2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Vision-Based Unmanned Aerial Vehicle Detection and Tracking for Sense and Avoid Systems

Krishna Raj Sapkota[1], Steven Roelofsen[2,3], Artem Rozantsev[1],
Vincent Lepetit[1,4], Denis Gillet[3], Pascal Fua[1] and Alcherio Martinoli[2]

***Abstract*— We propose an approach for on-line detection of small Unmanned Aerial Vehicles (UAVs) and estimation of their relative positions and velocities in the 3D environment from a single moving camera in the context of sense and avoid systems. This problem is challenging both from a detection point of view, as there are no markers on the targets available, and from a tracking perspective, due to misdetection and false positives. Furthermore, the methods need to be computationally light, despite the complexity of computer vision algorithms, to be used on UAVs with limited payload.**

To address these issues we propose a multi-staged framework that incorporates fast object detection using an AdaBoost-based approach, coupled with an on-line visual-based tracking algorithm and a recent sensor fusion and state estimation method. Our framework allows for achieving real-time performance with accurate object detection and tracking without any need of markers and customized, high-performing hardware resources.

I. Introduction

The area of small UAVs has experienced a tremendous growth in the recent years. UAVs have already been used in various applications ranging from delivery of goods to aerial photography. In the recent years, most of the research community's attention was centered on a few topics, including self-localization [1], path planning [2] and navigation [3], [4]. On the other hand, Sense and Avoid (SAA) neighboring aircraft remained relatively out of focus, due to computational complexity of most of the object detection algorithms or sensor cost.

The ability to detect and estimate the relative position of neighboring aircraft plays a crucial role in automated flight, central to such tasks as mid-air collision avoidance and formation flying [5]. Vision-based relative positioning is of particular interest as cameras generally require less power consumption and are more lightweight than active sensor alternatives such as radars and lasers. It also has potential application in non-collaborative flight scenarios or in situations where collision warning based on Global Navigation Satellite Systems are either unreliable or not commonly available on all aircraft. Marker-based visual solutions [5], [6] would require every aircraft to be equipped with such marker system and therefore lack of generality.

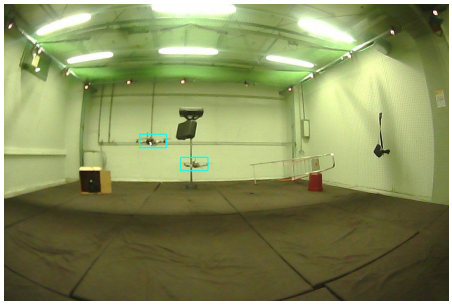


Fig. 1. Picture of the experimental environment, as seen from the UAV carrying the camera. The blue boxes visualize detections produced by the computer vision algorithm. Best seen in color.

Vision-based target tracking poses several challenges. Unlike other detection tasks a missed detection can be hazardous. A high detection accuracy is therefore required. UAVs travel at relatively high speeds and must be detected quickly, particularly in collision avoidance scenarios. Moreover, neighboring aircraft can appear across a wide range of distances and can be difficult to detect when far away.

Another challenge of vision-based target tracking, in particular for collision avoidance applications, is the multi-target aspect of the problem as it is inherently combinatorial, thus challenging from a computational point of view. Moreover, the number of targets in the scene is unknown for SAA and have to be estimated as well.

In this work, we developed a framework for both the detection of UAVs and the estimation of their relative positions and velocities, based just on single camera video stream. The framework is composed of vision and multi-target state tracker modules. The first one consists of an object tracker working in conjunction with an object detector. The detector runs every 5-10 frames to first initialize and then later on to correct the tracker. Visual tracking is used in between detections to keep track of the object, the rationale being that it is computationally cheaper than detection and is unlikely to fail within a few frames. Based on the output of the vision module, the multi-target state tracker module estimates positions and velocities of flying objects in the 3-dimensional environment. We assess the accuracy of the proposed framework in an indoor setup (shown in Fig. 1), leveraging equipment that allows for millimetric precision ground-truth. We compare the results with and without the visual tracking algorithm

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

[1] K. Sapkota, A. Rozantsev and P. Fua are with the Computer Vision Laboratory, School of Communication and Computer Sciences, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland.

