Preprint - final, definitive version available at http://www.ieeexplore.com/ accepted for ICRA2016 May 2016

Obstacle Detection, Tracking and Avoidance for a Teleoperated UAV

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

***Abstract*— In this paper, we present a collision-free indoor navigation algorithm for teleoperated multirotor Unmanned Aerial Vehicles (UAVs). Assuming an obstacle rich environment, the algorithm keeps track of detected obstacles in the local surroundings of the robot. The detection part of the algorithm is based on measurements from an RGB-D camera and a Bin-Occupancy filter capable of tracking an unspecified number of targets. We use the estimate of the robot's velocity to update the obstacles state when they leave the direct field of view of the sensor. The avoidance part of the algorithm is based on the Model Predictive Control approach. By predicting the possible future obstacles states, it filters the operator commands to prevent collisions. The method is validated on a platform equipped with its own computational unit, which makes it self-sufficient in terms of external CPUs.**

I. INTRODUCTION

Multirotor Unmanned Aerial Vehicles (UAVs) offer high maneuverability, hovering mode, vertical take-off/landing and other features that constitute an agile platform for many robotic applications. Unconstrained by ground conditions they can operate on places that are out of reach for other classical mobile robots, with applications such as exploration, inspection and surveillance. Considering the nature of the flying robots and due to legal regulations, operating in the real-world usually requires a skilled human operator/supervisor.

The high maneuverability and agility translates into a requirement for reliable control and good estimation of the platform's state. Recently, many commercial remote control (RC) UAVs have became available on the market. The classical RC transmitter, which normally allows to control the robot's acceleration in lateral directions by commanding the roll and pitch angles, is usually sufficient to perform open-air flights. A drift caused by integration of the inertial measurement can be compensated if the robot is in the line of sight of the operator and/or the GPS signal is available. However, if we are interested in operating in an unstructured, obstacle rich environment, where the GPS signal is unreliable and there is a limited space for maneuvers and errors, such a platform might not be sufficient.

Different feedbacks provided to the human operator, can facilitate the teleoperation task. The role of a haptic feedback, i.e. the force response of the input device to the user, and its positive impact on the operator situation awareness have been investigated in recent works (e.g. [1] and references therein,

[1]M. Odelga, P. Stegagno and H. H. Bülthoff are with the Max Planck Institute for Biological Cybernetics, Department of Human Perception Cognition and Action, Spemannstrasse 38, 72076 Tübingen, Germany {pstegagno, modelga, hhb}@tuebingen.mpg.de

[2]M. Odelga is also with the Chair of Cognitive Systems, Department of Computer Science, University of Tübingen, Tübingen, Germany

Fig. 1. Our quadrotor setup.

[2], [3] and [4]). Another important source of information are on-board cameras that provide visual feedback to the operator, widely used in teleoperation [5].

Nevertheless, in order to increase the number of potential users and applications, the command system of UAVs should be further simplified by embedding autonomous behaviors. The most basic are autonomous procedures such as lift-off and landing, hovering, and way-point navigation (following predefined trajectory). For widespread adoption of UAV teleoperation, more advanced features as obstacle detection and tracking as well as autonomous collision avoidance should be implemented to increase safety.

We can distinguish two main types of collision-free navigation. When we know in advance the environment and the desired position that the robot should reach, we usually deal with a path-planning problem. The goal is to optimize the trajectory in order to prevent collisions and fulfill other requirements like the shortest path [6]. The second type of navigation based on a reactive control law is usually implemented when either the desired state (position or velocity) or the environment is not known ([7], [8]). Teleoperation usually falls into this second category, since commands from the operator are not known in advance and the environment may be unknown.

Some previous works have dealt with autonomous obstacle avoidance for teloperated UAVs. In [2] and [3] the obstacle avoidance is performed using measurements from laser scanners and other range finders. Other works proposes the use of cameras and optical flow [9], [4]. In [5] readings from an RGB-D camera are used to feed an artificial potential based obstacle avoidance module.

In this paper, we present an obstacle avoidance algorithm for a collision-free teleoperation in unknown, obstacle-rich environment. Since we are using an RGB-D camera with limited field of view, we have integrated an obstacle tracking algorithm with a Model Predictive Control ([10]) inspired avoidance to modify the velocities commanded by the opera-

Preprint - final, definitive version available at http://www.ieeexplore.com/ accepted for ICRA2016 May 2016

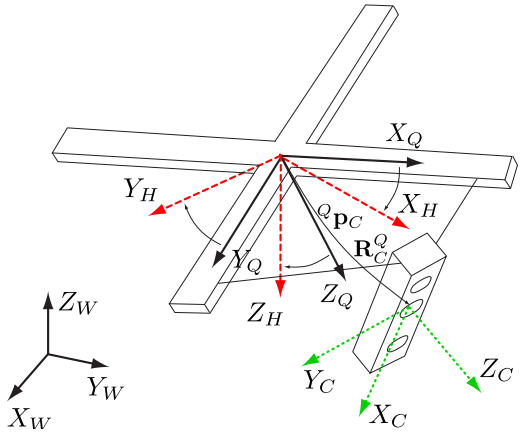

Fig. 2. A representation of all frames involved in the control and estimation of the velocity of the quadrotor.

tor. Usually obstacle tracking is a computationally expensive task due to the amount of data that needs to be processed. Since we are performing all the computations on-board, in our development we have given particular attention to the computational efficiency of our method. Our UAV platform, with the RGB-D sensor and the computational unit, is shown in Fig. 1.

