A Ground-Based Vision System for UAV Tracking

Nuno Pessanha Santos & Victor Lobo
Portuguese Navy Research Center (CINAV)
Portuguese Navy (MGP)
2810-001, Almada, Portugal
nuno.pessanha.santos@marinha.pt, vlobo@novaims.unl.pt

Alexandre Bernardino
Institute for Systems and Robotics (ISR)
Instituto Superior Técnico (IST)
1049-001, Lisboa, Portugal
alex@isr.ist.utl.pt

***Abstract*—It is presented a vision system based on a standard RGB digital camera to track an unmanned aerial vehicle (UAV) during the landing process aboard a ship. The developed vision system is located on the ship's deck and is used to track the UAV during all the landing process. A Ground-based vision system makes it possible to use an UAV with small size and weight, since the UAV will have less computer requirements. The proposed method uses a particle filter (PF) for pose estimation and an unscented Kalman filter (UKF) approach for filtering. This combination provides an adapted filtering framework for tracking. The implemented particle filter is inspired in the evolution strategies present in genetic algorithms with modifications in the mutation and crossover operators to avoid sample impoverishment. Results show that position and angular estimates precision is compatible with the automatic landing system requirements.**

Keywords*—Computer Vision, Model Based Tracking, Autonomous Vehilcles, Military Systems, Particle Filters, Unscented Kalman Filters.

I. INTRODUCTION

The Portuguese exclusive economic zone (EEZ) is the 3rd largest of the European Union and the 10th largest in the world. This represents a total coastline of 1860 km (1000 NMI) long (including Madeira and Azores islands), with approximately 41553 km^2 of territorial waters. In these territorial waters the fast patrol boats (FPB) of the Portuguese Navy have a remarkable share, patrolling and supervising all the maritime traffic to ensure that all the applicable laws are being respected.

The use of UAV's can improve the FPB efficiency, making it possible to obtain real time video streams of aerial images with GPS reference for improved surveillance capacity. In this specific operating environment the UAV should be deployed and recovered onboard two types of FPBs: Argos Class (27 m long, 5.9 m in breadth, draft 2.8 m and displace 97 tons) and Centauro Class (28.4 m long, 5.95 m in breadth, draft 2.8 m and displace 98 tons). For landing in both ships, the stern section is the only possible area, consisting of a very small and irregular area with a size of around 5x6m. The available landing area limits the maximum UAV payload, the used UAV has a 5 kg maximum-take-off weight (MTOW), 180 cm of wingspan and 150 cm of length. In environments where GPS jamming can occur, a vision system can increase the system performance and provide an alternative means of landing [1, 2]. The majority of the research made in this filed is based on UAV on-board sensors and processing units, using markers to obtain the UAV relative position to the platform [1, 3-11]. However, the vision based ground systems [12-20] are not usually considered. Using a computer vision system on the ship's deck circumvents the need to install additional instrument the UAV's. The observation of the UAV from the ship's cameras provides the tracking information necessary to feed the control loop in order to generate a correct landing trajectory. The Portuguese navy has started first tests with low cost commercial off-the-shelf (COTS) UAVs in 2005, testing several hardware and landing configurations [21].

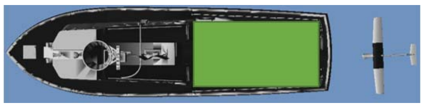
Fig. 1. Landing process illustration.

Using the UAVs 3D CAD model makes it possible to obtain in real time its 3D position and orientation estimation. We propose using a combination of methods for 3D model-based tracking based on a particle filter (PF) and temporal filtering via Unscented Kalman Filter (UKF). The particle filters represent the distribution of an object's 3D pose as a set of weighted hypotheses (also called particles) [22, 23]. Particles are tested by explicitly projecting the object model in the image, with a certain rotation and translation, to compare with the actual image pixel information. Each particle receives a score based on the match to image information and the best particle is filtered using an UKF in order to smooth the pose estimative between frames. The initialization of the filter is a normally a challenge in particle filters, especially when no a-priori or current object pose is known. When this occurs, usually particles are distributed randomly in space around a predefined position [24]. If the predefined position used for initialization is far away from the real target's position, this method is slow to converge, resulting in poor filter robustness and performance. In this paper it is used a simple and efficient initialization method, based on an initial database of known object poses [16-20]. The filter's resampling step is inspired in the evolution strategies present in genetic algorithms [25-28], to avoid sample impoverishment [29]. Adapted crossover and mutation operators are used [16], increasing the filter performance (convergence time) and decreasing the number of particles needed for UAV tracking.

