Chasing a moving target from a flying UAV

C. Teuliere, L. Eck, E. Marchand

Chasing a moving target from a flying UAV

Céline Teulière, Laurent Eck, Eric Marchand

***Abstract*— This paper proposes a vision-based algorithm to autonomously track and chase a moving target with a small-size flying UAV. The challenging constraints associated with the UAV flight led us to consider a density-based representation of the object to track. The proposed approach to estimate the target's position, orientation and scale, is built on a robust color-based tracker using a multi-part representation. This object tracker can handle large displacements, occlusions and account for some image noise due to partial loss of wireless video link, thanks to the use of a particle filter. The information obtained from the visual tracker is then used to control the position and yaw angle of the UAV in order to chase the target. A hierarchical control scheme is designed to achieve the tracking task. Experiments on a quad-rotor UAV following a small moving car are provided to validate the proposed approach.**

I. INTRODUCTION

The vision-based control of Unmanned Aerial Vehicles (UAVs) has become a very active field of research in the last decade [21] [1] [20] [6] [11]. Vision indeed provides a cheap, passive and rich source of information, and low-weight cameras can be embedded even on small-size flying UAVs. Until now, most of the efforts have been concentrated on developing vision-based control methods for autonomous take off, landing, stabilization and navigation, in which the visual information is usually obtained using a known model of a target [20] or the environment [22], key images [6], or texture points for motion estimation [21] [16] or optical flow computation [11].

In this paper we consider the specific task of chasing a moving object, in an unknown environment, and without any *a priori* model of the object (see figure 1). The reliability of the visual information is critical for the good realization of the vision-based control task. For autonomously performing such a task, one has to be able to robustly extract the object location from images despite difficult constraints: large displacements, occlusions, image noise, illumination and pose changes or image blurr.

Considerable work has already been done in visual tracking to address the aforementioned challenges, starting with designing an adequate representation for the object to be tracked. A simple and widely used way to describe an object is the image template, which stores luminance or color values, and their locations [9][15]. Describing how the object looks like pixel-wise, image templates can accurately recover a large range of motions. However, they are very sensitive to some modifications in the object appearance due to pose changes, lighting variations, blurr or occlusions.

C. Teulière and L. Eck are with CEA, LIST, Interactive Robotics Unit, 18 route du Panorama, BP6, Fontenay aux Roses, F- 92265 France firstname.name@cea.fr

E. Marchand is with Université de Rennes 1, IRISA, INRIA, Lagadic Project, Rennes, France firstname.name@irisa.fr

This work was realized in the context of the French ANR national project SCUAV (ANR Psirob SCUAV project ref ANR-06-ROBO-0007-02)

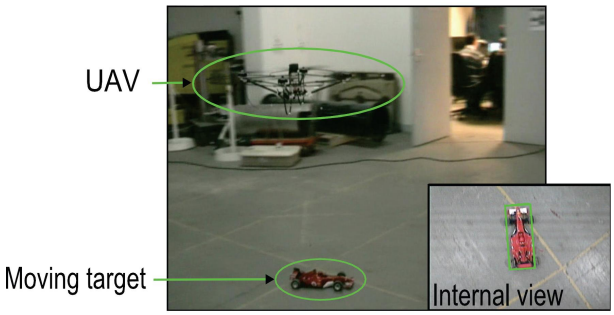

Fig. 1. Quad-rotor UAV tracking a small vehicle and internal view from the embedded camera.

Density-based descriptors such as color histograms represent an attractive alternative for their low computational complexity and robustness to appearance changes [5] [19]. In the challenging UAV application context, strong simplifying assumptions are usually made in the vision algorithms. [2] [24]. In [2] a color-based algorithm is used to track a fixed target and autonomously stabilize a UAV above it. Due to hardware limitations their proposed tracking approach is simplified and assumes that the target is clearly visible, without handling occlusions nor the presence of distractors of similar color.

In this paper, our objective is to provide a full vision-based system, using a color-based tracking method to robustly localize a moving object through frames and control the UAV to chase it. To our knowledge, this has never been done while considering potential loss due to occlusions, and estimating not only the position of the object but its rotation and scale changes in the image, which will allow us to control the UAV's attitude and yaw.

We consider a quad-rotor UAV (see figure 1 and 7) equipped with a camera attached to its airframe, pointing downward, an inertial measurement unit (IMU), and a barometer. Except for the low level embedded attitude control, the computations are deported to a ground station. The data are transmitted between the ground station and the UAV through a radio transmission. Figure 2 gives an overview of the proposed system.

