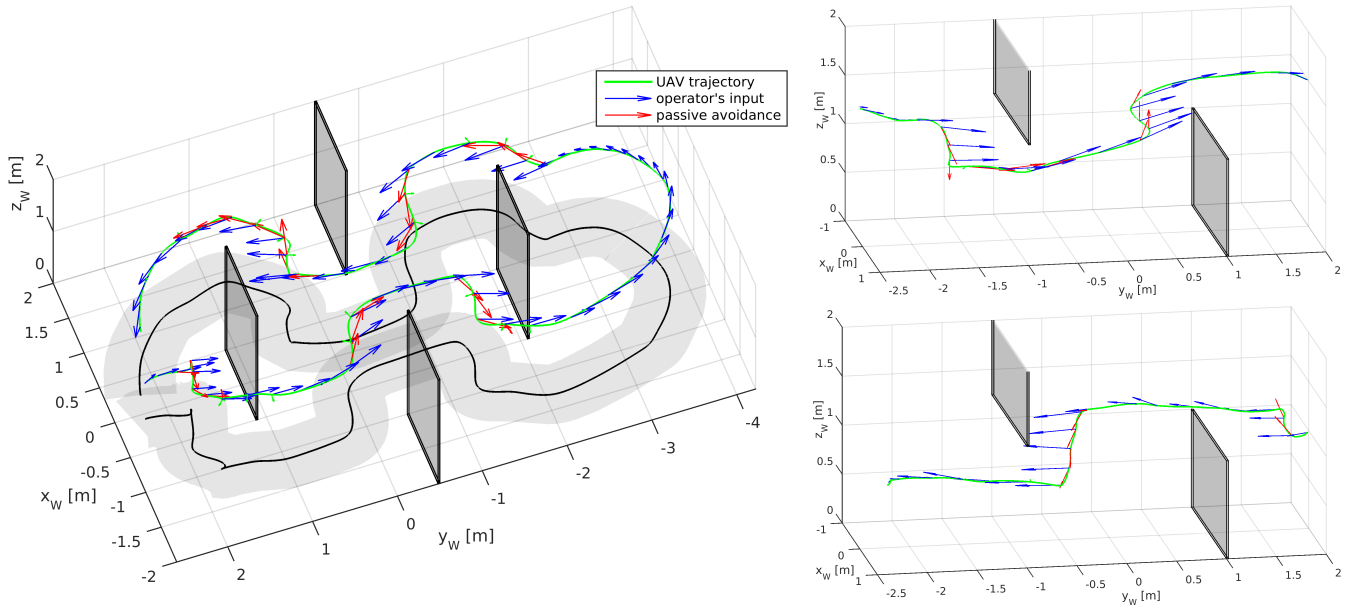

Fig. 6: 3D views of obstacles and trajectories of the robot in two experiments, (left) zig-zag between four vertical panels with the trajectory and robot size projection on the ground, (right) avoidance of two vertical obstacles, two parts of the same experiment.