The outline of this paper is as follows. In Section II we introduce our settings and system architecture. In Section III we explain the obstacle tracking. In Section IV we describe the obstacle avoidance method. In Section V we present our platform in details. In Section IV we provide experimental results and Section VII concludes the paper.

II. SYSTEM SETUP

In this work, we represent the relevant quantities in the reference frames defined below and shown in Fig. 2. The quadcopter frame of reference, defined as $Q : \{O_Q, X_Q, Y_Q, Z_Q\}$, is attached to the middle point of the robot, which is ideally its centre of mass. It is represented in the common aerospace North-East-Down (NED) notation (where the X_Q axis defines the front direction and the Z_Q axis points downward). The global frame of reference is defined in North-West-Up (NWU) convention as $W : \{O_W, X_W, Y_W, Z_W\}$ with the Z_W axis pointing in the opposite direction with respect to the gravity vector.

The robot position and orientation in the world frame are defined as ${}^W\mathbf{p}_Q \in \mathbb{R}^3$ and $\mathbf{R}_Q^W \in SO(3)$, respectively. With ${}^W\phi_Q$, ${}^W\theta_Q$, ${}^W\psi_Q$ we denote the roll, pitch and yaw angles that represent the rotation of the robot. Hence, the $\mathbf{R}_Q^W$ matrix is defined as follows:

$$\mathbf{R}_Q^W = \mathbf{R}_x(\pi)\mathbf{R}_z({}^W\psi_Q)\mathbf{R}_y({}^W\theta_Q)\mathbf{R}_x({}^W\phi_Q) \qquad (1)$$

where $\mathbf{R}_x(\cdot)$, $\mathbf{R}_y(\cdot)$, $\mathbf{R}_z(\cdot)$, the basic rotation matrices, 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.

It has been proved ([5]) that the robo-centric approach is convenient in teleoperation tasks, i.e. both, the state of the system and the commands are expressed in a local, horizontal frame in which the XY plane is parallel to the world $X_W Y_W$ plane. Hence, we introduce the frame $H : \{O_H, X_H, Y_H, Z_H\}$, defined such that its origin is in the center of mass of the UAV $O_H \equiv O_Q$ and its orientation differs from the orientation of the world frame only by the yaw angle and the NED-NWU conversion $\mathbf{R}_H^Q = \mathbf{R}_y({}^W\theta_Q)\mathbf{R}_x({}^W\phi_Q)$. In such a frame, the position of the quadrotor ${}^H\mathbf{p}_Q$ as well as its yaw angle ${}^H\psi_Q$ are equal to zero. Therefore, the relevant quantities for the control purpose in H are:

$${}^H\mathbf{q} = \left\{v_x, v_y, v_z, {}^W\phi_Q, {}^W\theta_Q, {}^W\dot{\psi}_Q\right\}, \qquad (2)$$

where ${}^H\mathbf{v}_Q = (v_x, v_y, v_z)^T$ is the velocity of the UAV. The state vector ${}^H\mathbf{q}$ includes also the roll and pitch angles and the angular velocity around the Z_H axis. We assume that a noisy estimate of the robot's state is available, either using external or on-board sensors. For example, the roll and pitch angles can be estimated using a complementary filter as the one described in [11], [12] using IMU measurements, while the velocity can be retrieved using a camera-IMU integration paradigm [13], [14].

The UAV is equipped with an RGB-D camera, which provides color and depth images at ∼30 fps. To express its measurements, we define $C : \{O_C, X_C, Y_C, Z_C\}$ – the frame in which the camera captures images. 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 and are assumed to be known through a previous calibration. In particular, in our platform the camera is mounted with a yaw angle of 45° in order to avoid occlusions by the propellers.

Navigation System Architecture

The navigation system, whose block scheme is shown in Fig. 3, consists of two main components. The first one is a local obstacle tracking algorithm based on the Bin-Occupancy filter first derived in [15]. In principle, we could use directly the stream of the RGB-D camera in order to perform the obstacle avoidance. However, the field of view (FOV) of the camera is limited with respect to the motion abilities of the UAV and the depth measurements are affected by noise and occlusions. Through the use of the filter, we can extend the knowledge of obstacles nearby the robot to include regions that are not instantly visible and reduce the measurement noise as well.

In order to detect obstacles, the Bin-Occupancy filter uses information from depth images and the roll and pitch angles. Additionally, it is able to propagate in time the state of the obstacles using the measured velocities. The filtered obstacles are then provided to the obstacle avoidance module, which modifies the operator commands when needed in order to enforce a collision-free flight. The output commands are provided to the quadrotor as the reference velocity and are tracked using the controller described in [16].

III. OBSTACLE DETECTION AND TRACKING

In this section we first provide some background on the Bin-Occupancy filter and then we explain the details of our implementation.