- *The visual tracking system* aims to provide an estimate of the relative position and in-plane rotation between the UAV and the object. To achieve this in a robust way, a color-based representation is chosen and the tracking is performed in the particle filtering framework. The tracking system is presented in section II.

- *The control scheme* is presented in section III. Estimations of the relative position and translational velocity between the UAV and the object, obtained by the vision system, are used as an input to the proposed control law.

Experiments using the quad-rotor UAV are finally presented in section IV.

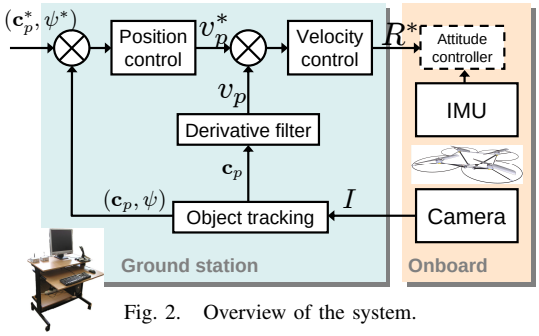

Fig. 2. Overview of the system.

II. Object tracking algorithm

The robust tracking of a moving object from a flying UAV, on long sequences, involves challenging constraints. In particular, the vision-based system has to face: large interframe displacements, due to both the UAV and the target motions, partial or full occlusions of the target (see figure 3-a and b), noisy images due to occasional erroneous transmission (see figure 3-c), background changes (see figure 3-d). The system also requires real-time capabilities.

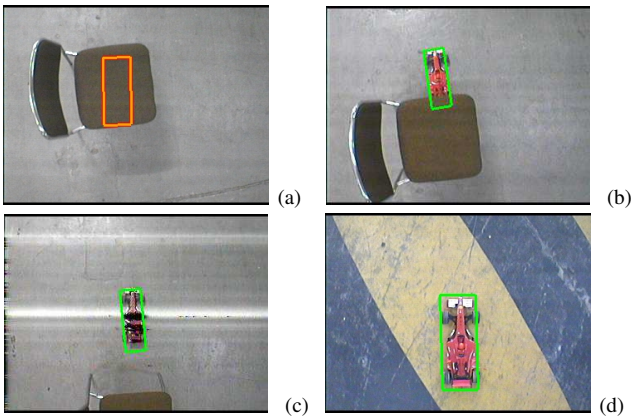

Fig. 3. Examples of critical cases: full occlusion (a) and recovery (b), transmission noise (c) and background changes.

A. Related works in color-based object tracking

Color-based object tracking has been extensively studied, especially since the successful application of the *mean shift* algorithm [5] to the tracking problem. The well-known drawback of choosing color histograms as a representation for the object is the loss of spatial information [10], making difficult to track more complex motions than a simple translation. In the past few years different approaches have been proposed to tackle this issue, the main objective being to add some spatial information on the object to track while keeping the robustness of color histogram descriptors. In kernel-based methods, [10] and [7] proposed a Newton-like framework for using a set of multiple kernels. The spatial configuration between them allows to recover high-dimensional motion parameters. However, the search being deterministic, the algorithm can fail in case of total occlusion, presence of another object of similar color, or large interframe displacements.

Within another framework, [19] [17] propose to build a particle filter using the same kind of color-based similarity criterium to detect the translation and scale changes. The posterior density of probability of the object location is discretized in a set of weighted particles. The use of a probabilistic framework provides better robustness w.r.t. occlusions or presence of similar objects. To improve tracking accuracy, [19] [18] introduce the idea of dividing the object in several parts and to build a multi-part likelihood function, but the motion they estimate is limited to location and scale parameters. [13] proposed a multi-part target representation for a better detection of scale and in-plane rotation parameters. More recently [14] [23] proposed to combine both deterministic and bayesian approaches using different multi-part target configurations. However, loss detection and initialisation or reinitialisation of the trackers are not mentionned.

B. Object representation

Object state: The goal of the tracking algorithm is to estimate the position of the object of interest through frames. The object is represented by a rectangle of fixed aspect ratio $r = \frac{h}{w}$, which state is defined by its position $\mathbf{c} = (x, y)^\top$, orientation ψ and area $A = hw$ (see figure 4). The state $\mathbf{x}_k$ of the object in frame k is then given by $\mathbf{x}_k = (x_k, y_k, \psi_k, \frac{1}{\sqrt{A_k}})^\top$. This state will be used in section III as an input for the UAV control law.