The temporal filtering framework uses the obtained measures over time to generate an estimative that is less affected by noise and uncertainty. For linear systems the standard filtering approach is the classical Kalman Filter (KF). This filter is used normally for noisy data smoothing and parameter estimation [30, 31]. For tracking the UAV's translation and rotation motion over time, the nonlinear relationship between estimated orientation (rotation) prevents the usage of the classical KF. In these cases, several extensions to the KF have been proposed in the literature, most notably the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). In this work we adopt the UKF framework that approximates the Gaussian probability distribution by a set of sample points called sigma points. These points are propagated through the original system equations and used to estimate the posterior state estimate. This results in accurate estimates with low computational cost [32-34].

This paper is organized as follows. In Section 2 it is described the 3D model-based tracking estimation system architecture, which is divided in the: (1) Target Detection, (2) Pose estimation and (3) temporal filtering. In section 3 our system's setup and experimental results are shown. Finally, in section 4 we present the conclusions of the paper and provide directions for further research work.

II. 3D Model-Based Tracking System

This section introduces the proposed 3D model-based pose estimation and temporal filtering methods used in our tracking system (as illustrated in Fig. 2). The UAV detection system operates in an outdoor environment (Fig. 3) by a moving ship. The main goal is to estimate the UAV pose at each time, and send this information to a Ground Control Station (GCS) to be used in the landing control loop.

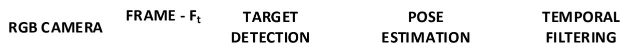

Fig. 2. Simplified system architecture.

Fig. 3. Outdoor environment: clear sky (*left*) and a cloudy sky (*right*).

The proposed method is divided in three parts (Fig. 2):

- **Target Detection** – Consists of searching in the image for regions of interest (ROI). This ROI represents an image region that may contain an object classified as the UAV to land;
- **Pose estimation** – Consists of particle initialization (using a pre-trained database of the UAV in multiple poses), particle evaluation (rank each particle by likelihood) and a local optimization (particles are resampled and optimized to best fit the object appearance in the image);
- **Temporal filtering** – The best particle obtained in the pose estimation stage is filtered between frames using an UKF framework, to obtained a smoothed pose estimate over time.

A. Target Detection

The initial target detection is very important since we are operating in an outdoor environment and the presence of other objects (e.g. birds or clouds) affect the system's performance. It is very important also to achieve illumination invariance, ensuring a high object detection rate.

The initial UAV detection is made using a trained local binary pattern (LBP) cascade classifier [35]. After a training stage, where the classification error is minimized in a labeled dataset the classifier can be used to predict whether observed image regions contain the objects of interest. [36, 37]. To train the classifier a large dataset of both positive (Fig. 4) and negative samples (Fig. 5) are needed.

Fig. 4. Positive training samples (*dataset example*).

Fig. 5. Negative training samples (*dataset example*).

Several other methods were tested such as template matching [38, 39] and other classifier features (Haar-like features [40] and histogram of oriented gradients [41]). The most successful results were obtained with LBP features, resulting in the best scores in ROI detection (Fig. 6).

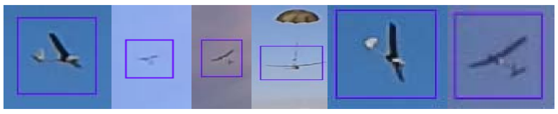

Fig. 6. Obtained ROI using a cascade classifier (*LBP*).