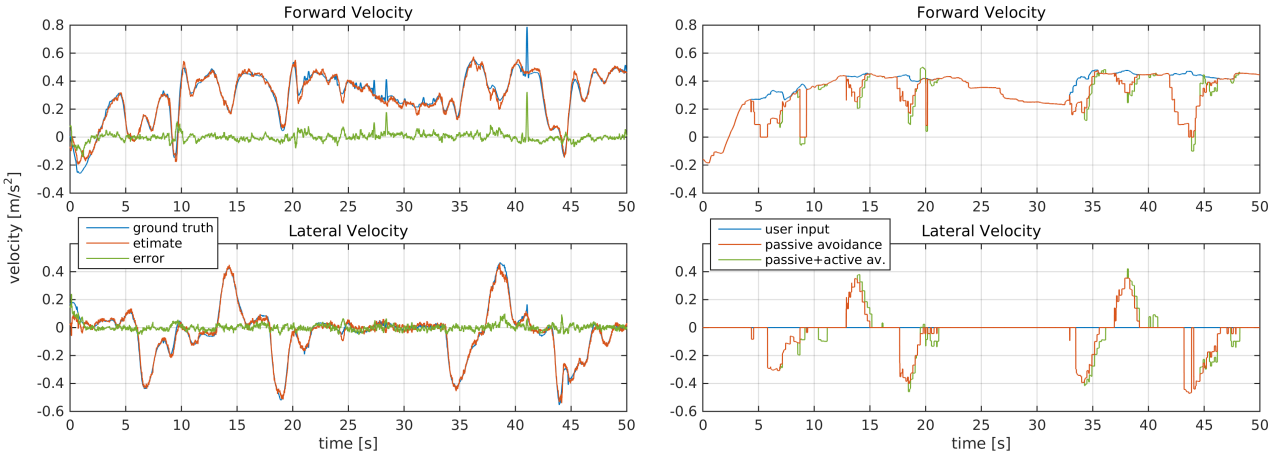

Fig. 7: Results from the zig-zag experiment, (left) state estimation, longitudinal and lateral velocities, (right) commanded velocity - user input and output of the algorithm - avoidance velocity.

parameters. With its radius doubled with respect to [1], the size of surveillance region S was set as:

$$\rho \in [0, 3\,m], \quad z \in [-1.5\,m, 1.5\,m],$$
$$\Delta\rho = 0.1\,m, \quad \Delta\psi = \frac{1}{30}\pi\,rad, \quad \Delta z = 0.1\,m\,,$$

and the restricted area S_R occupied by the robot as:

$$\rho \in [0, 0.5\,m], \quad z \in [-0.3\,m, 0.2\,m]\,.$$

The weights used in the cost calculation (21) are:

$$w_t = 10,\ w_\psi = 3,\ w_z = 3.$$

These values have been hand-tuned according to heuristic criteria through experiments by the operator to adjust the avoidance behavior.

The first experiment consisted of a corridor with four walls perpendicular to the desired trajectory. The gap between the walls was set such that it did not allow for a straight, collision-free motion from the one end to the other. The operator commanded the robot to fly along the corridor, make a u-turn after the last obstacle, and return. A full 3D view of the obstacle setup and the robot's trajectory together with the operator's and the avoidance commands can be seen in the left part of Fig. 6. Additionally, to illustrate the compactness of the setting, we show the ground projection of the trajectory with the gray shaded area corresponding to the robot's size. The left part of Fig. 5 shows a close view of avoidance of one of the walls with detailed presentation of corresponding commands. As can be seen in the right part of Fig. 7, the avoidance algorithm added additional lateral components to navigate between the obstacles, while limiting the longitudinal component when necessary.

The left part of Fig. 7 presents the results of the state estimation (x and y velocities) from this experiment together

TABLE II: Mean and standard deviation of the velocity estimation error

	μ_{error}	σ_{error}
$\dot{x}$	0.0004	0.0312
$\dot{y}$	0.0018	0.0231

with the corresponding ground truth data, obtained with an external tracking system. The mean and standard deviation of the error is shown in Table II.

The second experimental setup presented here, consists of two horizontal obstacles along the desired path of the robot. In this case, our algorithm altered the commanded velocity in the vertical direction resulting in motions below and above corresponding obstacles. The 3D trajectory of the robot can be seen in the right section of Fig. 6, split in two subplots (top going forward, bottom coming back after the u-turn) for better clarity.

The interested reader is invited to watch the accompanying video illustrating the presented experiments and the results of our algorithm.

VIII. Conclusions

In this work, we have developed a self-contained teleoperated quadrotor, in terms of sensor equipment and computations. The platform is able to estimate its state in an indoor, GPS-denied environment, using IMU and optical flow integration. It can also detect and avoid obstacles using a single RGB-D camera. In order to extend the limited field of view of the camera, we have implemented an obstacle tracking algorithm based on the Bin-Occupancy filter. The estimated obstacle state is used to predict possible collisions and to modify the velocity commanded by the operator to avoid obstacles. We have successfully performed multiple experiments in different obstacle setups to validate our approach, performing all computation on-board.

References

[1] M. Odelga, P. Stegagno, and H. H. Bülthoff, "Obstacle detection, tracking and avoidance for a teleoperated uav," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2984–2990.

[2] V. Grabe, M. Riedel, H. H. Bülthoff, P. R. Giordano, and A. Franchi, "The telekyb framework for a modular and extendible ros-based quadrotor control," in *2013 European Conference on Mobile Robots*, Sept 2013, pp. 19–25.

