Reciprocal Collision Avoidance For Quadrotors Using On-board Visual Detection

Steven Roelofsen[1,2], Denis Gillet[2] and Alcherio Martinoli[1]

***Abstract*— In this paper we present a collision avoidance system based on visual detection. Our hardware consists of a Hummingbird quadrotor equipped with a large red marker with two built-in fish-eye cameras. Fusion of the measurements from the two cameras is done using a Gaussian-mixture probability hypothesis density filter, which allows for tracking several aircrafts at the same time. Our collision avoidance algorithm is based on navigation functions designed to cope with cameras characterized by limited field of view. Its mathematical correctness has been proven in a former paper [1]. The collision avoidance maneuver is performed without the vehicles explicitly exchanging information via communication but instead relying solely on on-board sensors. Our system has been validated in an indoor space with four different collision scenarios. Trajectory data was recorded with an external motion capture system and demonstrate good robustness against sensing noise.**

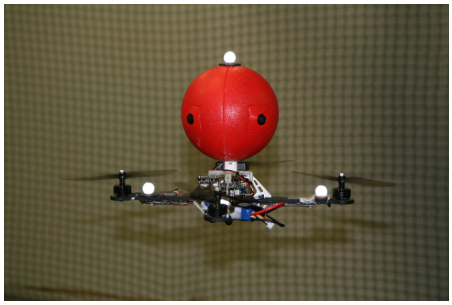

Fig. 1. Picture of one of the AscTec quadrotor equipped with the marker. The cameras (two large black dots) and the computational system are embedded into the marker. The small white spheres are markers for the 3D motion capture system.

I. Introduction

Unmanned Aerial Vehicles (UAVs), and quadrotors in particular, are becoming increasingly popular in academia, industry and with individual aircraft hobbyists. Available in various shapes and sizes, UAVs are used is numerous applications including environmental monitoring, military operations as well as entertainment and various indoor demonstrations. However, even in the state-of-the-art applications the UAVs are typically controlled under human supervision. Recently, new application opportunities for UAVs in the area of surveillance and package delivery have been attracted the attention of both research and industrial communities. Such applications challenges the former remote control paradigm and push for additional autonomy of the vehicles, as it would be neither competitive nor financially viable to have one operator for each vehicle.

Fully autonomous aircraft are a challenge because of the hazards that they may encounter and represent, especially in: bad weather conditions and densely cluttered environments populated by static (e.g., buildings) and mobile (e.g., other vehicles, human beings, birds) obstacles. Achieving such dependable flying autonomy is especially difficult for small UAVs weighing no more than a few kilograms because of the limited payload for sensing and computation.

One of the key challenges associated with UAVs is collision avoidance with other aircraft. Both vehicles may or may not collaborate during the collision avoidance maneuver, the second being true, for instance, for quadrotor toys encountering birds. Thus the UAV has to rely on detection and tracking of an obstacle with on-board resources to move away from the threat. A natural solution is to mimic the ability of human pilots and use vision as main sensing modality [2]. In [3] and [4] a camera is used for collision avoidance. The position of encounter on the image is directly used to derive a control law for the collision avoidance. While the proposed algorithm has been designed to take into account the inherent characteristics of an on-board camera, it is unclear whether it can cope with multiple targets. The experiments have been done with only a single quadrotor avoiding static obstacles monitored by a Motion Capture System (MCS) and the visual processing carried out off-board. In a successive contribution, the authors improved their methods by optimizing the trajectories using a model predictive control method [5]. Another solution with a vision system has been proposed in [6], where the on-board camera of an AR-Drone was used to estimate the other quadrotor's position using a specific marker. As avoidance algorithm the authors used Optimal Reciprocal Collision Avoidance (ORCA). The quadrotors were remotely piloted by humans who attempted to make the quadrotors collide. The ORCA algorithm was responsible to avoid the collision and generated smooth and optimal trajectories in terms of discrepancy with the desired vehicles' velocities. However, no avoidance guarantee was given. Oscillation were noticed by the authors probably due to errors in the velocity measurement. Our present work aims to address several of the points above, namely designing a system capable to track and avoid multiple aircraft with guarantees under unreliable velocity estimates.

Some background on collision avoidance algorithms and

This work has been financially supported by Honeywell, and has benefitted of the administrative and technical coordination of the EPFL Transportation Center.

[1] S. Roelofsen and A. Martinoli are with the Distributed Intelligent Systems and Algorithms Laboratory, School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne.

[2] S. Roelofsen and D. Gillet are with the the Coordination and Interaction System Group, School of Engineering, École Polytechnique Fédérale de Lausanne.

tracking algorithms is given in Section II. The proposed solution is presented in Section III. Experiment results are presented in Section IV. A conclusion is given in Section V.

II. Background

A. Tracking

The tracker needs not only to estimate the trajectory of an encountered aircraft but also to give the correct number of aircraft present in the surrounding area. Several multi-target tracking algorithms have been designed over the years and can be separated in three major categories: Joint Probabilistic Data Association (JPDA) [7], Multiple Hypothesis Tracking (MHT) [8] and Random Finite Set (RFS) trackers [9].

RFS theory is a source of new and effective tracking techniques with a wide range of possible applications. A brief introduction will now be presented. For more in depth theoretical explanations, please refer to [9], [10].

RFS extends the notion of random variables to sets. Those sets come from considering that at time k there are $M(k)$ targets present in the scene. Their states $\mathbf{x}_{k,1}, ..., \mathbf{x}_{k,M(k)} \in \mathcal{X}$ become $\mathbf{x}_{k+1,1}, ..., \mathbf{x}_{k+1,M(k+1)}$ at time $k+1$. $\mathcal{X}$ is the state space. At time k the sensor generates $N(k)$ measurements $\mathbf{z}_{k,1}, ..., \mathbf{z}_{k,N(k)} \in \mathcal{Z}$ with $\mathcal{Z}$ the measurement space. The number of targets and measurements may not be the same due to misdetections and clutter. The random sets are defined as:

$$X_k = \left\{\mathbf{x}_{k,1}, ..., \mathbf{x}_{k,M(k)}\right\} \in \mathcal{F}(\mathcal{X}) \quad (1)$$