[2] S. Roelofsen and A. Martinoli are with the Distributed Intelligent Systems and Algorithms Laboratory, School of Architecture, Civil and Environmental Engineering, EPFL, Switzerland.

[3] S. Roelofsen and D. Gillet are with the Coordination and Interaction System Group, School of Engineering, EPFL, Switzerland.

[4] V. Lepetit is with the Institute for Computer Graphics and Vision, TU Graz, Austria.

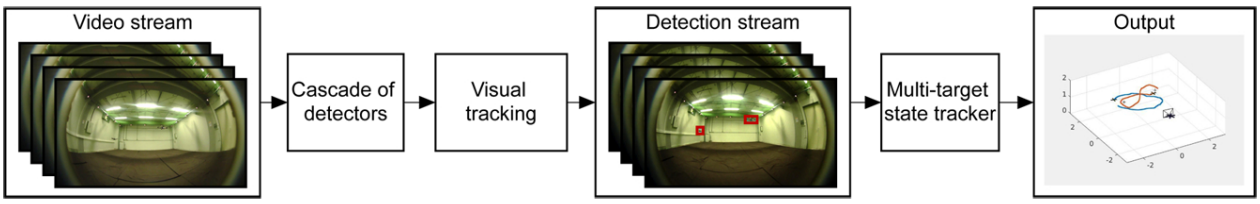


Fig. 2. Overview of different stages of our approach. Best seen in color.

enabled.

The paper is organized as follows: Section II presents related work on the different topics leveraged in our framework. Section III describes the different algorithms used in our framework. Section IV depicts the experimental setup. Results are presented in Section V.

II. Related Work

Previous works on tracking of UAVs for SAA systems, such as [7] rely on relatively simple detection techniques that are capable of finding objects in the sky. In [8] a hidden Markov model filter is used to track aircraft in the image. Both works focus on image processing, without clearly stating the tracking accuracy in position and velocity. Both works also require computational power usually not available on small UAVs.

We approach the object detection task with an AdaBoost algorithm, similar to [9], [10], which has been demonstrated to perform well across variety of detection tasks [11], [12], [13] and has low computational complexity, which is required for on-line performance. We further on improve detection accuracy and robustness with the visual-based tracking algorithm [14]. Unlike the other approaches [15], [13], [16], this one allows for a dynamic adaptation to the changing appearance of the object and is computationally effective.

For multi-target state tracking we use Random Finite Set theory that extends the Bayesian framework to multiple targets [17]. That theory has led to effective algorithms, both in terms of tracking capability and computational power. For this reason, the state estimation in this framework is performed with a Gaussian-Mixture Probability Hypothesis Density filter (GM-PHD) described in [18]. The GM-PHD filter is a multi-target tracker that has been successfully applied to marker-based single-camera multi-target tracking in previous work [19]. In this paper, we apply the same methodology to the more general problem of marker-less multi-target tracking.

III. Proposed Framework

In this section, we first briefly describe our framework and then more thoroughly discuss all its components. Fig. 2 depicts the overview of our method. The framework consists of a vision module that incorporates detection together with a visual-based tracking, which filters out false detections as well as fills up missed ones. These detections are then sent to the state tracker, which estimates the three dimensional position and velocity of the targets using the measured 2D locations in the image and the apparent width of the target. The framework outputs the position and velocity of UAVs in the scene.

A. Object Detection

For object detection we use the classical cascade of detectors trained using HOG (Histogram of Oriented Gradients) features. It consists of ensemble learners trained in a greedy manner via the AdaBoost algorithm. Object detection works by sliding a patch (detection window) all over the image at different scales (to account for different object sizes) and classifying the patch as containing or not containing the object. Object detection is hence a binary classification task but since it has to be carried out at all image locations and at different scales, it is computationally expensive.

Cascade of detectors reduces the computational complexity by exploiting the fact that usually images contain more non-object patches than object patches. The former ones can thus be rejected at an earlier stage with less complex detectors. Hence ensemble learners earlier in the cascade are very simple (with only few features) and can reject most non-object patches so that more complex learners down the cascade can focus on correctly classifying difficult patches.