[3] M. Nieuwenhuisen, D. Droeschel, J. Schneider, D. Holz, T. Läbe, and S. Behnke, "Multimodal obstacle detection and collision avoidance for micro aerial vehicles," in *2013 European Conference on Mobile Robots*, Sept 2013, pp. 7–12.

[4] F. Bonnin-Pascual, A. Ortiz, E. Garcia-Fidalgo, and J. P. Company, "A micro-aerial platform for vessel visual inspection based on supervised autonomy," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 46–52.

[5] R. Mebarki, V. Lippiello, and B. Siciliano, "Nonlinear visual control of unmanned aerial vehicles in gps-denied environments," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 1004–1017, Aug 2015.

[6] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2520–2525.

[7] R. W. Houskamp, "Obstacle protection with unmanned vehicles," in *28th IEEE Vehicular Technology Conference*, vol. 28, March 1978, pp. 153–155.

[8] N. Gageik, P. Benz, and S. Montenegro, "Obstacle detection and collision avoidance for a uav with complementary low-cost sensors," *IEEE Access*, vol. 3, pp. 599–609, 2015.

[9] A. Ferrick, J. Fish, E. Venator, and G. S. Lee, "Uav obstacle avoidance using image processing techniques," in *2012 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, April 2012, pp. 73–78.

[10] P. C. Merrell, D.-J. Lee, and R. W. Beard, "Obstacle avoidance for unmanned air vehicles using optical flow probability distributions," *Mobile Robots XVII*, vol. 5609, pp. 13–22, 2004.

[11] A. Al-Kaff, Q. Meng, D. Martín, A. de la Escalera, and J. M. Armingol, "Monocular vision-based obstacle detection/avoidance for unmanned aerial vehicles," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, June 2016, pp. 92–97.

[12] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for mav flight," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 50–56.

[13] S. Xu, D. Honegger, M. Pollefeys, and L. Heng, "Real-time 3d navigation for autonomous vision-guided mavs," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 53–59.

[14] P. Gohl, D. Honegger, S. Omari, M. Achtelik, M. Pollefeys, and R. Siegwart, "Omnidirectional visual obstacle detection using embedded fpga," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 3938–3943.

[15] P. Stegagno, M. Basile, H. H. Bülthoff, and A. Franchi, "A semi-autonomous uav platform for indoor remote operation with visual and haptic feedback," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3862–3869.

[16] D. E. Schinstock, "Gps-aided ins solution for openpilot." [Online]. Available: http://wiki.openpilot.org/download/attachments/950387/INSGPSAlg.pdf

[17] S. Suksakulchai, S. Thongchai, D. M. Wilkes, and K. Kawamura, "Mobile robot localization using an electronic compass for corridor environment," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 5, 2000, pp. 3354–3359 vol.5.

[18] P. Stegagno, M. Cognetti, G. Oriolo, H. H. Bülthoff, and A. Franchi, "Ground and aerial mutual localization using anonymous relative-bearing measurements," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1133–1151, Oct 2016.

[19] S. Bektas, "Least squares fitting of ellipsoid using orthogonal distances," *Boletim de Ciências Geodésicas*, vol. 21, pp. 329–339, June 2015. [Online]. Available: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1982-21702015000200329&nrm=iso

[20] P. Martin and E. Salaün, "The true role of accelerometer feedback in quadrotor control," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 1623–1629.

[21] V. Grabe, H. H. Bülthoff, and P. R. Giordano, "On-board velocity estimation and closed-loop control of a quadrotor uav based on optical flow," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 491–497.

[22] A. Santamaria-Navarro, J. Solà, and J. Andrade-Cetto, "High-frequency mav state estimation using low-cost inertial and optical flow measurement units," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 1864–1871.

[23] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 1736–1741.

[24] O. Erdinc, P. Willett, and Y. Bar-Shalom, "The bin-occupancy filter and its connection to the phd filters," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4232–4246, Nov 2009.

[25] G. Gioioso, M. Ryll, D. Prattichizzo, H. H. Bülthoff, and A. Franchi, "Turning a near-hovering controlled quadrotor into a 3d force effector," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 6278–6284.