$$Z_k = \left\{\mathbf{z}_{k,1}, ..., \mathbf{z}_{k,N(k)}\right\} \in \mathcal{F}(\mathcal{Z}) \quad (2)$$

with $\mathcal{F}(\mathcal{X})$ and $\mathcal{F}(\mathcal{Z})$ being the respective collections of all finite subsets of $\mathcal{X}$ and $\mathcal{Z}$. One can write the targets' time evolution using RFS. If X_{k-1} is the multi-target state at time $k-1$, each $\mathbf{x}_{k-1} \in X_{k-1}$ either continues to exist at time k with probability $p_{S,k}(\mathbf{x}_{k-1})$ or disappears with probability $1 - p_{S,k}(\mathbf{x}_{k-1})$. If the target survives, it will transition from state $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$ with probability density $f_{k|k-1}(\mathbf{x}_k|\mathbf{x}_{k-1})$. Using those definitions, one can build the set $S_{k|k-1}(\mathbf{x}_{k-1})$ that returns $\{\mathbf{x}_k\}$ with probability $p_{S,k}(\mathbf{x}_{k-1})$ and $\varnothing$ with probability $1 - p_{S,k}(\mathbf{x}_{k-1})$. Targets can also appear between time $k-1$ and k, which is described with the RFS Γ_k. In this paper we ignore targets birth from other targets. The time evolution of X_k is given by:

$$X_{k+1} = \left[\bigcup_{\zeta \in X_{k-1}} S_{k|k-1}(\zeta)\right] \cup \Gamma_k \quad (3)$$

In a similar way a RFS model can be written for the measurement set. At time k, a target $\mathbf{x}_k \in X_k$ can either generate a measurement $\mathbf{z}_k$ with probability $p_{D,k}(\mathbf{x}_k)$ or generate none with probability $1 - p_{D,k}(\mathbf{x}_k)$. Each target $\mathbf{x}_k$ generates a RFS $\Theta_k(\mathbf{x}_k)$ that either returns a measurement $\{\mathbf{z}_k\}$ according to $g_k(\mathbf{z}_k|\mathbf{x}_k)$ if it is detected or $\varnothing$ if not. The sensor might also be corrupted with clutter, which is described by the RFS K_k. The RFS Z_k given X_k is given by:

$$Z_k = \left[\bigcup_{\mathbf{x} \in X_k} \Theta_k(\mathbf{x})\right] \cup K_k \quad (4)$$

Similar to the single target case, the multi-target transition and observation randomness can be captured with a multi-target transition density $f_{k|k-1}(X_k|X_{k-1})$ and a multi-target likelihood $g_k(Z_k|X_k)$.

If $p_k(\cdot|Z_{1:k})$ denotes the multi-target posterior density then the optimal multi-target Bayes filter is given by:

$$p_{k|k-1}(X_k|Z_{1:k-1}) = \int f_{k|k-1}(X_k|X) p_{k-1}(X|Z_{1:k-1}) \mu_s(dX), \quad (5)$$

$$p_k(X_k|Z_{1:k}) = \frac{g_k(Z_k|X_k) p_{k|k-1}(X_k|Z_{1:k-1})}{\int g_k(Z_k|X) p_{k|k-1}(X|Z_{1:k-1}) \mu_s(dX)} \quad (6)$$

Computing Equations 5 and 6 requires the evaluation of several integrals on $\mathcal{F}(\mathcal{X})$, which is intractable.

To have a tractable algorithm, it is possible to propagate a first order statistical moment instead of the full multi-target posterior density. Such filter is called a Probability Hypothesis Density (PHD) filter. The first order moment of a RFS X with probability distribution P is a non-negative function v on $\mathcal{X}$ called intensity such that for each region $S \subseteq \mathcal{X}$,

$$\int |X \cap S| P(dX) = \int_S v(\mathbf{x}) d\mathbf{x} \quad (7)$$

Equations 5 and 6 can be approximated using an intensity measure, leading to the following equations:

$$v_{k|k-1}(\mathbf{x}) = \int p_{S,k}(\zeta) f_{k|k-1}(\mathbf{x}|\zeta) v_{k-1}(\zeta) d\zeta + \gamma_k(\mathbf{x}), \quad (8)$$

$$v_k(\mathbf{x}) = [1 - p_{D,k}(\mathbf{x})] v_{k|k-1}(\mathbf{x}) + \sum_{\mathbf{z} \in Z_k} \frac{p_{D,k}(\mathbf{x}) g_k(\mathbf{z}|\mathbf{x}) v_{k|k-1}(\mathbf{x})}{\kappa(\mathbf{z}) + \int p_{D,k}(\zeta) g_k(\mathbf{z}|\zeta) v_{k|k-1}(\zeta) d\zeta} \quad (9)$$

with $\gamma_k(\mathbf{x})$ the intensity of targets appearing and $\kappa(\mathbf{z})$ the intensity of clutter in the measurements. Even if this formulation reduces the computational cost, Equations 8 and 9 do not admit a general closed form solution. By approximating the intensity as the sum of Gaussians, the Gaussian Mixture Probability Hypothesis Density (GM-PHD) allows for a closed form solution in a similar form to the Kalman filter. The intensity of the GM-PHD filter is of the form:

$$v_k(\mathbf{x}) = \sum_{i=1}^{J_k} w_k^{(i)} \mathcal{N}(\mathbf{x}; m_k^{(i)}, P_k^{(i)}) \quad (10)$$

with J_k the number of Gaussians in the intensity at time k. $m_k^{(i)}$ is i gaussian's mean (in this case a vector composed of the target's position $\mathbf{q}_k^{(i)} = [x_k^{(i)}, y_k^{(i)}, z_k^{(i)}]^T$ and velocity $\mathbf{v}_k^{(i)}$) at time k and $P_k^{(i)}$ is its covariance. $w_k^{(i)}$ is the weight associated with the Gaussian.

B. Avoidance

Several collision avoidance algorithms can be classified in three different categories: direct control, Navigation Function (NF) based and Velocity Obstacle (VO) based. Direct control uses the sensor measurements as input of the controller in [4] where the position on the image is directly used to compute

the desired vertical velocity and yaw rate. Another category of algorithms are the ones based on NFs. NF algorithms use an analytic function of which the gradient is the direction to follow. The NF can be adapted to fixed-wing aircraft [11], [12] or quadrotors [13]. Sensing limitations can also be incorporated. Finally, there is also the VO-based algorithms, where the other aircraft are represented as forbidden velocities areas in the velocity space. Host aircraft velocity is then computed as being both allowed and closest to desired velocity. Such algorithms have successfully been applied to quadrotor with a MCS [14] and vision as sensing modality [6].