Preprint - final, definitive version available at http://www.ieeexplore.com/ accepted for ICRA2016 May 2016

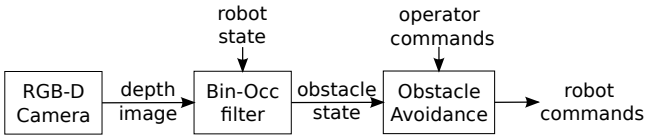

Fig. 3. Block diagram of the navigation system.

A. The Bin-Occupancy Filter

The Bin-Occupancy Filter [15] is as an algorithm capable of tracking multiple targets. It approaches the problem of multitarget tracking (MTT) by investigating if there is a target at a given point in space. It uses a model of the surveillance region which employs small "bins", which a target may or may not occupy. Hence, its working principle is opposite to most MTT algorithms, which instead estimate the number of targets and simultaneously track them. In our case, we are interested in detecting and tracking obstacles that might collide with the platform, without the actual need to know their number. Therefore, the use of the first approach is more reasonable.

The bins in the surveillance region S are assumed to be sufficiently small such that each of them is potentially occupied by at most one target. If two targets (regardless of their dimension) happen to be located close together, the bin volume can be reduced so that this assumption remains valid. Moreover, one target should give raise to only one measurement, and should generate it independently of the other targets.

The algorithm has the structure of a recursive filter whose state is the probability of occupancy of each bin in the region of interest S. Hence, it is composed of two phases:

1) State Update: At a given time k the event *bin i contains a target* is possible if:

a) a new target appears in bin i,
b) a target from bin j moves to bin i,

where i and j identify two generic bins. The second case includes also the case $j = i$, i.e., the target in bin i stays in this bin. These events, considering the assumptions above, are mutually exclusive - two targets cannot occupy the same bin.

The state update equation that computes the probability $p_{k|k}(U_k(i))$ of bin i being occupied at time k, given a set of measurements from time 1 up to $k-1$ is:

$$p_{k|k-1}(U_k(i)) = b(i) + \sum_j p(i|x_j) P_s(x_j) p_{k-1|k-1}(U_{k-1}(j)) \,, \quad (3)$$

where $U_k(i) \in \{0, 1\}$ is a random variable which is 1 if bin i contains a target at time k and 0 otherwise, $b(i)$ is the probability that a new target appears in bin i, x_j is the middle point of bin j, $P_s(x_j)$ is the probability of survival of a target located at x_j to the next time step, $p(i|x_j)$ is the probability that a target located at x_j moves to bin i, and $p_{k-1|k-1}(U_{k-1}(j))$ is the probability of bin i being occupied at time $k-1$ given a set of measurements from time 1 to $k-1$. Equation (3) describes the sum of probabilities of the two independent events a) and b).

2) Measurement Update: Assuming that at time k, the sensor detects m targets in z_s, $s \in \{1, \ldots, m\}$. The bin-occupancy probability of bin i is then updated using the general update equation:

$$p_{k|k}(U_k(i)) = p_{k|k-1}(U_k(i)) \left[(1 - P_d(x_i)) + \sum_{s=1}^{m} \frac{P_d(x_i) f(z_s|x_i)}{\sum_j P_d(x_j) f(z_s|x_j) p_{k|k-1}(U_k(j))} \right] , \quad (4)$$

which describes the probability that at time k, given the current measurement, bin i is occupied. $P_d(x_i)$ describes the probability of detection of a target located at x_i and $f(z_s|x_i)$ is a probability density function (pdf) of a single measurement z_s given a target at x_i.

The second part of (4) corresponds to the influence of measurement $s \in \{1, \ldots, m\}$ on the updated bin i, weighted by the pdf function $f(z_s|x_i)$, which depends on the measurement location. If there is no measurement, the second part vanishes.

B. Robo-Centric Obstacle State

In our case, we are interested in tracking the obstacles in a near vicinity of the UAV. Hence, we define a quantized region of interest S in a cylindrical coordinate frame $M : \{O_M, P_M, \Psi_M, Z_M\}$ around the robot, wherein the obstacles will be tracked, and (ρ, ψ, z) a generic point expressed in M. The P_M, Ψ_M coordinates are, respectively: the radial and the azimuth distances. The frame is defined such that $O_M \equiv O_H$ and $Z_M \equiv Z_H$. The azimuth Ψ_M reference plane is the surface at 45° to the $X_H Z_H$ plane (i.e., when the roll and pitch angles are equal to 0 it coincides with the $X_C Z_C$ camera plane). The surveillance region boundaries are defined as $(\rho_{S_{min}}, \rho_{S_{max}})$ and $(z_{S_{min}}, z_{S_{max}})$ on the P_M and Z_M axes, respectively.

The surveillance region is partitioned into cells, whose size in M is $\Delta\rho \times \Delta\psi \times \Delta z$. In the left part of Fig. 4 one "slice" of S and its partition into bins are shown. The inner circle represents the restricted area A, which is considered occupied by the quadrotor, hence no obstacle should get inside A. In the right part of Fig. 4, we show a 3D view of S and A. The black cross in the middle represents the quadcopter and its size with respect to S.