Each feature (for HOG that would be one of the histogram bins) forms a decision stump (weak learner)

$$h_j(X) = \begin{cases} 1, f_j(X) < \theta_j \\ 0, \text{otherwise} \end{cases}, \tag{1}$$

where, X is the image patch, $f_j(X)$ is the j^{th} feature of the patch, θ_j is the decision threshold, and $h_j(X)$ is the weak learner corresponding to the j^{th} feature.

Each cascade is an ensemble of such weak learners and can be represented as

$$D(X) = \sum_{j=1}^{N} \alpha_j h_j(X), \qquad \alpha_j = \log\left(\frac{1-\epsilon_j}{\epsilon_j}\right), \tag{2}$$

where N is the number of weak learners $h_j(\cdot)$ and ϵ_j is the classification error at the j^{th} iteration of the AdaBoost training.

Fig. 3 shows a cascade of ensemble learners. If an image patch is rejected at any stage (classified as not containing object) of the cascade, it is not processed further. Thus, only image patches that make it all the way through the entire cascade are labeled as containing object. Hence it is important for the ensemble learners in the cascade to have very low false negative rate as samples falsely labeled as negatives are not further processed and hence such mistakes cannot be later refined.

As it is often done in computer vision, we apply non-maximum suppression, which effectively looks for the neighborhood of every detection and prunes it if a detection with

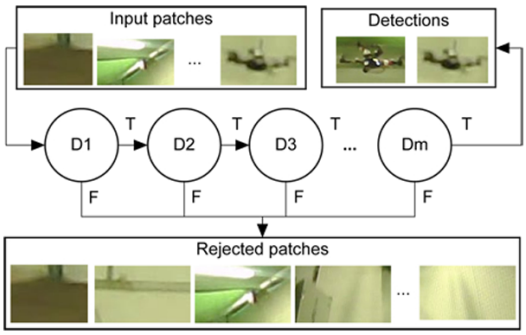


Fig. 3. Cascade of detectors $D_1..D_m$. Patches that are rejected at any stage of the cascade are considered as negatives. On the other hand, patches that make it all the way through the cascade are viewed as positives and are further processed by the visual tracking system.

higher score is found. This allows us to significantly reduce the number of repeating detections; however, it may, for example, lead to accidentally removing the evidence for a UAV partly occluded by another one. As our target application is collision avoidance, the aforementioned behavior is not critical, as in this case the UAV that is closer to the camera will still be reliably detected, because it will likely have more evidence, in comparison to that partly occluded.

B. Visual tracking

For visual-based object tracking we employ an online boosting tracker [20]. Briefly, it learns an appearance model of the target, given an initial image and a target bounding box. In consecutive frames, patches around the current bounding box are sampled and the target bounding box is moved to the location with the highest response (see Fig. 4(a) and Fig. 4(b)). The appearance model is then updated with the patch from the new target location as a positive sample and patches around the target as negative ones. For a more detailed explanation of online boosting please refer to [14].

One of the limitations of such a technique is that the update phase might introduce errors if the target is not correctly localized, an issue that will make the tracker drift away from the target over time. To mitigate such problem, we correct the tracker on regular intervals (every 5-10 frames) using an offline trained detector, introduced in the previous section. Note that this correction does not reset the model learned online, but rather it adds positive examples, provided with the help of a more reliable detection system.

Another drawback of this technique is the sampling approach, as it does not take into account the motion of the observed object. Hence, we introduce an optimization step that uses motion-guided sampling to reduce the search space. Assuming that the velocity of the object does not change much in consecutive frames, we employ a motion model based on a Kalman filter to predict the location of the object. We further compute the covariance of the prediction and sample based on this covariance centered around the predicted location. While effective, this approach may lead to losing track of the UAV in case the camera orientation changes sharply; however, in such situations the track will be rapidly reinitialized with a new detection.