In this paper sensing is based on vision, which typically has a limited field of view. Two types of algorithms have been used with vision so far. The first is the control law from [3], which is not multitarget. Another proposed solution is VO based avoidance [6], for which they noticed oscillations due to noise in the sensing. In [1] we proposed a new collision avoidance method for fixed-wing aircraft based on NF. The algorithm uses the NF when no other aircraft is near and switches to a collision avoidance maneuver when an aircraft gets too close to the host vehicle.

III. Proposed solution

Our solution consists of equipping a quadrotor with two fisheye cameras. The images from the two cameras are processed using color segmentation and fused using a GM-PHD filter. The information about encountered quadrotors is fed to the collision avoidance algorithm that sends the commands further to the low-level control, as it is illustrated in Figure 2. Our work is based on the following assumptions:

- They have the same hardware and use the same detection, tracking, and avoidance algorithms.
- They carry the same marker.
- They fly at a similar height.
- They have at all times information about their own state, namely position, velocity and orientation.

Throughout this paper we will use the following notation. The quadrotor state is defined by its position $\mathbf{q} = [x, y, z]^T$, its velocity $\mathbf{v} = [\dot{x}, \dot{y}, \dot{z}]^T$ and its orientation with regard to the world frame in the form of a rotation matrix $R_W = R_\psi R_\phi R_\theta$ with ψ yaw, ϕ roll and θ pitch angles. The transformation from the world to the body frame is noted $R_B = R_W^T$. The quadrotor has two cameras, each one with a Field Of View (FOV) of α_d as shown in Figure 3. The two cameras give the quadrotor a combined FOV of $2\alpha_s$. Each quadrotor is circumscribed by a virtual safety cylinder of radius r that should never be entered. We consider a collision occurs when two quadrotors come closer than a distance $2r$.

A. Platform

To test our algorithms, we use a Hummingbird quadrotor from Ascending Technologies as baseline platform. The quadrotor is augmented with a ARM-based CPU from Gumstix and two fisheye (185° FOV) USB cameras at a resolution of 320 by 240 pixels, all encapsulated in a red marker with a diameter of 15 cm. Such marker allows

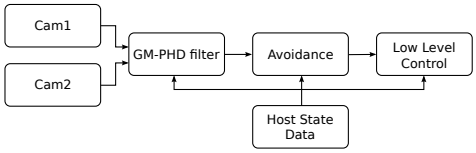

Fig. 2. Flowchart of the proposed solution. The two camera blocks capture and process the image, including the color segmentation. The measurements are fed into the GM-PHD filter which outputs target's locations. The avoidance block computes the desirable velocity and yaw rate to avoid the collision. The low-level block computes the motor speeds required to fulfil the commands of the avoidance algorithm. All blocks have access to host's state.

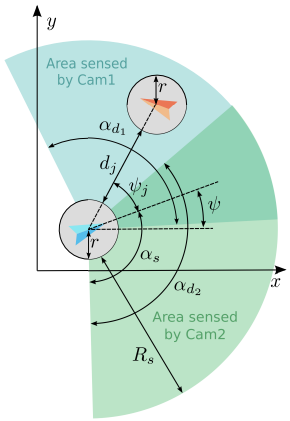

Fig. 3. Illustration of the important angles and distances appearing in this paper.

to be seen from all sides to the expense of accuracy and reliability compared to other marker types. The cameras are rotated toward each side with an angle of 45° so that their combined field of view is more than 220°. Figure 1 shows one quadrotor during flight. The software is written within the Robotic Operating System (ROS) framework. To interface our software with the quadrotor, the asctec_mav_interface [15] ROS package was used.

B. Computer Vision

There exist numerous computer vision algorithms to detect objects, but only a few of them have been adapted for detection of a generic aircraft and none with a low enough computational complexity to be run on-board of small quadrotors. In order to relax the computational requirements of a on-board solution based on computer vision and at the same time make the effort relevant for future enhanced embedded computational capabilities, we have decided to simplify the detection task by adding an easy-to-detect marker on the vehicle in the form of a red sphere (see Figure 1). While the detection task is now simplified, all the other difficulties are maintained: estimation of the position and size based on monocular vision, sensing affected by noise (clutter and misdetections). Color segmentation is used to detect the 15 cm in diameter red marker. The algorithm returns the position of the color blob's center as well as the number of pixels it covers. The color segmentation is performed on the raw image obtained by the camera. No distortion correction

or smoothing is performed in order to save CPU time. As all markers are identical, there is no possible identification of the quadrotors..

C. Tracking

The GM-PHD filter has four fundamental steps. First, a prediction step is done by applying the motion model on previous intensity. Then the intensity is updated with the newly acquired measurements. To keep the algorithm tractable, the Gaussians are merged and pruned in the third step. Finally the targets are inferred from the intensity map during the extraction step. Each step is performed at the time one of the cameras returns a measurement. We will now describe each stage individually.

1) Prediction: During the prediction step, the intensity is computed as:

$$v_{k|k-1}(\mathbf{x}) = \sum_{i=1}^{J_k} p_{S,k}(m_{k|k-1}^{(i)}) w_k^{(i)} \mathcal{N}(\mathbf{x}; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}) + \gamma_k(\mathbf{x}). \tag{11}$$