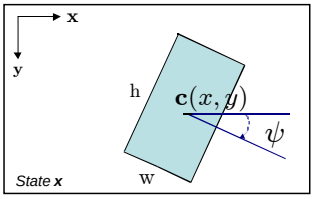

Fig. 4. Parameters of the state and rectangular representation.

Object descriptor: Given the state $\mathbf{x}_k$ of the object in frame k, and $\{\boldsymbol{l}_i\}_{i=0..n_k}$ the pixel locations inside the rectangular boundaries, the color histogram is given by $\mathbf{q}(\mathbf{x}_k) = \{q_u(\mathbf{x}_k)\}_{u=1...m}$, m being the number of bins in which the color space is divided. For each bin u:

$$q_u(\mathbf{x}_k) = \sum_{i=1}^{n_k} K(\boldsymbol{l}_i - \mathbf{c})\delta_u(b(\boldsymbol{l}_i)) \quad (1)$$

where $b(\boldsymbol{l}_i)$ is the bin corresponding to the color of the pixel at location $\boldsymbol{l}_i$, δ designates Kronecker's function, and K is a kernel function centered in $\mathbf{c}$, weighting image locations. In our case, K is the Epanechnikov kernel, which we suppose normalized so that $\sum_{u=1}^{m} q_u = 1$.

Let $\mathbf{q}^* = \{q_u^*\}_{u=1...m}$ denote the reference histogram, determined in the tracking initialisation step. Then, the tracking process aims to find in each frame k, the candidate state $\mathbf{x}_k$ which histogram $\mathbf{q}(\mathbf{x}_k)$ is the "closest" to the reference histogram $\mathbf{q}^*$. To achieve this, a correlation criterion in the histogram space is provided by Bhattacharyya coefficient:

$$\rho(\mathbf{x}_k) = \rho(\mathbf{q}^*, \mathbf{q}(\mathbf{x}_k)) = \sum_{u=1}^{m} \sqrt{q_u^* q_u(\mathbf{x}_k)}. \quad (2)$$

A candidate state $\mathbf{x}_k$ for the object in frame k is then compared to the reference object using the Bhattacharyya distance:

$$d(\mathbf{x}_k) = d(\mathbf{q}^*, \mathbf{q}(\mathbf{x}_k)) = \sqrt{1 - \rho(\mathbf{x}_k)}. \quad (3)$$

Multi-kernel representation: To improve the sensitivity of the object descriptor to different motions we use a generic multi-kernel representation. A set of identical kernels are positionned in the rectangular object.Weighting pixels locations with those kernels thus gives them a different importance according to their position in the object.

Formally, a state $\mathbf{x}_k$ is associated to n_h histograms $\{\mathbf{q}_j(\mathbf{x}_k)\}_{j=1..n_h}$ computed with the kernels $K_j(\boldsymbol{l} - \mathbf{c}_j)$. The distances $d_j(\mathbf{x}_k) = d(\mathbf{q}_j^*, \mathbf{q}_j(\mathbf{x}_k))$ are defined as in (3):

$$d_m(\mathbf{x}_k) = \frac{1}{n_h} \sum_{j=1}^{n_h} d_j(\mathbf{x}_k) \quad (4)$$

C. Tracking process

For the tracking to be robust to short occlusions of the object or noise due to transmission errors between the UAV and the ground station, we use the particle filtering framework. The proposed tracking scheme is based on the well-known CONDENSATION algorithm [12]. The main idea is to represent the probability density function (p.d.f) $p(\mathbf{x}_k \mid \mathbf{z}_{1:k})$ of the state $\mathbf{x}_k$ at frame k, by a finite set $\{(s_k^{(i)}, \pi_k^{(i)})\}_{i=1..N}$ of N samples, or particles, $s_k^{(i)}$ associated with the weights $\pi_k^{(i)}$. Each particle $s_k^{(i)}$ represents a potential state for the object and $\mathbf{z}_{1:k}$ are the observations until frame k. For each new frame, the particles first evoluate according to a given dynamic model. Then, the likelihood of every particle is measured in the image and a weight is derived. The output considered is the weighted mean of the resulting set of particles. The particle set is updated by performing a random weighted draw among the particles. This resampling step promotes the best particles by duplicating them, to avoid degeneracy issues and keep a fair representation of the p.d.f.