Such motion-guided sampling has several benefits with respect to the original uniform sampling:

- significant reduction of search space;
- successful tracking of objects moving at high velocities;
- reduction in identity swaps for similarly looking objects;

As a side benefit, using motion-guided sampling, we were able to achieve a tracking speed-up by a factor of two without any degradation in performance, due to a ten times reduction in the number of samples. Fig. 4(c) illustrates how the motion model helps to significantly reduce the search space while looking for objects in the next frame.

C. Visual Tracking Details

We have now introduced two methods for object detection and tracking, respectively. In order to effectively use them together we need to do data association between detections and tracks. We employ the Hungarian algorithm [21] to solve this problem. As discussed above, the detection algorithm is running at $0.1 - 0.2Hz$, thus for the frames where detections are not available, tracks are propagated according to the online boosting model.

Apart from a computational speed up, other important benefit of tracking is that it outputs a single detection per object, which is one of the assumptions required for the GM-PHD filter application, as described in Section III-D.

As our approach is based on machine learning it has the limitation of being able to detect objects that look similar to the ones that are present in the training set. However, it is relatively easy to add another class of objects, which allows the detector not only to localize the drone, but also to distinguish between different types of UAVs, which might be important for some applications.

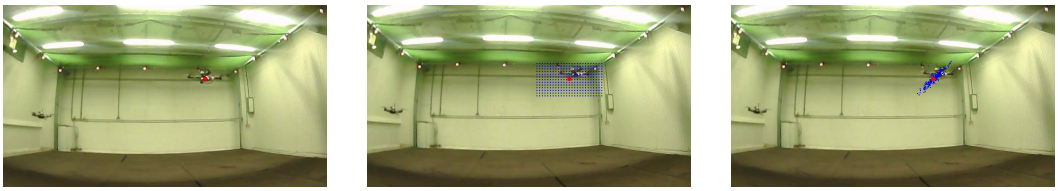
D. Multi-Target State Tracking

In this section, we present our implementation of the GM-PHD filter for multi-target state tracking of UAVs with a single camera as a sensing device. The first part will be a quick introduction about this filtering technique, For more insight of the GM-PHD filter, please refer to [17] and [18] . The second part will focus on the implementation.

1) Probability Hypothesis Density Filter: The main concept behind the GM-PHD filter, and Probability Hypothesis Density filters in general, is to estimate a map of the density, called intensity, of targets present in the environment instead of single, distinguishable tracks. This allows us to estimate the state of all targets in a single state space $\mathcal{X}$ instead of its power set $\mathcal{F}(\mathcal{X})$ as it would be the case for the classical multi-target problem. The second specificity of the GM-PHD filter is that the targets are described using a mixture of Gaussians. That is, the intensity v_k at time k for state $\mathbf{x}$ is given by

$$v_k(\mathbf{x}) = \sum_{i=1}^{J_k} w_k^{(i)} \mathcal{N}(\mathbf{x}; m_k^{(i)}, P_k^{(i)}), \tag{3}$$

with $w_k^{(i)}$ the weight of the Gaussian i, $m_k^{(i)}$ its mean and $P_k^{(i)}$ its covariance. The assumption of using a mixture of



(a) Original frame (t_0) (b) Uniform sampling ($t_0 + 1$) (c) Motion model ($t_0 + 1$)

Fig. 4. Sampling algorithms for finding the location of the object at $t_0 + 1$ frame, based on its position at t_0, which is marked with a red dot. **(a)** depicts the original frame at step t_0. **(b)** illustrates the conventional uniform sampling around the position of the object. **(c)** search space can be significantly reduced using motion-model-based sampling. Best seen in color.

Gaussians allows reformulating the problem to use Kalman filtering to predict and update each Gaussian.

The prediction step for the GM-PHD filter is computed as

$$v_{k|k-1}(\mathbf{x}) = \sum_{i=1}^{J_k} p_{S,k} w_k^{(i)} \mathcal{N}(\mathbf{x}; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}) + \gamma_k(\mathbf{x}), \tag{4}$$