The appearance model is defined to cover the quadrotor's sensing region. Thus the appearance is defined in the quadrotor's body frame: $m_{\gamma,k}^{(i)} = [\mathbf{q}_{\gamma,k}^{(i)}, 0, 0, 0]^T$ with $\mathbf{q}_{\gamma,k}^{(i)} = R_b \mathbf{q}_{\gamma,b}^{(i)} + \mathbf{q}$. The predicted mean and covariance are defined as:

$$m_{k|k-1}^{(i)} = F_{k-1} m_{k-1}^{(j)} \tag{12}$$

$$P_{k|k-1}^{(i)} = Q_{k-1} + F_{k-1} P_{k-1}^{(i)} F_{k-1}^T \tag{13}$$

with F_{k-1} the transition model and Q_{k-1} motion noise model. Note that Equation 11 differs from [10] in that the survival probability $p_{S,k}$ is not constant but is a function of the gaussian mean. This is an approximation necessary to cope with the limited FOV of the sensors as the targets might accumulate in non-sensed areas if care is not taken. In this application $p_{S,k}$ is defined as:

$$p_{S,k}(m_{k|k-1}^{(i)}) = \begin{cases} 0.9 & \text{if } |\psi_j| < 110^\circ \\ 0.2 & \text{otherwise} \end{cases} \tag{14}$$

2) Update: The update step for a set of measurements Z_k is given by:

$$v_k(\mathbf{x}) = \sum_{i=1}^{J_k} (1 - p_{D,k}(m_{k|k-1}^{(i)})) w_{k|k-1}^{(i)} \mathcal{N}(\mathbf{x}; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}) + \sum_{\mathbf{z} \in Z_k} \sum_{i=1}^{J_k} w_k^{(i)}(\mathbf{z}) \mathcal{N}(\mathbf{x}; m_{k|k}^{(i)}(\mathbf{z}), P_{k|k}^{(i)}) \tag{15}$$

with

$$w_k^{(i)}(\mathbf{z}) = \frac{p_{D,k}(m_{k|k-1}^{(i)}) w_{k|k-1}^{(i)} l_k^{(i)}(\mathbf{z})}{\kappa(z) + \sum_{i=1}^{J_k} p_{D,k}(m_{k|k-1}^{(i)}) w_{k|k-1}^{(i)} l_k^{(i)}(\mathbf{z})} \tag{16}$$

$$l_k^{(i)}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; H_k^{(i)} m_{k|k-1}^{(i)}, R_k + H_k^{(i)} P_{k|k-1}^{(i)} [H_k^{(i)}]^T) \tag{17}$$

$$m_{k|k}^{(i)}(\mathbf{z}) = m_{k|k-1}^{(i)} + K_k^{(i)} (\mathbf{z} - H_k^{(i)} m_{k|k-1}^{(i)}) \tag{18}$$

$$P_{k|k}^{(i)} = \left[I - K_k^{(i)} H_k^{(i)} \right] P_{k|k-1}^{(i)} \tag{19}$$

$$K_k^{(i)} = P_{k|k-1}^{(i)} [H_k^{(i)}]^T (H_k^{(i)} P_{k|k-1}^{(i)} [H_k^{(i)}]^T + R_k)^{-1}. \tag{20}$$

with $\kappa(\mathbf{z})$ the clutter level and R_k the sensor noise covariance. Because the targets are only detectable when in the FOV of the cameras, the detection probability $p_{D,k}$ of camera $l \in 1, 2$ is not constant and instead is equal to:

$$p_{D_l,k}(m_{k|k-1}^{(i)}) = \begin{cases} 0.95 & \text{if } \psi_{c,l} - \frac{\alpha_{d_l}}{2} < \psi_j < \psi_{c,l} + \frac{\alpha_{d_l}}{2} \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

with $\psi_{c,l}$ is the heading of camera l in the world frame and α_{d_l} the field of view. This approach is similar as the one found in [16]. $H_k^{(i)}$ is the linearized sensor model:

$$H_k^{(i)} = \left. \frac{\partial h_k(\mathbf{x}_k)}{\partial \mathbf{x}_k} \right|_{\mathbf{x}_k = m_{k|k-1}^{(i)}}. \tag{22}$$

The camera model is defined as follows: First the location of target i is put in camera's body frame:

$$\mathbf{q}_{b,k}^{(i)} = R_{c,l} \left(\mathbf{q}_k^{(i)} - \mathbf{q} \right) \tag{23}$$

Then the position on the image $[x_p^{(i)}, y_p^{(i)}]^T$ is computed, including camera's distortion:

$$x_p^{(i)} = -F_x x_d^{(i)} + X_0 \tag{24}$$

$$y_p^{(i)} = -F_y y_d^{(i)} + Y_0 \tag{25}$$

$$r_y = \frac{y_{b,k}^{(i)}}{x_{b,k}^{(i)}} \tag{26}$$

$$r_z = \frac{z_{b,k}^{(i)}}{x_{b,k}^{(i)}} \tag{27}$$

$$x_d^{(i)} = \left[k_r r_y + 2P_1 r_y r_z + P_2 \left(s_r + r_y^2 \right) \right] \tag{28}$$

$$y_d^{(i)} = \left[k_r r_z + 2P_2 r_y r_z + P_1 \left(s_r + r_z^2 \right) \right] \tag{29}$$

$$s_r^{(i)} = r_y^2 + r_z^2 \tag{30}$$

$$k_r^{(i)} = 1 + K_1 s_r + K_2 s_r^2 \tag{31}$$

with K_1, K_2, P_1, P_2, F_x, F_y, X_0 and Y_0 are parameters of the camera. The intrinsic parameters of the camera are measured using a dedicated ROS calibration package [17]. The sensor model for target's pixel size $s_p^{(i)}$ has been derived empirically:

$$s_p^{(i)} = \frac{\pi F_x^2 S_t^2}{4 \left(\|q_{b,k}^{(i)}\|^2 + K_S \|q_{b,k}^{(i)}\|^4 \right)} \tag{32}$$

with S_t the marker's size and K_S a distortion parameter.

3) Selection: To keep the problem tractable, the number of Gaussians needs to be limited. This involves two steps: pruning and merging. First, only the Gaussians with a weight more than a certain value are selected: $I = \{i = 1, ..., J_k | w_k^{(i)} > T\}$. Then the Gaussians are merged as follows. First the largest gaussian is selected, that is $j = \text{argmax}_{i \in I} w_k^{(i)}$. Then all Gaussians that are close enough are selected:

$$L := \left\{ i \in I | (m_k^{(i)} - m_k^{(j)})^T (P_k^{(i)})^{-1} (m_k^{(i)} - m_k^{(j)}) \leq U \right\} \tag{33}$$

and are merged into a unique gaussian:

$$\widetilde{w}_k^{(l)} = \sum_{i \in I} w_k^{(i)} \tag{34}$$

$$\widetilde{m}_k^{(l)} = \frac{1}{\widetilde{w}_k^{(l)}} \sum_{i \in I} w_k^{(i)} m_k^{(i)} \tag{35}$$

$$\widetilde{P}_k^{(l)} = \frac{1}{\widetilde{w}_k^{(l)}} \sum_{i \in I} w_k^{(i)} \left(P_k^{(i)} + (\widetilde{m}_k^{(l)} - m_k^{(i)})(\widetilde{m}_k^{(l)} - m_k^{(i)})^T \right) \tag{36}$$

4) Extraction: Extracting the target from the RFS intensity is performed by selection of the largest maximas of the intensity function. For the GM-PHD filter, this reduces to deciding on the Gaussians with weight larger than 0.5, as explained in [10].