Likelihood function: The likelihood function is derived from the distance defined in equation (4). The spatial information is yielded by computing n_h histograms in the n_h parts as in [19] [14] [23]. Formally, the likelihood of a state $\mathbf{x}_k$ is then defined by:

$$p(\mathbf{z}_k \mid \mathbf{x}_k) \propto exp(-\lambda {d_m}^2(\mathbf{x}_k)) \quad (5)$$

where λ is a constant parameter tuned empirically (a typical value is $\lambda = 20$).

For initialisation, the particles are sampled on a Gaussian distribution around the initial known position. The particles are propagated in the evolution step by a simple constant velocity model. The particle filtering output considered is the estimator of probability expectation: $E[\mathbf{x}_k] = \sum_{i=1}^{N} \pi_k^{(i)} s_k^{(i)}$. The algorithm is summarized in figure 5.

Knowing the set of N particles $\{s_{k-1}^{(i)}\}_{i=1..N}$ with equal weights $\frac{1}{N}$ at step $k-1$:

- **Evolution** of the particles according to a constant velocity model, giving a new set of particles: $\{s_k^{'(i)}\}_{i=1..N}$.
- **Update:** Using the observation $\mathbf{z}_k$ the weight of each predicted particle is computed according to (5). $\pi_k^{(i)} \propto p\left(\mathbf{z}_k \mid \mathbf{x}_k = s_k^{'(i)}\right)$, with $\sum_{i=1}^{N} \pi_k^{(i)} = 1$.
- **Resampling** by performing a random weighted draw of N particles from $\left\{(s_k^{'(i)}, \pi_k^{(i)})\right\}_{i=1..N}$, thus giving the new set: $\left\{(s_k^{(i)}, \frac{1}{N})\right\}_{i=1..N}$.

Fig. 5. Color-based tracking with particle filtering.

D. Loss detection

In case of complete occlusions or temporary image losses (see figure 3-a and c) the distance measurement becomes irrelevant and the resampling step of the particule filter promotes the best particules in a wrong way. In such a situation, it is better to rely only on the evolution model. The tracker is considered to be lost when the minimal particle's distance d_{min} (4) is above a given threshold d_{lim}. For now, this threshold is tuned empirically, and refined in the beginning of the sequence, where we assume the object is not yet occluded, by taking $d_{lim} = 1.3 d_{min}$. However it would benefit from an automatic adaptation for sequences with large illumination changes.

When the object is declared to be lost, the particle filter switches to a critical mode with no resampling step, until one particle gets under the threshold and the filter goes back to normal. This process allows the tracker to handle short occlusions or peaks of noise in the images. If the filter fails to recover, a reinitialisation procedure needs to be performed. This is discussed in the following section.

E. Automatic initialisation and reinitialisation

The tracking algorithm presented above assumes that the reference histograms of the object are available. They can be computed by selecting the target in the image in the beginning of the sequence as in [2]. Here, to avoid this manual detection we provide the algorithm with an image of the target in which the model is computed.

Then the algorithm is able to automatically initialise by using a method similar to Camshift detection [4]. First, the reference histogram is used as a look-up table to build the image of probabilities (the "back projection" image) corresponding to the current image. This operation replaces the pixel values of the input image with the value of the corresponding bin of the histogram. Then, the resulting

image represents the probability of each pixel to belong to the object (see figure 6). The histogram considered at this stage is the global histogram of the whole object, without multi-kernel division.

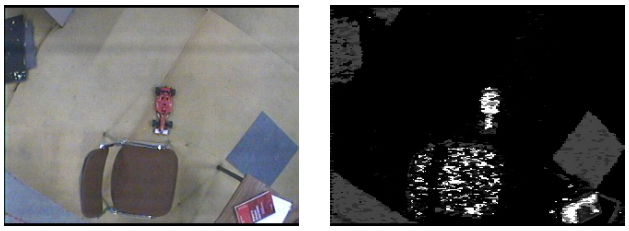

Fig. 6. Example of back projection.

The detection is achieved by running several Camshift searches from different initial windows resulting in several candidate positions for the object. The positions are then tested using the distance criterium (4) to determine which one corresponds to the object of interest.