with

$$m_{k|k-1}^{(i)} = F_{k-1} m_{k-1}^{(i)} \tag{5}$$

$$P_{k|k-1}^{(i)} = Q_{k-1} + F_{k-1} P_{k-1}^{(i)} F_{k-1}^T, \tag{6}$$

where F_{k-1} is the motion model from time $k-1$ to k and Q_{k-1} is the process noise during that interval. $\gamma_k(\mathbf{x})$ is the intensity of new targets appearing, described as a Mixture of Gaussians. In this work, $\gamma_k(\mathbf{x})$ is implemented as a single Gaussian two meters in front of the camera. $p_{S,k}$ is the probability that a target survives between time $k-1$ and k. The update step is

$$v_k(\mathbf{x}) = \sum_{i=1}^{J_k} \left(1 - p_{D,k} w_{k|k-1}^{(i)} \mathcal{N}(\mathbf{x}; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)})\right) + \sum_{\mathbf{z} \in Z_k} \sum_{i=1}^{J_k} w_k^{(i)}(\mathbf{z}) \mathcal{N}(\mathbf{x}; m_{k|k}^{(i)}(\mathbf{z}), P_{k|k}^{(i)}), \tag{7}$$

with

$$w_k^{(i)}(\mathbf{z}) = \frac{p_{D,k}(m_{k|k-1}^{(i)}) w_{k|k-1}^{(i)} l_k^{(i)}(\mathbf{z})}{\kappa(z) + \sum_{i=1}^{J_k} p_{D,k}(m_{k|k-1}^{(i)}) w_{k|k-1}^{(i)} l_k^{(i)}(\mathbf{z})} \tag{8}$$

$$l_k^{(i)}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; H_k^{(i)} m_{k|k-1}^{(i)}, R_k + H_k^{(i)} P_{k|k-1}^{(i)} [H_k^{(i)}]^T) \tag{9}$$

$$m_{k|k}^{(i)}(\mathbf{z}) = m_{k|k-1}^{(i)} + K_k^{(i)}(\mathbf{z} - H_k^{(i)} m_{k|k-1}^{(i)}) \tag{10}$$

$$P_{k|k}^{(i)} = \left[I - K_k^{(i)} H_k^{(i)}\right] P_{k|k-1}^{(i)} \tag{11}$$

$$K_k^{(i)} = P_{k|k-1}^{(i)} [H_k^{(i)}]^T (H_k^{(i)} P_{k|k-1}^{(i)} [H_k^{(i)}]^T + R_k)^{-1}, \tag{12}$$

where $\kappa(\mathbf{z})$ is the clutter level and R_k the sensor noise covariance. $p_{D,k}$ is the probability that a target i is detected and $H_k^{(i)}$ is the matrix that transforms the target's state to a measurement. Note that Equations 10, 11 and 12 are similar to a Kalman filter update.

2) Implementation: In our framework, the computer vision module returns the bounding boxes of detected objects. Due to the fact that width and height of these bounding boxes are heavily correlated, the measurement of a target is $z = [x_c, y_c, w_c]^T$ with the components being the detection center in horizontal and vertical coordinate and the width of the bounding box, respectively. A sample of the measurements is shown in Fig. 5. The implementation of the camera sensor model is similar to [19], which uses the standard camera with distortion model, with the exception of the measurement model for the apparent width $w_c^{(i)}$ of target i at time k. The equations for our sensor model are

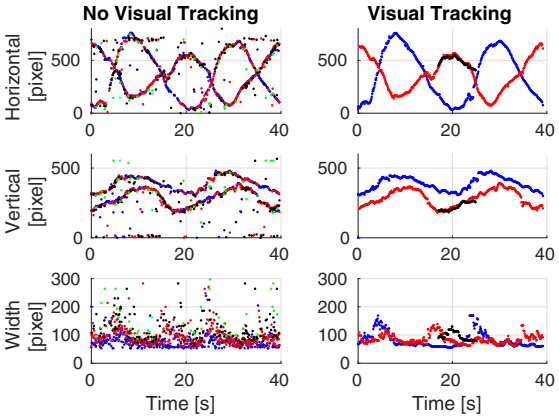


Fig. 5. Position estimates obtained from the computer vision algorithm. The data on the left is obtained from the cascade of detectors with non-maximum suppression of 1, and the data on the right is from the visual tracking algorithm with non-maximum suppression set to 3. Different colors correspond to different measurement for the same timestep with this order: blue, red, black, green, cyan. Best seen in color.