D. Collision Avoidance

While the avoidance algorithm proposed in [1] has been initially designed for fixed-wing aircraft, it can be also deployed on more agile rotor-based vehicles as they can follow a fix-wing type dynamics. Proceeding along these lines has an experimental interest as it is easy to carry out experiments with quadrotors in limited indoor space and allow us to easily consider multiple vehicle dynamics. The algorithm proposed in [1] has also the advantage to not rely on velocity estimates of other aircraft, which are poor due to low camera resolution and marker characteristics. However, such a decision has the drawback to not having being designed to take advantage of the native agility of a quadrotor. The design of a more competitive, custom solution for quadrotors is beyond the scope of this paper and will be investigated in the future.

The avoidance algorithm proposed in [1] has been designed for unicycle type vehicle moving in the plane, so the dynamical equation of the vehicle is:

$$\begin{bmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{\psi}_t \end{bmatrix} = \begin{bmatrix} c\cos\psi \\ c\sin\psi \\ u \end{bmatrix} \tag{37}$$

with u the turning rate command and c vehicle's forward speed command. Those commands are given by the following equations:

$$\begin{aligned} u =& K\left(1 - \max_j\left(\beta_{a_j}\beta_{d_j}\right)\right)(\psi_t - \psi) \\ &+ \sum_j \beta_{a_j}\beta_{d_j}\frac{-\pi V_{max}}{d_{ij}} \end{aligned} \tag{38}$$

$$c = V_{max} - \Delta V \max_j\left(\beta_{a_j}\beta_{d_j}\right) \tag{39}$$

with K a constant gain, V_{max} aircraft's maximum forward velocity and ΔV an allowed change in velocity. The $\beta_{a_j}\beta_{d_j}$ are responsible for the smooth transition between the navigation and collision avoidance. They use the range d_j and bearing ψ_j information from target j with state q_j:

$$d_j = \|\mathbf{q}_j - \mathbf{q}\| - r - r_j, \tag{40}$$

$$\psi_j = \operatorname{atan2}\left(\frac{y_j - y}{x_j - x}\right) - \psi. \tag{41}$$

The term β_{d_j} is responsible for the transition when another vehicle enters or leaves quadrotor's sensing range and is given by:

$$\beta_{d_j} = \beta\left(\frac{R_s - 0.4(R_s - R_a)\frac{|\psi_j|}{\alpha_s} - d_j}{R_s - 0.4(R_s - R_a)\frac{|\psi_j|}{\alpha_s} - R_a}\right) \tag{42}$$

with R_s quadrotor's sensing range and R_a a range for with host aircraft will only perform avoidance. In this paper β_{d_j} differs slightly from what was proposed in [1] in order to diminish the avoidance strength on the sides of the quadrotor so as to obtain smoother trajectories. Nonetheless, the guarantee of collision avoidance remains as the proof of it is based on the region where $\beta_{d_j}\beta_{a_j} = 1$, which remains the same as in [1]. The term β_{a_j} is responsible for a smooth transition when the encounter leaves or enters the border of quadrotor's FOV and is given by the following equation:

$$\beta_{a_j} = \beta\left(\frac{\alpha_s - |\psi_j|}{\alpha_s - \frac{\pi}{2}}\right) \tag{43}$$

The $\beta(\cdot)$ function is a smooth monotonically increasing function between 0 and 1 :

$$\beta(a) = \begin{cases} 0 & \text{if } a < 0 \\ f(a) & \text{if } 0 \leq a < 1 \\ 1 & \text{otherwise} \end{cases} \tag{44}$$

$$f(a) = 3a^2 - 2a^3 \tag{45}$$

The navigation to a goal $\mathbf{q}_{d_i}$ is given by the following navigation function:

$$V_i(\mathbf{q}_i) = \|\mathbf{q}_i - \mathbf{q}_{d_i}\|^2 + K_p\beta\left(\frac{\|\mathbf{q}_i - \mathbf{q}_{d_i}\|}{R_p}\right)e_i^2 \tag{46}$$

with K_p some constant gain, $e_i = \frac{\|(\mathbf{q}_{o_i} - \mathbf{q}_{d_i}) \times (\mathbf{q}_i - \mathbf{q}_{d_i})\|}{\|\mathbf{q}_{o_i} - \mathbf{q}_{d_i}\|}$ a term that brings the quadrotor on its desired trajectory defined as a line between its starting point $\mathbf{q}_{o_i}$ and the goal point $\mathbf{q}_{d_i}$. The desired heading ψ_t is defined by the opposite direction of NF's gradient, that is, $\psi_t = \operatorname{atan2}(y_t, x_t)$ with $[x_t, y_t]^T = -\nabla_{q_i} V_i(\mathbf{q})$.

Presented collision avoidance considers motion in the horizontal plane but has no influence on the vertical axis, which can be controlled independently. As we assume that the quadrotors fly at a constant height, the vertical control is done by a proportional velocity control:

$$\dot{z}_t = K_z\left(z_t - z\right) \tag{47}$$

with z_t being the target height. It is desirable to keep the height constant instead of also avoiding on the vertical plan as the quadrotor create a strong downward airflow. This airflow reduces the controllability of the quadrotor and should be avoided. The desired velocity $\mathbf{v}_t = [\dot{x}_t, \dot{y}_t, \dot{z}_t]^T$ and desired yaw rate $\dot{\psi}_t$ are then sent to the low level control node.