The dimension of cells $\Delta\rho, \Delta\psi, \Delta z$ is constant in the cylindrical coordinates, but considering this coordinate system properties, the absolute volume of cells differ. The standard Bin-Occupancy filter (subsection III-A) assumes the same volume for all the bins, thus one could argue if our approach fulfills this assumptions. Measurements from the RGB-D camera consist of depth frames, in which each pixel describes the distance to the area that it covers. Considering the model of an ideal pinhole camera, this area changes proportionally to the distance from the sensor. The volume of the cells changes on azimuth corresponding to this relation, thus ensuring the same probability of detection when we take into account the simultaneous change in volume of the cells and the pixels relative size.

Preprint - final, definitive version available at http://www.ieeexplore.com/ accepted for ICRA2016 May 2016

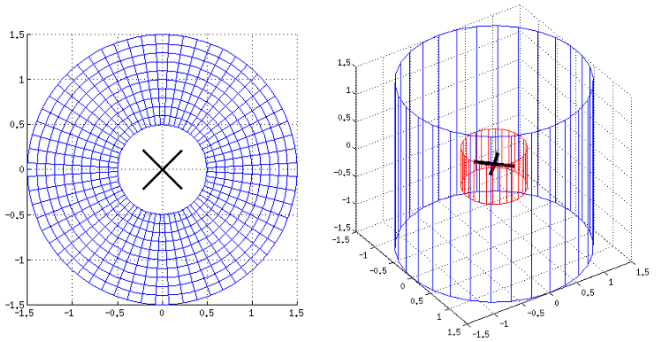

Fig. 4. Left: top view of the surveillance region divided into partitions. Right: in blue a 3D view of the surveillance region; in red the restricted area.

1) State Update: The prediction stage of our algorithm is performed based on the estimated state of the robot. The obstacles in S are updated using three independent transformations:

- a translation along the Z_M axis, given the UAV velocity v_z,
- a rotation around the Z_M axis, given the UAV yaw rate $\dot{\psi}$,
- a translation on the $P_M\Psi_M$ plane, given the UAV velocities v_x and v_y.

In order to keep the computational time bounded and limit the spread of the probabilities in S due to the quantization noise, these three transformations are not performed every time that a velocity measurement is available. Instead, the velocities are integrated and a transformation is performed only when the corresponding cumulated value reaches the corresponding cell size.

Similarly, obstacles that leave the surveillance region are erased from the memory. This also keeps the necessary memory bounded and independent from the duration of experiment without limiting the area of exploration. Note that to compute equation (3) the probability of survival $P_s(i)$ and the probability of birth of new targets $b(i)$ are needed. These parameters depend on the environment and should be set such that a boundary zones should have relatively high $b(i)$, while a value of $P_s(i) < 1$ can be used to reduce artifacts.

2) Measurement Update: Measurements from the camera (i.e. depth frames) must be preprocessed in order to be expressed in the same system as the obstacles state. At first the depth image is scaled down to the resolution of 30x40. The scaling process significantly reduces the computation time, but it does not limit the ability to detect obstacles (at a distance of 1.5 m from the sensor, each pixel cover an area of 4×4 cm). The scaled depth image is then transformed into a point-cloud using the camera intrinsic parameters. Next, the points are expressed in the frames Q and H using the camera extrinsic parameters (i.e. ${}^Q\mathbf{p}_C$ and $\mathbf{R}_C^Q$) and the roll and pitch angles. Finally, they are expressed in the cylindrical frame M. A scheme of the whole process is presented in Fig. 5.

The set of measurements $\{z_s, s \in 1, \ldots, m\}$ is computed as a list of bins that have at least one depth point inside

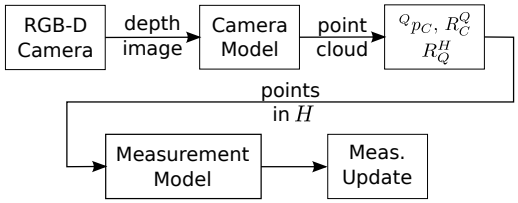

Fig. 5. Diagram of the measurement update of the Bin-Occupancy filter.

of them. The obstacles state is updated using (4), where the probability of detection $P_d(x_i)$ is defined as:

$$P_d(x_i) = V_i \left(\prod_j \left(1 - p_{k|k-1}(U_k(i)) \right) \cdot O_i(j) \right) , \quad (5)$$

where V_i is the visibility factor (1 for cells inside the camera FOV, 0 otherwise) and $O_i(j)$ is the occlusion rate of bin i by bin j, which depends on the relative location of bin j with respect to bin i and the camera center. Note that all $O_i(j)$ can be precomputed in order to save the computational time.

IV. OBSTACLE AVOIDANCE

The obstacle avoidance module is the second part of our algorithm. At every time step t the algorithm tries to predict possible collisions by computing the future obstacles state by performing a sequence of state updates (III-B.1) in different directions in the range of possible motion. Its working principle is based on Model Predictive Control (MPC) [10].