The same process is used for reinitialisation in case of tracking failure (see section II-D). The reinitialisation can be requested manually by an operator from the ground station or automatically triggered when the tracking system remains lost after a given number of iterations.

III. UAV CONTROL

The tracking system presented in the previous section provides us with an estimate of the position, orientation and size of the object in the image. In this section we present the closed-loop control scheme used for the control the UAV using those estimates as measurements.

A. UAV Modelling

The UAV is represented by a rigid-body of mass m and of tensor of inertia $\mathbf{I} \in \mathbb{R}^{3\times3}$. Let us define the frame $\mathcal{F}_c$ attached to the vehicle in its centre of mass, and assume it coincides with the camera frame (see figure 7). The position of the centre of mass of the vehicle relative to the world frame ${}^w\mathbf{p}_c$ is denoted by $\mathbf{p}$. For simplicity of notation the rotation ${}^w\mathbf{R}_c$ of the body frame $\mathcal{F}_c$ relative to $\mathcal{F}_w = (\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$ is denoted by $\mathbf{R}$. We also define the frame $\mathcal{F}_p$, centered in the centre of mass like $\mathcal{F}_c$, but whose axis are parallel to those of $\mathcal{F}_w$, so that ${}^p\mathbf{R}_c = {}^w\mathbf{R}_c = \mathbf{R}$.

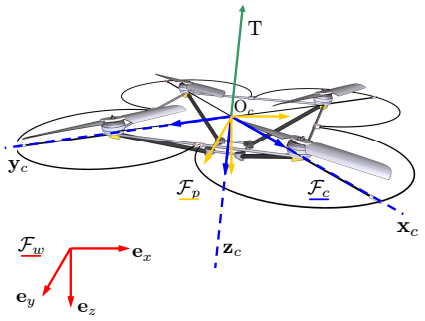

Fig. 7. Frame definitions.

Let $\mathbf{v}$ (respectively $\mathbf{\Omega}$) be the linear (resp. angular) velocity of the center of mass expressed in the world frame $\mathcal{F}_w$ (resp. in $\mathcal{F}_c$). The control inputs to send to the vehicle are: T, a scalar input termed thrust or heave, applied in direction $\mathbf{z}_c$ and $\mathbf{\Gamma} = [\mathbf{\Gamma}_x \mathbf{\Gamma}_y \mathbf{\Gamma}_z]^\top$ the control torques relative to the Euler angles.

Assuming the world frame is Galilean, Newton's equations of motion yield the following:

$$\begin{cases} \dot{\mathbf{p}} = \mathbf{v} \\ m\dot{\mathbf{v}} = T\mathbf{R}\mathbf{e}_z + mg\mathbf{e}_z \\ \dot{\mathbf{R}} = [\mathbf{\Omega}]_\times \\ \mathbf{I}\dot{\mathbf{\Omega}} = -\mathbf{\Omega} \times \mathbf{I}\mathbf{\Omega} + \mathbf{\Gamma} \end{cases} \tag{6}$$

where g is the gravity constant.

The quad-rotor UAV is an underactuated system with 4 inputs. Its translational motion results from the rotations (pitch and roll). In this work we assume that the system's attitude is already controlled onboard with a separate high gain control loop [3]. Therefore, our control scheme acts as a controller sending orientation commands to a low-level controller which is responsible for robust flight.

B. Control scheme

Translational control: in our vision-based control scheme, the translational motion is controlled using the estimates of the location $\mathbf{c} = (x, y)^\top$ of the object in the image plane obtained from the vision system. This location is expressed in the camera frame, that is, if $(X_c, Y_c, Z_c)^\top$ denote the unknown coordinates of the object in $\mathcal{F}_c$, then the estimate $\mathbf{c}$ from the tracker corresponds to $x = \frac{X_c}{Z_c}$ and $y = \frac{Y_c}{Z_c}$. As illustrated in figure 8, the displacement observed in the image plane is partly due to the UAV attitude. Note that it is independant of the distance Z_c. We compensate this displacement using an estimate of the rotation matrix $\mathbf{R}$ computed onboard from the IMU, so that the position that we regulate to zero is the position $\mathbf{c_p} = (x_p, y_p)^\top$ of the object in the projected frame $\mathcal{F}_p$.