E. Low-level Control

The low-level control node is responsible of translating the desired velocity and yaw rate in commands understandable by the quadrotor. The Hummingbird quadrotor is already performing the attitude control, leaving us to compute the desired thrust and attitude. The inputs are the thrust, roll

and pitch angles, and the yaw rate. The yaw rate from the collision avoidance algorithm is directly fed to the quadrotor.

The velocity is controlled by acceleration. First, the control acceleration $\mathbf{a}_c$, needed to obtain the desired velocity, is computed:

$$\mathbf{a}_c = K_v(\mathbf{v}_t - \mathbf{v}) + mg\mathbf{z} + \mathbf{a}_t \tag{48}$$

with $mg\mathbf{z}$ the gravity compensation and and a_t quadrotor's target acceleration. This target acceleration is useful to bring the quadrotor's dynamics closer to a unicycle type vehicle as used in [1] by adding a radial acceleration:

$$\mathbf{a}_t = uc \begin{bmatrix} -\sin\psi \\ \cos\psi \\ 0 \end{bmatrix} \tag{49}$$

with u and c given by Equations 38 and 39 respectively. The control acceleration is used to compute the thrust and the roll and pitch angles. To be consistent with the quadrotor commands, the control acceleration first needs to be aligned with quadrotor's yaw angle

$$\mathbf{a}_b = R_\psi^T \mathbf{a}_c. \tag{50}$$

The thrust is then given by the vertical component of control acceleration $a_{b,z}$. The thrust is also compensated for the orientation of the quadrotor to ensure that its real vertical acceleration corresponds to the control acceleration:

$$u = \frac{a_{b,z}}{\cos(\phi)\cos(\theta)} \tag{51}$$

The roll ϕ_c and pitch θ_c commands are given by the control acceleration direction

$$\phi_c = \arctan(a_{b,z}, a_{b,y}) \tag{52}$$

$$\theta_c = \arctan(a_{b,z}, a_{b,x}). \tag{53}$$

The thrust T, roll angle ϕ_c, pitch angle θ_c and yaw rate are then sent to the quadrotor using *asctec_hl_interface* package [15].

IV. Experiments

A. Experimental Setup

The experiments have been realized indoors. To replace the GPS+IMU unit, we used a MCS from Motion Analysis consisting of 20 Osprey cameras to determine the pose of the quadrotor. The room's useful volume is about 4 by 7 meters and 2.5 meters high. Each quadrotor gets its position and orientation through WiFi. The WiFi is only used to send to the quadrotor its pose and commands from the operator. There is no communication from quadrotor to quadrotor.

B. Scenarios

Four scenarios were designed to validate the proposed system. They are illustrated in Figure 5. In the first scenario, called "Head-on", a quadrotor starts at $[0, -1.6, 1.2]m$ and aims to go to $[0, 1.6, 1.2]m$, and vice-versa for the other quadrotor. The second scenario, called "Cross", is a crossing scenario where one quadrotor starts at $[0.8, 1.6, 1.2]m$ and with the target position at $[-0.8, -1.6, 1.2]m$. The other quadrotor starts at $[0.8, -1.6, 1.2]m$ and aims to go to $[-0.8, 1.6, 1.2]m$. In the third scenario, called "Side", one quadrotor starts at $[0.9, -1.6, 1.2]m$ and aims to go to $[-0.9, 1.6, 1.2]m$. The other quadrotor starts at $[-0.9, -1.6, 1.2]m$ and aims to go to $[0.9, 1.6, 1.2]m$. The final scenario, named "Three", involves three quadrotors to show the capability of our system in handling multiple encounters simultaneously. Two quadrotors have the same start and goal positions as in the second scenario, and the third quadrotor starts at $[-1, 0, 1.2]m$ and the goal at $[1, 0, 1.2]m$.

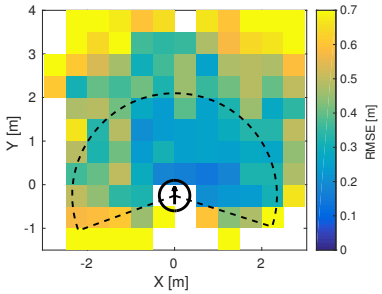

Fig. 4. Tracking error of a single target obtained by systematically moving a quadrotor in front of another quadrotor with a fix position. The black circle shows the quadrotor position, the black arrow is the heading and the dashed lines the region used for avoidance.

The parameters for the tracking and collision avoidance algorithms were the same for all scenarios. The birth model for the GM-PHD filter is a sum of three Gaussians of weight 0.01 at positions $[-1, 0.2, 0]m$, $[1, 0.2, 0]m$ and $[0, 1.2, 0]m$ (in quadrotor's body frame). Their birth velocity is zero. The variance on position is $32[m^2]$ and $4[m^2/s^2]$ for velocity. The motion model is given by:

$$F_k = \begin{bmatrix} I_3 & \Delta I_3 \\ 0_3 & I_3 \end{bmatrix}, Q_k = \begin{bmatrix} \frac{\Delta^4 \sigma_m}{4} I_3 & \frac{\Delta^3 \sigma_m}{2} I_3 \\ \frac{\Delta^3 \sigma_m}{2} I_3 & \Delta^2 \sigma_m I_3 \end{bmatrix} \tag{54}$$

with $\sigma_m = 0.5$ and the time delay $\Delta = 10ms$. The measurement noise is

$$R_k = \begin{bmatrix} 16 & 0 & 0 \\ 0 & 16 & 0 \\ 0 & 0 & 64 \end{bmatrix}. \tag{55}$$

The clutter level is different depending on the measurement size $s_p^{(i)}$

$$\kappa(z) = \begin{cases} \frac{1}{320*240*200} & \text{if } s_p^{(i)} < 10 \\ \frac{0.01}{320*240*200} & \text{otherwise} \end{cases} \tag{56}$$

as clutter has usually only a few pixels. The cameras are pointing left and right from the x axis with an angle of $45°$. The cameras' FOV α_d is $127°$. The half overall FOV α_s is $110°$ The merging parameter $U = 2$ and the Gaussians are pruned if their weight are less than $T = 10^{-5}$. For the collision avoidance algorithm the parameters are: $r = 0.35m$, $V_{max} = 0.3\frac{m}{s}$, $\Delta V = 0.1\frac{m}{s}$, $R_s = 1.2m$, $R_a = 0.4m$, $\alpha_s = 110°$, $K = 0.5$, $K_p = 0.8$.