A. Model Predictive Control

In MPC, a model of the system is used to calculate the control input that minimizes an objective function. The system state is predicted at a finite-time horizon. To determine the control input, an Optimal Control Problem (OCP) is solved sequentially at each instance of the system. Then, the first control input is applied to the system, while keeping the future steps into account.

An example of such objective function is:

$$J(x, u) = p(x_t) + \sum_{t=0}^{N-1} q(x_t, u_t) , \quad (6)$$

where the time horizon is expressed as N computation steps in which the cost, expressed as $q(x_t, u_t)$ and $p(x_t)$, is computed. The optimization problem consist of finding the value of u_t that minimizes the cost function (6).

B. Commanded Velocity

The quadcopter is commanded in the horizontal frame H, i.e. the desired velocity is given in directions parallel and perpendicular to the gravity vector. Since the platform is intended for teleoperation, the range of possible commanded velocities is limited. In particular, we have allowed the following movements:

- translation along the axis defined by a unit vector $\mathbf{i} = [cos({}^Q\psi_C), sin({}^Q\psi_C), 0]^T$ (forward/backward[1]), to

[1]the backward motion, since there is no visual feedback, is limited by the dimension of S.

allow motion in the direction of the camera, where ${}^{Q}\psi_C$ is the yaw angle of C in Q,
- along the Z_H axis (up-/downward),
- rotation around the Z_H axis (left/right).

The commanded velocity ${}^{H}\mathbf{v}_D$ is:

$$ {}^{H}\mathbf{v}_D = \begin{bmatrix} v_{Dx} \\ v_{Dy} \\ v_{Dz} \end{bmatrix} = \begin{bmatrix} cos({}^{Q}\psi_C)v_D \\ sin({}^{Q}\psi_C)v_D \\ v_{Dz} \end{bmatrix} \tag{7} $$

and ${}^{H}\dot{\psi}_D$ is the commanded yaw rate, where v_D, v_{Dz} and ${}^{H}\dot{\psi}_D$ are three inputs provided by the operator.

C. Obstacle Avoidance

In order to avoid obstacles, we first define the probability of collision with an obstacle in bin i as:

$$ w_i \cdot p(U_k(i)) \,, \tag{8} $$

which is the product of the probability of bin i being occupied $p(U_k(i))$ and weight $w_i \in [0,1]$ which expresses the probability of having a collision with bin i when $U_k(i) = 1$. In general, $w_i \neq 0$ if and only if $x_i \in A$, and it depends on the location of bin i and on the shape of the quadrotor.

Assuming that collisions with different bins are independent, the probability of collision with any obstacle in S at step k can be expressed as:

$$ 1 - \prod_{i \in S} \Big(1 - w_i \cdot p(U_k(i)) \Big) \,. \tag{9} $$

To achieve the collision-free teleoperation, our algorithm solves an optimization problem over N timesteps:

$$ \begin{aligned} \underset{\mathbf{u}_t}{\operatorname{argmin}} J = & \sum_{t=1}^{N} w_t \left(1 - \prod_{i \in S} \Big(1 - w_i \cdot p(U_{k+t}(i) | \mathbf{u}_t) \Big) \right) \\ & + w_\psi \cdot |u_\psi| \\ & + w_z \cdot |u_z| \,, \end{aligned} \tag{10} $$

and produces a set of N optimal control inputs $\mathbf{u}_t^\star = (u_\rho, u_\psi, u_z)^T$ and gives the number $N_s \leq N$ of collision-free steps. The parameters w_ψ and w_z are weights that we explain later in this Section. The term $p(U_{k+t}(i)|\mathbf{u}_t)$ expresses the probability of bin i being occupied in step t. It is determined using the model of the system (i.e.: the obstacles state and the transformations defined in III-B.1) and the input $\mathbf{u}_t$. The probability of collision in each step t is weighted by w_t.

The control input $\mathbf{u}_t$, expressed in the cylindrical coordinates M, is a single transformation of the obstacles state, as defined in (III-B.1), and must be in the form:

$$ \mathbf{u}_t = \begin{bmatrix} u_\rho \\ u_\psi \\ u_z \end{bmatrix} = \begin{bmatrix} \Delta\rho \\ n \cdot \Delta\psi \\ m \cdot \Delta z \end{bmatrix}, \begin{matrix} n \in \{\mathbb{Z} : n\Delta\psi \in [-\frac{\pi}{2}, \frac{\pi}{2}]\} \\ m \in [-3, 3]. \end{matrix} \tag{11} $$

The above relation also constrains our optimization problem (10) by defining the feasible region of $\mathbf{u}_t$. In fact, it allows, in each prediction step t, to make a translation of the size of a cell $\Delta\rho$ in the azimuth direction of $n\Delta\psi$ and a potential change of elevation along the Z_M axis.

The number of prediction steps N is not predefined, but is decided based on the forward velocity commanded by the operator v_D. In particular, it corresponds to the number of control steps $\mathbf{u}_t$ needed to achieve the desired translation of $v_D \cdot T_h$, where T_h is the time horizon of the prediction. The value of N can be computed as:

$$ N = \left\lceil \frac{v_D \cdot T_h}{\Delta\rho} \right\rceil . \tag{12} $$

Since on the $X_H Y_H$ plane we only allow a forward/backward commanded velocity (IV-B), in (10), with w_ψ and w_z, we penalize the change of direction and elevation by adding weighted costs.