B. Pose estimation

The pose estimation stage is divided in three parts:

- **Particle initialization** – An interest point detection algorithm is applied to the detected ROI, and it is computed an oriented bounding box (BB) that contains the obtained corner points. Then the obtained BB is compared with a pre-trained database of UAV BB in multiple poses. All the poses in the database receive a match score, and all the poses with a sufficient score will generate pose hypotheses (particles) for the next stage;
- **Particle evaluation** – The particles generated in the previous stage will be ranked by likelihood using two different metrics according to the target distance;
- **Local optimization** – Based on the likelihood of the particles, resampling and optimization steps are made to best fit the image appearance.

1) Particle Initialization

The target detection phase results in an oriented BB that indicates the image position and rough posture of the UAV. In the particle initialization stage the obtained BB characteristics are compared with a database of synthetically generated bounding boxes of the UAV in multiple poses (Fig. 7). This database was created offline and indexed in an efficient way for real-time retrieval. Since we use a perspective camera model, the database can be made independent of object position in the image. The detected bounding boxes are obtained applying the FAST [35, 42] feature detector on the observation frame, selecting the key points that are inside the obtained ROI.

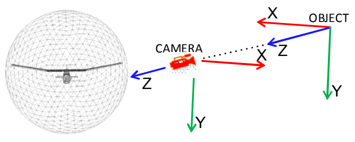

Fig. 7. Virtual sphere representation used for the database (*left*) and the camera and object referential relationship (*right*).

The database is created by generating multiple UAV synthetic images in different poses, and obtaining the BB that best fits the projected object. . This information is stored in a database indexed by angle (θ) and aspect ratio (AR). The difference between the angle and the AR of the observation and database is calculated online using the *Euclidean* distance. We assume that the object's points are all in the same depth (Z coordinate), projected in a plane parallel to the image. The Z coordinate can be computed by the relationship between the BB areas and depth.

$$Z = Z_{Database}\sqrt{\frac{Area_{Database}}{Area_{Observation}}} \quad (1)$$

$$X = \frac{Z(BB_X - C_X)}{f_X} \quad (2)$$

$$Y = \frac{Z(BB_Y - C_Y)}{f_Y} \quad (3)$$

where f and C represents the camera intrinsic parameters – focal length and camera centre coordinates (obtained by previous camera calibration). A correct camera calibration step is essential to ensure precision in system performance.

2) *Particle evaluation*

The quality of each particle is evaluated accordingly to two different likelihood metrics. To decrease the estimation error (due to an outdoor environment estimation) two different likelihood functions were used [16-20]: Texture likelihood (above 25 meters estimation) and a Hybrid approach (under 25 meters estimation). For the color histogram-based likelihood (Fig. 9) the histograms are obtained in the RGB colour space (b_{max} = 12 bins), and the distance between them are calculated using the *Bhattacharyya* similarity metric as in [16, 43, 44]:

$$L_{texture} = 1 - \sum_{b=1}^{b_{max}} \sqrt{h^{inner}(b).\,h^{outer}(b)} \quad (4)$$

where h^{inner} is the inner histogram, h^{outer} is the outer histogram and b is the respective histogram bin. The hybrid likelihood function combines the likelihood function described before with contour information. The edge line segments of the contour are identified and a sample point (v) in the middle is generated. Then a 1D perpendicular search (as seen in Fig. 10) is made to match the sample points with the nearest edge (m). After calculating the matches the contour likelihood is calculated as in [16, 45]:

$$L_{contour} = exp^{\left(-\lambda_v \frac{(p_v - p_m)}{p_v}\right)} \times exp^{(-\lambda_e \bar{e})} \quad (5)$$

where $\bar{e}$ is the arithmetic average distance between the sample points and the edge points, λ_v and λ_e are sensitivity terms used to tune the contour likelihood function, p_v is the number of visible sample points and p_m is the number of matched sample points. The magnitude and orientation (Sobel filter) is calculated and stored for each obtained frame. To decrease the obtained error in the matching process, for each sample point the gradient's angle is compared to the matched point and if this diverges by 45 degrees the matched point is excluded.