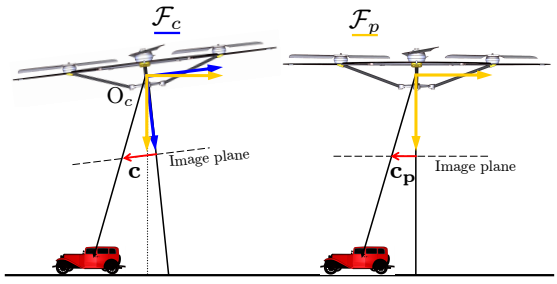

Fig. 8. Attitude compensation: $\mathbf{c}$ is the position of the object that is estimated by the visual tracking algorithm, and $\mathbf{c_p}$ is the corresponding position in the frame $\mathcal{F}_p$ where the rotation has been compensated.

The relative position and velocity errors are defined by:

$$e_{\mathbf{c_p}} = \mathbf{c}^*_\mathbf{p} - \mathbf{c_p} \tag{7}$$

$$e_{\mathbf{v_p}} = \mathbf{v}^*_\mathbf{p} - \mathbf{v_p} \tag{8}$$

where $\mathbf{c}^*_\mathbf{p}$ is the desired position of the object in the image plane. In practice for the chasing task $\mathbf{c}^*_\mathbf{p} = \mathbf{0}$ since we want to see the object in the center of the image. The velocity

$\mathbf{v_p}$ is deduced from the differentiation of the position $\mathbf{c_p}$.

We use a hierarchical control. The inner-loop is a PI controller on the velocity, and the outer-loop a simple proportional control on the position:

$$\mathbf{v_p}^* = -K_p e_{\mathbf{c_p}} \tag{9}$$

The inner-loop on the velocity is required to ensure the stability of the system. It acts as a damping in the UAV control.

Yaw control: the yaw angle ψ is controlled by using a proportional controller:

$$\mathbf{\Omega}_z^* = -K_{p\psi}(\psi^* - \psi) \tag{10}$$

where $K_{p\psi}$ denote the proportional gain. ψ^* is the desired yaw angle, that is the desired orientation of the object in the image plane. It is set to zero in the experiments so that the UAV follows the rotations of the car. The yaw velocity $\mathbf{\Omega}_z$ is controlled onboard using gyrometers measurements.

Altitude: although the vision system also provides an estimate of the size of the object which can be taken as a depth measurement, we found that the velocity obtained by its differentiation was too noisy to be used directly in the altitude control. For the experiments the altitude is controlled onboard using the barometer measurements to remain constant.

IV. Experimental validation

This section presents the experiments conducted on the quad-rotor (X4-flyer) (figure 1). The UAV sends the images from its embedded camera to the ground station (PC) through a wireless analogical link at 2.4GHz. The data is processed on the ground station and the desired orientation and thrust are sent back to the quad-rotor vehicle. Onboard, the exponential stability of the orientation toward the desired one is ensured by a 'high gain' controller (running at 166Hz in the DSP) [8]. On the ground station, the overall system (visual tracking and control computation) runs with a framerate of 20Hz. We tested the overall system on different sequences with a small colored car (figure 1). In all the experiments the RGB color space with $8 \times 8 \times 8$ bins has been used. The particle filter runs with 200 particles.

A. Stabilization task

We first present a stabilization task performed while the target car was static, to validate the good behaviour of the control scheme. Figure 9 shows the resulting position error for the stabilization task: $\mathbf{c_p} = (X_p/Z_p \quad Y_p/Z_p)^\top$.

In this experiment the measured altitude Z_p was less than 2 meters, which means that the UAV was stabilized above the target with a maximal translation error of $10cm$. Figure 9 shows that the maximum orientation error in stabilization is 3deg.

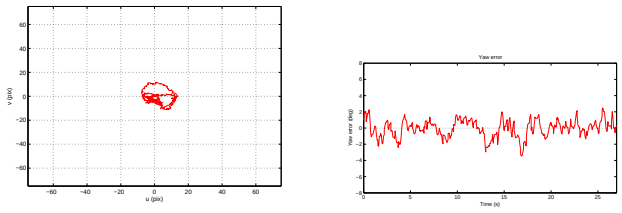

Fig. 9. Trajectory of the center of the car in the image (pixel coordinates) and orientation error for the stabilization task.

Chasing a moving target: In this section we describe a longer sequence (about 10000 frames) in which the car is moving, with some turns and several complete occlusions (see figure 12). Figure 10 shows the trajectory of the car in the image.