Lastly, the reference velocity in the frame H sent to the robot is

$$ {}^{H}\mathbf{v}_{ref} = \begin{cases} \frac{N_s}{N}\mathbf{R}_z\left(\frac{\pi}{4}\right) \begin{bmatrix} v_D \cdot \cos(n\Delta\psi) \\ v_D \cdot \sin(n\Delta\psi) \\ 0 \end{bmatrix} & \text{if } m = 0 \\ \frac{N_s}{N}\mathbf{R}_z\left(\frac{\pi}{4}\right) \begin{bmatrix} 0 \\ 0 \\ m \cdot \Delta z / T_h \end{bmatrix} & \text{if } m \neq 0. \end{cases} \tag{13} $$

Note that the coefficient N_s/N is meant to limit the velocity when there is no collision-free path over the whole time horizon T_h. In the extreme case when $N_s = 0$, the reference velocity is set to zero. This case correspond to the situation in which obstacles are in front of the robot in all feasible directions.

To summarize, our obstacle avoidance algorithm produces a reference velocity (13) as a set of translations of the obstacles state by minimizing the difference between the commanded velocity (7) and the output, reference signal. There are three possible cases:

- $\mathbf{v}_{ref} = \mathbf{v}_D$ when there is no predicted collision on the desired direction,
- lateral or vertical avoidance by projecting the desired velocity $\mathbf{v}_D$ on a new, collision-free direction,
- limit of the commanded velocity when there is no fully collision-free direction.

Remark 1: Although we do not allow the operator to command any lateral motion, the algorithm, knowing the obstacles state, can perform such a motion.

Remark 2: The algorithm cannot perform any motion by itself, any obstacle avoidance action is only possible under a presence of the operator input.

Remark 3: In order to compute (10), it is not needed to perform the prediction of the obstacles state in the whole region S. In fact, to reduce the computational time, it is possible to compute only the future state of the bins that have $w_i \neq 0$.

V. PLATFORM DESCRIPTION

The UAV used in the development of the presented navigation system is a MK-Quadro quadcopter from MikroKopter. It consists of a frame with four 10 inch propellers powered by brushless motors, motor controllers and a flight controller

board with an 8-bit microcontroller. The main board includes an inertial measurement unit (IMU), i.e., two 3-axis, 10-bit analog sensors: an accelerometer (with a range of ± 2 g) and a gyroscope (± 300 deg/s range), both read with an analog to digital converter. The board communicates with the brushless motor controllers through a standard I^2C bus and offers two serial connections with a 115 200 Bd baud rate.

The original firmware, which allows us to drive the quadcopter with a remote control, has been replaced with our own software that has new features and a different interface that allows us to control the robot through serial connection. It is used to send attitude (roll and pitch angles, and yaw rate) and thrust commands to the microcontroller at ~100 Hz, and to receive low-frequency data (~20 Hz) (i.e. battery level). The platform is powered by a 2600 mAh LiPo battery that provides approximately 10 min of flight.

In addition, we have equipped the system with ODROID-XU3, a double quad core[2] ARM microprocessor board that provides enough computational power to make the system independent of external computational units. Communication with the low-level flight controller is carried out by two XU3's serial ports. Power to the board and its peripherals is provided by a 5 V step-down voltage regulator connected to the LiPo battery. The board can exchange data with the fixed operator desk using a USB Wi-Fi adapter.

An on-board RGB-D sensor (Asus Xtion Pro) is used as the source of data for obstacle detection providing 640×480 RGB and depth images at ~30fps. It is mounted approximately 45° to the right with respect to the front propeller and rotated 90° around the camera axis to extend the vertical field of view (FOV). It is also rotated downward at about 20° to increase the number of visual features inside the FOV by framing a bigger portion of the ground, while simultaneously offering a horizontal line of sight to the operator. The Odroid board and the camera are attached rigidly to the frame with 3D printed parts. The complete platform depicted in Fig. 1 weights approximately 1.3 kg.

The velocity of the UAV is provided by an external tracking system. Hence, we are able to evaluate the performance of the tracking and obstacle avoidance algorithms independently from state estimation errors.

VI. EXPERIMENT

In order to validate the presented algorithm we performed two experiments: avoidance of a horizontal and a vertical, rectangular obstacles. In both experiments we commanded the robot to fly towards the outer edge of the obstacle, simulating a situation in which an operator, relying only on the visual and haptic feedbacks, may not be certain if the robot can pass safely near the obstacle. In such cases, the avoidance algorithm should alter the commanded velocity.