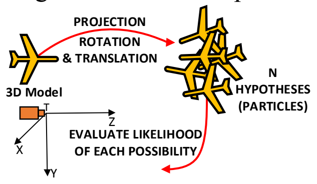

Fig. 8. Particle filter illustration.

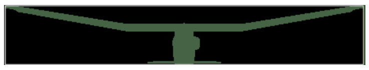

Fig. 9. Example of object inner (*dark green*) and outer (*black*) histograms.

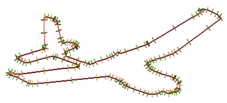

Fig. 10. Sample points and 1D search (*green and orange lines*).

The hybrid likelihood is created by the combination of the two likelihood functions described before in equations (4) and (5), and is obtained by:

$$L_{total} = L_{contour} + A \times L_{texture} \quad (6)$$

where A is a relative sensitivity term used to fine tune the hybrid likelihood function.

3) *Particle Initialization*

Given a set of particles and a likelihood function, the local optimization process operates in three phases [16-20]: **Bootstrap**, **Coarse Optimization** and **Fine Optimization**.

In the **Bootstrap phase** the best 100 possibilities obtained by comparison with the database are evaluated and the obtained weight is stored. The best 3 particles are stored in an auxiliary buffer (Top M ⇒ Top 3). The particles with weight very close to zero (below δ = 0.01) are eliminated and replaced with a random particle selected from the Top 3 buffer, added with Gaussian noise of covariance Σ_B. In each step all particles are evaluated and compared to those in the Top 3. If the obtained weight is higher the vector is updated. If there are at least two particles in the Top 3 with likelihood bigger than "Threshold min", the coarse optimization phase begins. Otherwise, each particle is perturbed with Gaussian noise with covariance Σ_B (If after 10 of these improvement steps no two particles are above "Threshold min", the process is restarted up to a maximum of 3 restarts). The best three particles have an important role in the optimization process because they will provide the "chromosomes" for an approach inspired on genetic algorithms [13, 14]. After extensive experimentation we have found such an approach much more effective than using only random perturbations and resampling as in conventional particle filters.

In the **coarse optimization** phase each particle is analysed, if the particle is a best one, it is perturbed with some Gaussian noise. If the particle weight is smaller, the Top 3 are combined with a random selection of attributes of this particles – crossover. Half of those particles will also be affected by a soft mutation adding Gaussian noise to the result – mutation. These operators allows particle diversity, avoiding possible local minima.

The **fine optimization** starts stops when at least two of the three particles in the Top 3 are above a value of "Threshold" (If this does not happen in 10 iterations the filter returns to the previous phase automatically). The fine optimization phase and the coarse phase are similar but the Gaussian noise variance applied in the mutation is lower, obtaining a final fine-tuning in the pose estimation (This phase ends after 5 iterations).

C. Temporal filtering

We consider that the UAV follows a constant velocity model, i.e. it suffers small accelerations between two consecutive time steps, t and $t+1$. Furthermore we consider that linear and angular motions are independent, thus the full model is composed of two separate dynamical systems. This is a rough approximation of the real dynamics of the aircraft since its orientation and linear velocity are usually highly correlated. The coupling between these dimensions will be subject of future work.

1) Translational Model

Given the stated assumptions the linear motion state vector is defined as:

$$\boldsymbol{x}^l = [\boldsymbol{p}^T \ \boldsymbol{v}^T]^T \tag{7}$$

where $\boldsymbol{p} = \left[p_x \, p_y \, p_z\right]^T$ and $\boldsymbol{v} = \left[v_x \, v_y \, v_z\right]^T$ are respectively the linear position and velocities of the vehicle. The time evolution of the state is:

$$\boldsymbol{x}_{t+1}^l = \boldsymbol{F}^l(\boldsymbol{x}_t^l, \boldsymbol{\xi}_t^l) = \boldsymbol{A}^l \boldsymbol{x}_t^l + \boldsymbol{\xi}_t^l \tag{8}$$

where $\boldsymbol{\xi}_t^l$ is a Gaussian noise random variable with zero mean and covariance $\boldsymbol{Q}_t^l$. $\boldsymbol{A}^l$ is the linear dynamics matrix:

$$\boldsymbol{A}^l = \begin{bmatrix} \mathbf{I}_{3\times3} & \Delta t \cdot \mathbf{I}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \tag{9}$$