$$x_c^{(i)} = F_x x_d^{(i)} + X_0 \tag{13}$$

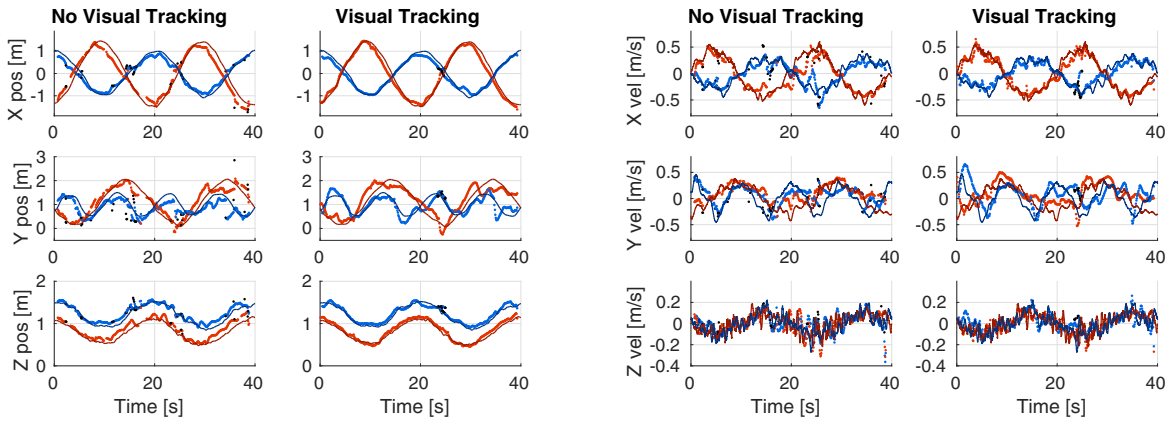
$$y_c^{(i)} = F_y y_d^{(i)} + Y_0 \tag{14}$$

$$w_c^{(i)} = \frac{F_x W_t (1 + K_1 s_r)}{\left(\|q_{b,k}^{(i)}\| + K_w \|q_{b,k}^{(i)}\|^2\right)} \tag{15}$$

$$\left[x_d^{(i)}, y_d^{(i)}\right] = \rho\,[r_x, r_y], \quad \rho = \left(1 + K_1 s_r + K_2 s_r^2\right) \tag{16}$$

$$r_x = \frac{x_{b,k}^{(i)}}{z_{b,k}^{(i)}}, \quad r_y = \frac{y_{b,k}^{(i)}}{z_{b,k}^{(i)}}, \quad s_r = r_x^2 + r_y^2, \tag{17}$$

with $q_{b,k}^{(i)} = \left[x_{b,k}^{(i)}, y_{b,k}^{(i)}, z_{b,k}^{(i)}\right]^T$ the position of the target relative to the camera in the camera frame. K_1, K_2, F_x, F_y, X_0 and Y_0 are the standard intrinsic camera parameters and were determined using OpenCV. W_t is the size of the target, and is assumed to be known. K_w is a parameter related to the change of the apparent width as function of the target's distance to the camera and has been determined



(a) Position estimates (b) Velocity estimates

Fig. 6. Results for our experiment with two quadrotors (blue and red track, respectively) in a cluttered environment. The solid, thinner lines are the true trajectories of the quadrotors acquired by the MCS. The dots are the estimated positions from the GM-PHD filter. The red and blue colored dots are the ones closest to the red and blue curves, respectively. The data association is obtained during the computation of the OSPA metric and not as a result of the GM-PHD filter. The black dots are the estimates that were not associated with a quadrotor. Best seen in color.

empirically. This parameter was added to compensate a discrepancy between the model and observed measurements where the cascade of detectors would report targets larger than reality when the quadrotors are far away. As the camera sensor model is not linear, the GM-PHD filter is implemented as an Extended Kalman PHD filter [18].

After updating, a pruning and merging step is performed as described in [18] in the following manner: Gaussians with a weight less than 10^{-5} are pruned and two Gaussians closer than two times their covariances are merged. Finally, all Gaussians that have a weight higher than 0.5 are reported as being targets.