The surveillance region S is parametrized as:

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

[2]four Cortex-A15 at 2.0 Ghz, four Cortex-A7 and Heterogeneous Multi-Processing (HMP) solution for tasks management

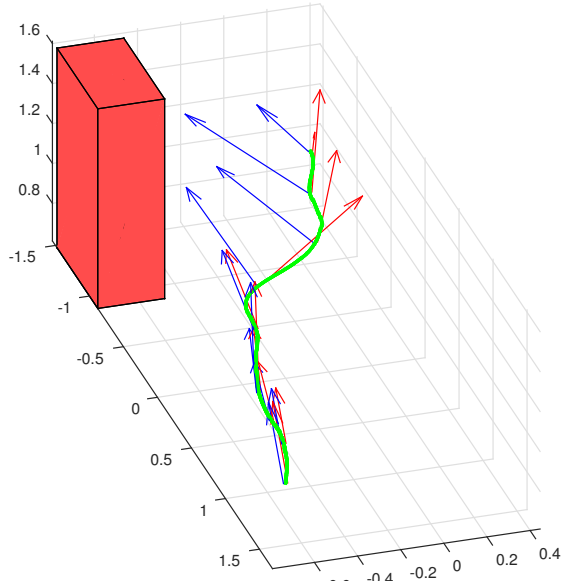

Fig. 6. Avoidance of a vertical obstacle. In green the trajectory of the robot. In blue the velocity commanded by the operator. In red the reference velocity from the obstacle avoidance module.

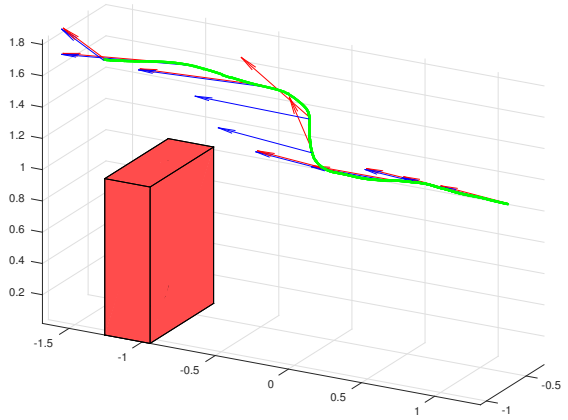

Fig. 7. Avoidance of a horizontal obstacle. In green the trajectory of the robot. In blue the velocity commanded by the operator. In red the reference velocity from the obstacle avoidance module.

The restricted area A occupied by the robot (i.e.: the set of bins that the algorithm keeps obstacle-free) is defined as:

$$\rho \in [0, 0.5\,m], \quad z \in [-0.4\,m, 0.3\,m]\,.$$

The weights used in the cost calculation (10) are:

$$w_i = \begin{cases} 1 & \text{if } x(i) \in A \\ 0 & \text{if } x(i) \notin A, \end{cases} \qquad \begin{aligned} w_t &= 10 \\ w_\psi &= 3 \\ w_z &= 2\,. \end{aligned}$$

These values have been tuned according to heuristic criteria through experiments. Finding optimal weights is not trivial as they depend on the desired behavior of the platform in presence of obstacles which depends on the task and the operator himself.

The operator input, desired velocity (7), is limited with as:

$$v_D, v_{Dz} \in \left[-0.5\,\frac{m}{s}, 0.5\,\frac{m}{s}\right], \quad {}^H\dot{\psi}_D \in \left[-1\,\frac{rad}{s}, 1\,\frac{rad}{s}\right].$$

while the time horizon is assumed to be $T_h = 1s$.

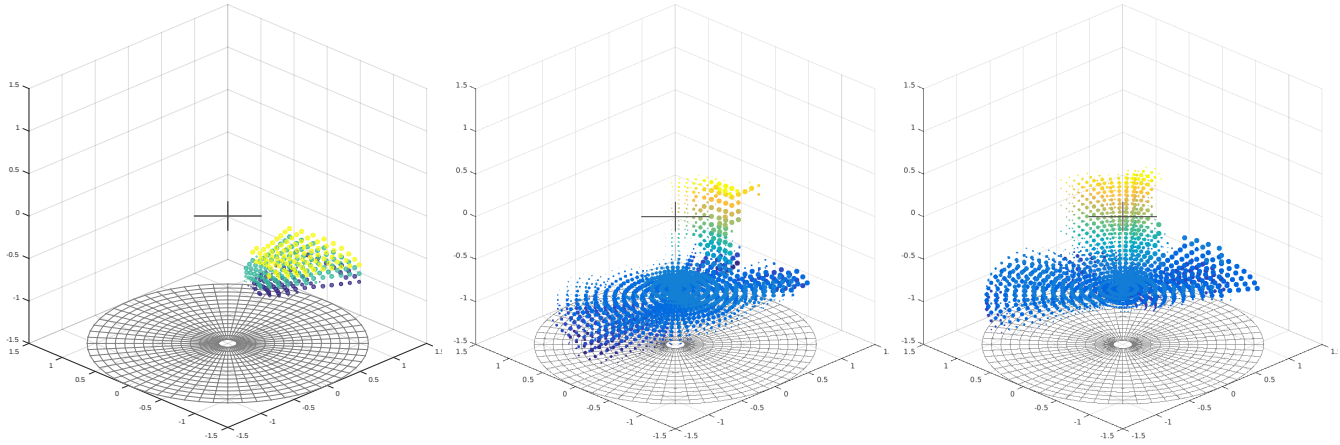

Fig. 8. Three frames of the estimated obstacles state in the horizontal avoidance experiment.