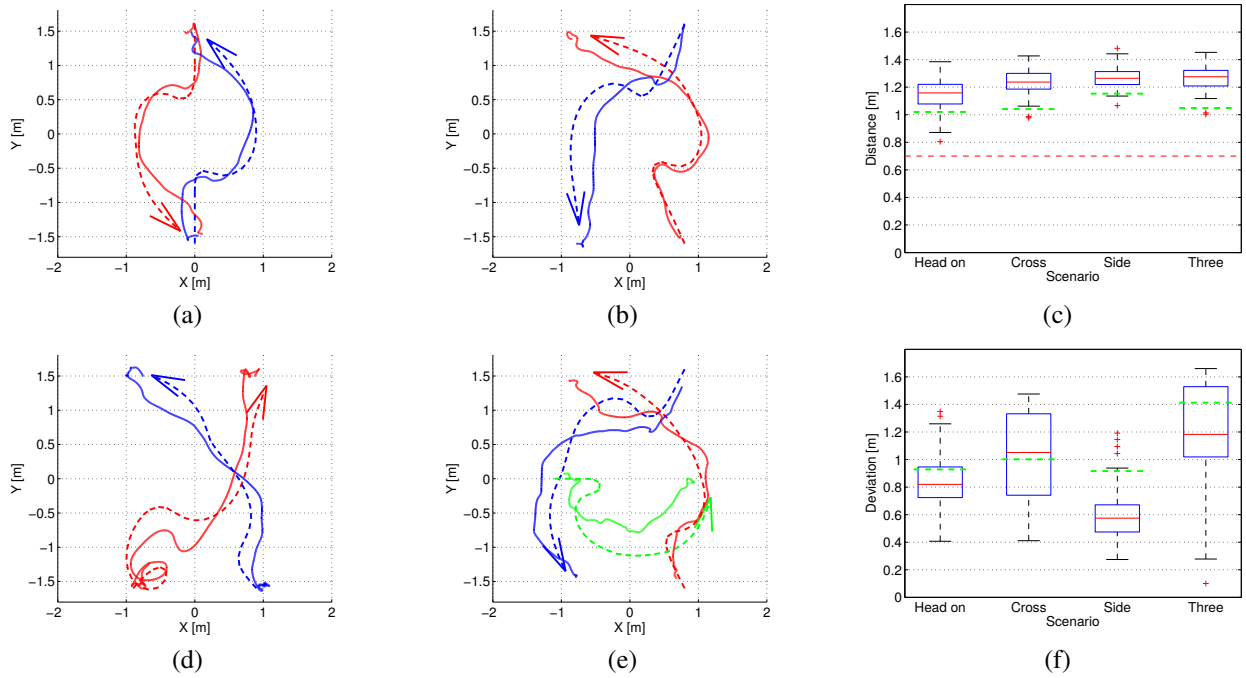

Fig. 5. Plots (a), (b), (d) and (e) show the results obtained during scenarios "Head-on", "Cross", "Side" and "Three" respectively. All solid lines corresponds to real data and all dashed lines to simulated data. The trajectories of each quadrotor are depicted in a different color (blue, red, and green for the first, second, and third quadrotor, respectively). Plot (c) shows the boxplots (red line is the median, the blue box represents first and last quantiles) of the minimal distance between real quadrotors for 50 experiments for all four scenarios. The red dashed line represents the distance for which quadrotors would be considered to have collided. The green dashed line represents the average performance of the simulation (realistic model) for 50 experiments. Plot (f) shows the boxplots of the maximum deviation from desired trajectory for 50 experiments using real quadrotors for all four experiments. The green dashed line represents the average performance of the simulation (realistic model) for 50 experiments.

C. Results

With the parameters' values mentioned above, the system is able to achieve around 10 Hz frame rate for each camera with a 200 milliseconds delay split between capturing time and processing time for each camera. Because we are using ROS, there is a lag of 50 milliseconds between the computation of the command and its execution by the quadrotor.

The sensing is very noisy, as illustrated in Figure 4. It ranges from a RMS error of 0.25 meters at close range to more than a meter at longer distances. This error is mostly due to the noise on the marker size. Indeed the apparent marker size ranges from a few pixels to a few hundred and the marker edge is not smoothed, resulting in a small signal-to-noise ratio. Nonetheless the sensing accuracy is enough in the region used for the collision avoidance (dashed lines in Figure 4).

Representative trajectories for the four scenarios are shown in Figure 5 (solid lines). Fifty collision course runs (using real quadrotors) have been performed for each scenario. The quadrotors never collided during the 200 runs. Figure 5 shows, overlapped to real robot data, also trajectories obtained with a point-mass, microscopic simulator implemented in Matlab and already leveraged in [1]. In order to shed further lights on the various source of errors, in contrast to our previous work, we have extended our simulator to include real world effects in terms of sensing, actuation, and computation. In particular, the simulation has been adapted to include a Gaussian noise on the sensing of 0.25 meters RMS, a sensing lag of 200 milliseconds, a control lag of 50 milliseconds. Also the model of the unicycle in the simulation has been adapted to include inertia. The aircraft in the simulator has thus the dynamical equation:

$$\begin{bmatrix} \dot{x_t} \\ \dot{y_t} \\ \dot{\psi_t} \\ \dot{\omega_t} \end{bmatrix} = \begin{bmatrix} c\cos\psi \\ c\sin\psi \\ \omega_t \\ -3(\omega_t - u) \end{bmatrix} \tag{57}$$

instead of Equation 37. A saturation on yaw acceleration of $3\frac{rad}{s^2}$ was also implemented. The obtained trajectories are less smooth than what was presented in [1] for two reasons. First, the sensing range R_S is smaller compared to the vehicle size than in [1] which results in sharper trajectories. Second, our system is now affected by real-world imperfections (noise, lag). This can drastically change the trajectory of the quadrotors as for the "Side" scenario the left quadrotor prefers to do a full 360° turn to avoid the other quadrotor. The interplay of those real-world effects can drastically modify obtained trajectories, as illustrated in Figure 6. It appears to be mostly due to the lag (both in sensing and control) as the 360° turn behavior does not appear if the sensing noise is applied without any lag.