Since the car's motion mainly occurs on the y axis, the position error is larger on this axis (up to $30cm$, which remains very low).

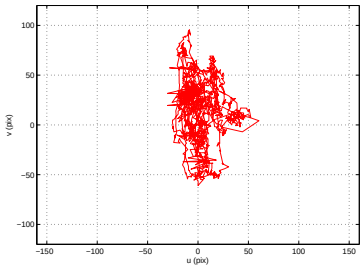

Fig. 10. Trajectory of the center of the car in the image (pixel coordinates) for the chasing task.

Figure 11 illustrates the yaw angle estimation and control with an example of turn.

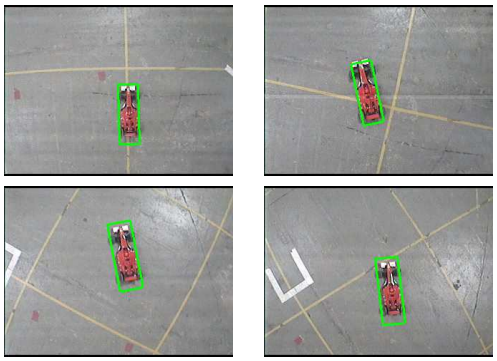

Fig. 11. Example of turn: the tracking system properly estimates the rotation and the UAV follows it to keep the object with the same orientation in the camera frame.

When a loss is detected, the estimate position is drawn in orange (figure 12-b). In the experiments, the occlusions are successfully detected, and the particle filter goes on predicting the car position with the constant velocity model, without resampling. When the car is detected again (the estimated position is drawn in green in the image frame), the particle filter goes back to its normal mode. Since

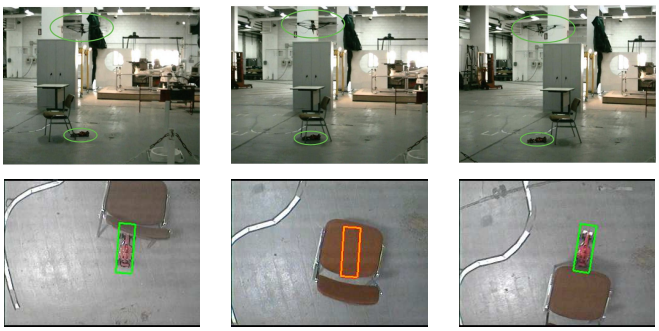

Fig. 12. Example of complete occlusion.

the algorithm assumes that target velocity does not change much while it is occluded, its estimate can be wrong at the end of the occlusion in case of large acceleration or orientation change during the critical mode. However, without the resampling step the particles tend to cover a larger part of the state space and in most of our experiments the tracking algorithm succeeds in finding back the target. We also provide one example of loss after an occlusion where the autonomous reinitialisation procedure was necessary (figure 13). On condition that the target remains in the field of view, the reinitialisation allows the tracker to successfully recover from the loss.

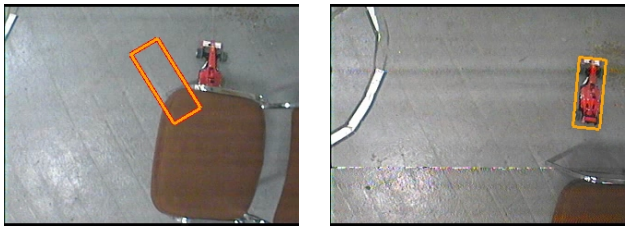

Fig. 13. Example of target loss (a) and reinitialisation (b).

Since the barometer gives an absolute altitude measurement with a high variance (about 1 meter), the altitude control still requires some improvements. Its fusion with the estimate from the vision estimate which provides a more accurate relative information will be tested in future experiments. However, current experiments show that the tracking algorithm can handle scale changes (see figure 14).

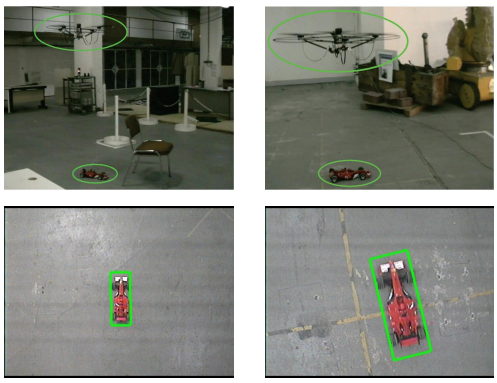

Fig. 14. Altitude changes.