We present the results of the two experiments in Fig. 6 and Fig. 7. In green, we show the trajectory of the robot. The blue arrows are the velocities commanded by the user ${}^{H}\mathbf{v}_D$ and in red, the reference velocities ${}^{H}\mathbf{v}_{ref}$ altered by the obstacle avoidance module. From the plots, it is clear how a horizontal or a vertical component are added to the motion in order to prevent collisions. The interested reader is invited to watch the accompanying video for a clip of these experiments.

In Fig. 8, we show three frames of the estimated obstacles states in the horizontal avoidance experiment. Each bin in S is represented as a sphere whose radius is proportional to the probability of occupancy. The color of the sphere depends on the z_M coordinate. The first frame correspond to the robot during takeoff, hence the only visible obstacles are part of the ground. In the second frame, the robot is approaching the obstacle, and in the last frame, after avoidance, the obstacle is on the left of the robot. In general, the frames show that the Bin-Occupancy filter is able to correctly estimate the obstacles state.

VII. CONCLUSION

In this work, we have developed a method for collision avoidance for UAVs 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 obstacles are used in an algorithm inspired by Model Predictive Control in order to modify the velocity commanded by the operator and avoid obstacles. We have successfully performed several experiments to validate our approach, performing all the computations on-board.

In future, we plan to integrate on our platform a visual odometry algorithm to estimate on-board the velocity of the robot without the need of external sensors.

REFERENCES

[1] T. M. Lam, H. W. Boschloo, M. Mulder, and M. M. van Paassen, "Artificial force field for haptic feedback in UAV teleoperation," IEEE Trans. on Systems, Man, & Cybernetics. Part A: Systems & Humans, vol. 39, no. 6, pp. 1316–1330, 2009.

[2] M.-D. Hua and H. Rifai, "Obstacle Avoidance for Teleoperated Underactuated Aerial Vehicles using Telemetric Measurement," IEEE Conf. on Control and Decision, 2010, pp. 262–267.

[3] S. Omari, M.-D. Hua, G. Ducard, and T. Hamel, "Bilateral Haptic Teleoperation of an Industrial Multirotor UAV," in *Gearing up and accelerating cross-fertilization between academic and industrial robotics research in Europe*, Springer International Publishing, 2014, pp. 301-320.

[4] R. Mahony, F. Schill, P. Corke, and Y. S. Oh, "A new framework for Force Feedback Teleoperation of Robotic vehicles based on Optical Flow," IEEE Conference on Robotics and Automation, 2009, pp. 1079–1085.

[5] P. Stegagno, M. Basile, H. H. Büthoff, and A. Franchi, "A Semi-autonomous UAV Platform for Indoor Remote Operation with Visual and Haptic Feedback," IEEE International Conference on Robotics and Automation (ICRA 2014), IEEE, Piscataway, NJ, USA, 3862-3869.

[6] D. Ferguson, M. Likhachev, and A. Stentz, "A Guide to Heuristic-based Path Planning," Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS), June, 2005.

[7] M. Becker, C. M. Dantas, and W. P. Macedo, "Obstacle Avoidance Procedures for Mobile Robots," ABCM Symposium Series in Mechatronics, vol. 2, pp. 250-257, 2006.

[8] M. Zohaib, M. Pasha, R. A. Riaz, N.Javaid, M. Ilahi, and R. D. Khan, "Control Strategies for Mobile Robot With Obstacle Avoidance," arXiv:1306.1144.

[9] J. Serres, F. Ruffier, and N. Franceschini, "Two optic flow regulators for speed control and obstacle avoidance," International Conference on Biomedical Robotics and Biomechatronics, 2006, pp. 750–757.

[10] E. F. Camacho, C. Bordons, "Introduction to Model Predictive Control," in *Model Predictive Control*, London: Springer-Verlag, 2007, pp. 1-11.

[11] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," IEEE Trans. on Automatic Control, vol. 53, no. 5, pp. 1203–1218, 2008.

[12] P. Martin and E. Salaün, "The true role of accelerometer feedback in quadrotor control," IEEE Int. Conf. on Robotics and Automation, Anchorage, AK, May 2010, pp. 1623–1629.

[13] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," IEEE Int. Conf. on Robotics and Automation, Karlsruhe, Germany, May 2013, pp. 3748–3754.

[14] Z. Fang and S. Scherer, "Real-time Onboard 6DoF Localization of an Indoor MAV in Degraded Visual Environments Using a RGB-D Camera," IEEE International Conference on Robotics and Automation, May 2015.

[15] O. Erdinc, P. Willett, Y. Bar-Shalom, "The Bin-Occupancy Filter and Its Connection to the PHD Filters," IEEE Trans. on Signal Proc., SP-57, November 2009, 4232-4246.

[16] D. J. Lee, A. Franchi, H. I. Son, H. H. Bülthoff, and P. Robuffo Giordano, "Semi-autonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles," IEEE/ASME Trans. on Mechatronics, Focused Section on Aerospace Mechatronics, 2013, vol. 18, no. 4, pp. 1334–1345.