This sensibility to lag is inherent to the collision avoidance algorithm. Indeed, when another quadrotor enters the collision avoidance zone of the host quadrotor, the later will turn until there is an equilibrium between the avoidance and the navigation component. This equilibrium happens mostly

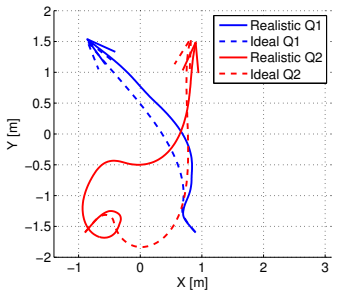

Fig. 6. Illustration of the trajectories of the noise-free and lag-free model (in dashed lines) and the realistic model that includes noise, lag, inertia and saturation (solid lines) for the "Side" scenario.

because the angular transition function β_{a_j} tends to 0 (as the bearing angle changes faster than the distance). But the region in which β_{a_j} ranges between 0 and 1 corresponds only to 20° in bearing. Since there is lag in sensing and control, the quadrotor overshoots the yaw angle. At overshooting, the quadrotor will switch behavior and go back to navigation. In the "Side" scenario case, the overshoot brings the quadrotor to a state where following the navigation function leads to a 360° turn.

The minimum horizontal distance between the aircraft during the runs extracted from the MCS are shown in Figure 5. The average minimum distance obtained in simulation is lower by approximately 20 centimeters for all scenarios. This might be due to unmodeled phenomenons, such as the airflow generated laterally by the quadrotors that tends to push them away from each other. The distances are well above the distance for which the quadrotors will be considered to have collided. As a second metric, the maximum deviation of the quadrotors from their desired course (virtual line between start point and goal) over a run has also been recorded (Figure 5f). Even if the simulations are quite close to the real trajectory for three out of four scenarios, there is a significant difference for the "Side" scenario resulting on a larger delta between the two mean results on the metric of Figure 5f. This is because, in both simulation and reality, the two behaviors illustrated in Figure 6 are present but not with the same weight. In reality, the most common case is where the quadrotor does a 360° turn. In simulation, the smoother trajectory is more common. The switching behavior is hard to capture, and has significant effect on the metric.

V. Conclusion

In this paper, we proposed a collision avoidance system for small UAVs. The system is composed of a small on-board computation unit and two cameras embedded in a large marker. Color segmentation is used for obstacle detection for the time being but in the future a more general algorithm can be used as long as sufficient computational resources are available on-board. A GM-PHD filter is used for sensor fusion and filtering. A previously published promising algorithm based on navigation functions and able to natively take into account limited FOV sensing was adapted for coping with real-world effects and ported to reality. The solution has been implemented on a real robotic system consisting of up to three quadrotors and validated with four different scenarios, including one with multiple targets to avoid. All computation was done on-board, except the host position information that was given by a MCS. A thorough analysis of the system has been performed and showed that while the algorithm is robust to sensing noise, it is quite sensitive to lag in sensing and control.

The proposed system should be able to work with a GPS/IMU unit instead of a MCS for outdoor flights. Future work includes improving the collision avoidance algorithm in order to generate smoother trajectories and being less sensitive to sensing and control lag. To further systematically validate the algorithms, we intend to test it in a high-fidelity simulator under a variety of scenarios.

References

[1] S. Roelofsen, A. Martinoli, and D. Gillet, "Distributed deconfliction algorithm for unmanned aerial vehicles with limited range and field of view sensors," in *American Control Conference*, 2015, pp. 4356–4361.

[2] B. C. Karhoff, J. I. Limb, S. Oravsky, and A. D. Shephard, "Eyes in the domestic sky: an assessment of sense and avoid technology for the army's" warrior" unmanned aerial vehicle," in *IEEE Systems and Information Engineering Design Symposium*, 2006, pp. 36–42.

[3] A. Mcfadyen, P. Corke, and L. Mejias, "Rotorcraft collision avoidance using spherical image-based visual servoing and single point features," in *IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 1199–1205.

[4] A. Mcfadyen, L. Mejias, P. Corke, and C. Pradalier, "Aircraft collision avoidance using spherical visual predictive control and single point features," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 50–56.

[5] A. Mcfadyen, P. Corke, and L. Mejias, "Visual predictive control of spiral motion," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1441–1454, Dec 2014.

[6] P. Conroy, D. Bareiss, M. Beall, and J. van den Berg, "3-D reciprocal collision avoidance on physical quadrotor helicopters with on-board sensing for relative positioning," *arXiv preprint arXiv:1411.3794*, 2014.

[7] Y. Bar-Shalom, T. Fortmann, and M. Scheffe, "Joint probabilistic data association for multiple targets in clutter," in *Proc. Conf. on Information Sciences and Systems*, 1980, pp. 404–409.

[8] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.

[9] R. P. Mahler, *Statistical multisource-multitarget information fusion*. Artech House, Inc., 2007.

[10] B.-N. Vo and W.-K. Ma, "The gaussian mixture probability hypothesis density filter," *IEEE Trans. on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, 2006.

[11] P. Panyakeow and M. Mesbahi, "Decentralized deconfliction algorithms for unicycle UAVs," in *American Control Conference*, June 2010, pp. 794–799.

[12] G. Roussos, D. V. Dimarogonas, and K. J. Kyriakopoulos, "3d navigation and collision avoidance for nonholonomic aircraft-like vehicles," *International Journal of Adaptive Control and Signal Processing*, vol. 24, no. 10, pp. 900–920, 2010.

[13] D. E. Chang and J. E. Marsden, "Gyroscopic forces and collision avoidance with convex obstacles," in *New trends in nonlinear dynamics and control and their applications*, 2003, pp. 145–159.

[14] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, pp. 1–21, 2015.

[15] "asctec_hl_interface ROS package," http://wiki.ros.org/asctec_hl_interface, 2015-02-04.

[16] K. Granstrom, S. Reuter, D. Meissner, and A. Scheel, "A multiple model PHD approach to tracking of cars under an assumed rectangular shape," in *IEEE 17th International Conference on Information Fusion*, 2014.

[17] "camera_calibration," http://wiki.ros.org/camera_calibration, 2015-02-25.