V. CONCLUSIONS

In this paper, we proposed a full vision-based system, using a color-based tracking method to robustly track a moving object through frames and control a quad-rotor UAV to chase it. A multi-part tracker provides an estimate of the relative position and orientation between the UAV and the target in real time. The system has been successfully validated on a sequence with noise, full occlusions, turns and scale changes. The loss detection and reinitialisation system provided is efficient as long as the target remains in the field of view.

REFERENCES

[1] E. Altug, J.P. Ostrowski, and R. Mahony. Control of a quadrotor helicopter using visual feedback. In *IEEE ICRA*, pp. 72-77, 2002.

[2] S. Azrad, F. Kendoul, and K. Nonami. Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm. *J. of System Design and Dynamics*, 4(2):255–268, 2010.

[3] S. Bertrand, T. Hamel, and H. Piet-Lahanier. Stability analysis of an uav controller using singular perturbation theory. In *IFAC World Congress*, pp. 5706–5711, 2008.

[4] G. Bradski. Computer vision face tracking for use in a perceptual user interface. Technical report, Intel Technology, 1998.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE CVPR*, pp. 142–149, 2000.

[6] J. Courbon, Y. Mezouar, N. Guenard, and P. Martinet. Vision-based navigation of unmanned aerial vehicles. *Control Engineering Practice*, 18(7):789 – 799, 2010.

[7] Z. Fan, Y. Wu, and M. Yang. Multiple collaborative kernel tracking. In *IEEE CVPR*, pp. 502 – 509 vol. 2, 20-25 2005.

[8] N. Guenard, T. Hamel, and Mahony. A practical visual servo control for an unmanned aerial vehicle. *IEEE T. on Robotics*, 24(2):331–340, April 2008.

[9] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *PAMI*, 20(10):1025–1039, 1998.

[10] G. Hager, M. Dewan, and C. Stewart. Multiple kernel tracking with ssd. In *IEEE CVPR*, pp. 790–797, 2004.

[11] B. Hérissé, T. Hamel, R. Mahony, and F.-X. Russotto. A terrain-following control approach for a vtol unmanned aerial vehicle using average optical flow. *Autonomous Robots*, pp. 1–19, 2010.

[12] M. Isard and A. Blake. Condensation: conditional density propagation for visual tracking. *IJCV*, 29(1):5–28, 1998.

[13] E. Maggio and A. Cavallaro. Multi-part target representation for color tracking. In *IEEE ICIP*, pp. 729–32, 11-14 2005.

[14] E. Maggio and A. Cavallaro. Accurate appearance-based bayesian tracking for maneuvering targets. *CVIU*, 113(4):544 – 555, 2009.

[15] E. Malis and S. Benhimane. Homography-based 2d visual tracking and servoing. *IJRR*, 26(7):661–676, 2007.

[16] M. Meingast, C. Geyer, and S. Sastry. Vision based terrain recovery for landing unmanned aerial vehicles. *CDC*, pp. 1670 – 1675, 2004.

[17] K. Nummiaro, E. Loller-Meier, and L. Van Gool. An adaptive color-based particle filter. *IVC*, 2003.

[18] K. Okuma, et al. A boosted particle filter: Multitarget detection and tracking. In *ECC*, pp. 28–39, 2004.

[19] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *ECCV*, pp. 661–675, May 2002.

[20] S. Saripalli, J.F. Montgomery, and G.S. Sukhatme. Visually guided landing of an unmanned aerial vehicle. *IEEE T. on Robotics and Automation*, 19(3):371 – 380, june 2003.

[21] O. Shakernia, Y. Ma, T. John Koo, and S. Sastry. Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control. *Asian Journal of Control*, 1:128–145, 1999.

[22] C. Teulière et al. 3d model-based tracking for uav position control. In *IEEE/RSJ IROS'10*, Taipei, Taiwan, October 2010.

[23] C. Teulière, E. Marchand, and L. Eck. A combination of particle filtering and deterministic approaches for multiple kernel tracking. In *IEEE ICRA*, pp. 3948–3954, Kobe, Japan, May 2009.

[24] Y. Watanabe et al. The onera ressac unmanned autonomous helicopter: Visual air-to-ground target tracking in an urban environment. In *American Helicopter Society 66th Annual Forum (AHS 2010)*, 2010.