IV. Implementation

The experimental setup is composed of an AscTec Firefly quadrotor carrying a camera and two AscTec Hummingbird quadrotors flying in front as targets of the tracking framework. The camera is equipped with a fish-eye lens with a field of view of $185°$ (an example of a video frame from the accompanying video[1] can be seen in Fig. 1.). All quadrotors flew autonomously using localization data acquired by an external Motion Capture System (MCS) and sent through WiFi to the vehicles. The MCS allows for tracking the pose of the quadrotors with millimetric level accuracy and is used as ground-truth to assess the performance of our framework. The trajectories of the Hummingbird quadrotors were set as follows: for one, an ellipsoid of 3×2 *m* and for the other one an ∞-shape of 2×1 *m*. The quadrotors performed their loop in 20 *s*. The Firefly was mostly in static flight, moving to another position from time to time. All the data were recorded on the Firefly to be post-processed by leveraging the rosbag functionality of the Robotic Operating System (ROS).

In the post-processing phase, the data were played back in real time and fed to the computer vision and tracking algorithms. The computation was performed on a desktop computer with an Intel Core i7, an hardware comparable to the high-end computation modules for quadrotors (e.g., AscTec Mastermind).

Section V presents the results of our framework with and without visual-based tracking. Although both methods operate on the same video stream, the parameters were set differently for fair comparison. An important difference resides in the neighborhood size of the non-maximum suppression step, which is set to 3 for the case with visual-tracking and 1 in case without. This is because visual tracking is more sensitive to false positives, as opposed to the multi-target state tracking which is more sensitive to missed detections. Other differences were in the parameters used in the multi-target state tracker. With visual tracking, the parameters were $p_D = 0.95$, $\kappa(z) = 3 \cdot 10^{-9}$ and $R = \mathrm{diag}(200, 200, 1600)$ and without they were $p_D = 0.5$, $\kappa(z) = 2.4 \cdot 10^{-9}$ and $R = \mathrm{diag}(200, 200, 3200)$.

V. Results

In this section, we present the results of our tracking framework. We compare two cases: the case for which the visual tracking is enabled, and the case for which it is disabled. The obtained trajectories for both cases are shown in Fig. 6(a). Fig. 6(b) shows estimated velocities for both cases. Both figures show the estimates as dots and the ground-truth as solid lines. The dots are colored according to the ground-truth they are closest to. On the one hand, the trajectories obtained without visual tracking are not only less accurate than those with visual tracking, but they do also suffer of target disappearing. This is due to the cascade of detectors not reliably detecting a target, and when one is not detected, the multi-target state tracker immediately reacts by lowering the weight of the estimate below the target existence threshold. On the other hand, the visual tracking will search the image for features that it learned previously. As a result, the algorithm returns a measurement for each quadrotor at each frame. This consistency allows the multi-target state tracker to correctly track both quadrotors and get a smooth estimate of their velocity.

[1]The video and more information about the project can be found at http://disal.epfl.ch/research/UAVCollisionAvoidance

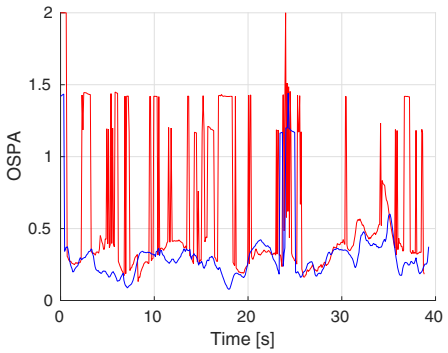


Fig. 7. Obtained OSPA [22] measurement for tracked position of the quadrotors. Blue curve is the OSPA for the case with visual-tracking enabled, red curve is for the case where it is disabled. Lower is better. The error is 2 at maximum. Best seen in color.

One limitation of the system comes from the interaction between the visual tracker and the multi-target state tracker. An example can be seen in Fig. 5 around 25 *s* where tracks are created from false detections. This happens due to the fact that when a track is being created by the visual tracking module, it stays for about 20 frames. This leads to its interpretation as a target by the state tracker, as depicted by Fig. 6. A possible solution will be to merge visual tracking and multi-target state tracking, which is one of our further research directions.

To assess the improvement that visual-based tracking brings to the tracking framework, the Optimal Subpattern Assignement (OSPA) [22] metric is used. The OSPA errors are shown in Fig. 7 for both in the case where visual tracking is enabled and in case it is disabled. Overall, adding visual tracking improves the result, mostly due to the two targets being consistently tracked. The OSPA metric goes up around 24 and 27 *s* of the experiment due to the visual tracking algorithm keeping two separate tracks for a single target. This is then interpreted as a two distinct targets by the multi-target state tracker. Without visual tracking, the target's appearance flickers as they are not reliably detected solely with the cascade of detectors.

VI. Conclusion

In this paper, we presented a framework for visual tracking and position estimation of UAVs using a single camera. Our framework relies on two computer vision algorithms that extract the position and size of an aircraft from a video frame. The detections are then given to a multi-target tracker to estimate the aircraft's position and velocity in three dimensions. We performed an indoor experiment with three quadrotors to prove the effectiveness of our method.

Future work includes testing the framework outdoors where lighting conditions and cluttered environment might be challenging. We plan to use the presented framework with a collision avoidance algorithm to assess the safety of the methods for a SAA scenario. To improve the tracking capability, the information from the multi-target state tracker could be fed back to the visual tracking to improve the motion model. Finally, it would be beneficial to add the capability to discriminate between different aircraft. This is a necessary step for a fully functional system able to accurately track any aircraft as it relies on the size of a target to estimate distance.

References

[1] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time Onboard Visual-Inertial State Estimation and Self-Calibration of MAVs in Unknown Environments," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 957–964.

[2] M. Achtelik, M. Achtelik, S. Weiss, and R. Y. Siegwart, "Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3056–3063.

[3] J. Zufferey, A. Beyeler, and D. Floreano, "Autonomous flight at low altitude with vision-based collision avoidance and GPS-based path following," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 3329–3334.

[4] S. Yang, S. A. Scherer, K. Schauwecker, and A. Zell, "Autonomous Landing of MAVs on an Arbitrarily Textured Landing Site Using Onboard Monocular Vision," *Journal of Intelligent & Robotic Systems*, pp. 27–43, 2013.

[5] D. Dias, R. Ventura, P. Lima, and A. Martinoli, "On-board vision-based 3D relative localization system for multiple quadrotors," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 1181–1187.

[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] T. Zsedrovits, A. Zarandy, B. Vanek, T. Peni, J. Bokor, and T. Roska, "Visual detection and implementation aspects of a uav see and avoid system," in *IEEE/RSJ International Conference on Circuit Theory and Design*, 2011, pp. 472–475.

[8] L. Mejias, S. McNamara, J. Lai, and J. Ford, "Vision-based detection and tracking of aerial targets for uav collision avoidance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 87–92.

[9] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 511–518.

[10] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *International Conference on Image Processing*, 2002, pp. 900–903.

[11] C. Zhang and Z. Zhang, "A survey of recent advances in face detection," Microsoft Research, Tech. Rep., 2010.

[12] R. Szeliski, *Computer Vision: Algorithms and Applications*. New York, NY, USA: Springer-Verlag New York, Inc., 2010.

[13] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[14] G. Helmut and B. Horst, "On-line boosting and vision," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 260–267.

[15] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 1806–1819, 2011.

[16] S. He, Q. Yang, R. W. Lau, J. Wang, and M.-H. Yang, "Visual tracking via locality sensitive histograms," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2427–2434.

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

[18] 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.

[19] S. Roelofsen, D. Gillet, and A. Martinoli, "Reciprocal collision avoidance for quadrotors using on-board visual detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 4810–4817.

[20] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2006, pp. 6.1–6.10.

[21] H. W. Kuhn and B. Yaw, "The hungarian method for the assignment problem," *Naval Res. Logist. Quart*, pp. 83–97, 1955.

[22] D. Schuhmacher, B.-T. Vo, and B.-N. Vo, "A consistent metric for performance evaluation of multi-object filters," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3447–3457, 2008.