The 3D model-based tracking algorithm described in the previous section measures the pose but not the velocity. The observation model is thus the following:

$$\mathbf{z}_t^l = \mathbf{C}^l \boldsymbol{x}_t^l + \eta_t^l \tag{10}$$

where $\boldsymbol{\eta}_t^l$ is a Gaussian noise random variable with zero mean and covariance matrix $\boldsymbol{R}_t$ and $\mathbf{C}^l = [\mathbf{I}_{3\times3} \ \mathbf{O}_{3\times3}]$.

2) Rotational Model

The angular velocity is represented by $\boldsymbol{\omega} = \left[\, \omega_x \, \omega_y \, \omega_z\right]^T$. Its time evolution is modelled as:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{F}^{\omega}(\boldsymbol{\omega}_t, \boldsymbol{\xi}_t^{\omega}) = \boldsymbol{\omega}_t + \boldsymbol{\xi}_t^{\omega} \tag{11}$$

where $\boldsymbol{\xi}_t^{\omega}$ is a zero mean Gaussian noise vector with covariance matrix $\boldsymbol{Q}_t^{\omega}$.

To represent absolute orientation we use a unit quaternion $\boldsymbol{q} = [\boldsymbol{\varrho} \ q_4]^T$, with $\boldsymbol{\varrho} = [q_1 \, q_2 \, q_3]^T = \hat{\boldsymbol{e}} \sin(\vartheta/2)$ and $q_4 = \cos(\vartheta/2)$, where $\hat{\boldsymbol{e}}$ is the axis rotation and ϑ is the angle of rotation. The time evolution of the orientation can be written as:

$$\boldsymbol{q}_{t+1} = \boldsymbol{q}_t \otimes \delta \boldsymbol{q}_t^{\omega} \otimes \delta \boldsymbol{q}_t^{r} \tag{12}$$

where $\otimes$ represents quaternion multiplication (composition of orientations) and $\delta\boldsymbol{q}_t^{\omega}$ and $\delta\boldsymbol{q}_t^{r}$ are quaternions representing the integration of the effect of the angular velocity and rotation noise, assumed constant during a sampling interval Δt:

$$\delta\boldsymbol{q}_t^{\omega} = Q(\boldsymbol{\omega}_t) \qquad \delta\boldsymbol{q}_t^{r} = Q(\xi_t^r) \tag{13}$$

with

$$Q(x) = \left[\frac{x}{|x|}\sin\left(\frac{|x|\Delta t}{2}\right) \quad \cos\left(\frac{|x|\Delta t}{2}\right)\right] \tag{14}$$

Thus the effect of the noise vector ξ_t^r is considered as a random angular velocity disturbance and has a 3×3 covariance matrix $\boldsymbol{Q}_t^r$.

Because quaternions are not minimal representation of orientation, it is not straightforward to represent the state covariance. To address this issue we adopt the formulation of [46] and represent the orientation dynamics in terms of error-quaternions with respect to the current state. A local error quaternion is defined as $\boldsymbol{s} = [\delta\boldsymbol{\varrho}^T \ \delta q_4]$ but a minimal representation is adopted using a vector of generalized Rodrigues parameters:

$$\boldsymbol{r} = R(\boldsymbol{s}) = f\frac{\delta\boldsymbol{\varrho}}{a+\delta q_4} \tag{15}$$

where $a = 1$ and f is a scale factor. We choose $f = 2(a+1)$ so that $\|\boldsymbol{r}\| = \vartheta$ for small angles [46]. The inverse transformation from $\boldsymbol{r}$ to $\boldsymbol{s}$, denoted $R^{-1}(\boldsymbol{r})$ is given by:

$$\delta q_4 = \frac{-a\|\boldsymbol{r}\|^2 + f\sqrt{f^2 + (1-a^2)\|\boldsymbol{r}\|^2}}{f^2 + \|\boldsymbol{r}\|^2} \tag{16}$$

$$\delta\boldsymbol{\varrho} = f^{-1}(a + \delta q_4)\boldsymbol{r} \tag{17}$$

With this parameterization, the incremental rotation dynamics is given by:

$$\boldsymbol{r}_{t+1} = \boldsymbol{F}^r(\boldsymbol{r}_t, \boldsymbol{q}_t, \boldsymbol{\omega}_t, \xi_t^r) \tag{18}$$

with

$$\boldsymbol{F}^r(\boldsymbol{r}, \boldsymbol{q}, \boldsymbol{\omega}, \boldsymbol{\xi}) = R(\boldsymbol{q} \otimes R^{-1}(\boldsymbol{r}) \otimes \boldsymbol{Q}(\boldsymbol{\omega}) \otimes \boldsymbol{Q}(\boldsymbol{\xi})) \tag{19}$$

Finally the state vector for the angular dynamics is defined as:

$$\boldsymbol{x}^a = [\boldsymbol{r} \ \boldsymbol{\omega}] \tag{20}$$

with covariance:

$$\boldsymbol{Q}_t^a = \begin{bmatrix} \boldsymbol{Q}_t^r & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \boldsymbol{Q}_t^{\omega} \end{bmatrix} \tag{21}$$

At each time step the error vector $\boldsymbol{r}_t$ is reset to zero. After its update, $\boldsymbol{r}_{t+1}$ is accumulated to the absolute orientation quaternion $\boldsymbol{q}_{t+1} = \boldsymbol{q}_t \otimes R^{-1}(\boldsymbol{r}_{t+1})$.

3) The Unscented Kalman Filter

To filter along time the measurements obtained by our system, we will use a discrete-time Unscented Kalman Filter (UKF) [32, 46-49]. We adopt the formulation of [32, 46, 49].

The process model1 is represented as:

1 For notational simplicity we consider the time invariant case, i.e. process and measurement models do not depend on time.

$$\boldsymbol{x}_{t+1} = F(\boldsymbol{x}_t, \boldsymbol{\xi}_t) \quad (22)$$

where $\boldsymbol{\xi}_t$ is a Gaussian noise random variable with zero mean and covariance $\boldsymbol{Q}_t$.

As described in the previous section, for the translational motion we have $\boldsymbol{F} = \boldsymbol{F}^l$ and $\boldsymbol{Q}_t = \boldsymbol{Q}_t^l$. For the angular motion we have $\boldsymbol{F} = [\boldsymbol{F}^r\ \boldsymbol{F}^\omega]$ and $\boldsymbol{Q}_t = \boldsymbol{Q}_t^a$ (11), (18) and (21).

The measurement model is represented as:

$$\boldsymbol{z}_{t+1} = \boldsymbol{H}(\boldsymbol{x}_{t+1}, \boldsymbol{\eta}_{t+1}) \quad (23)$$

where $\boldsymbol{\eta}_t$ is a Gaussian noise random variable with zero mean and covariance $\boldsymbol{R}_t$. Function $\boldsymbol{H}$ relates the measurement $\boldsymbol{z}_{t+1}$ to the values of the state vector $\boldsymbol{x}_{t+1}$ and describes the influence of a random variable $\boldsymbol{\eta}_{t+1}$ on the measured value. For the translational motion we have $\boldsymbol{z}_{t+1} = \mathbf{p}_{t+1} + \boldsymbol{\eta}_{t+1}$ and $\boldsymbol{R}_t = \boldsymbol{R}_t^l$ (10). For the angular motion, our observation is the orientation error quaternion, encoded by incremental Rodrigues parameters $\boldsymbol{z}_{t+1} = R(\boldsymbol{q}_{t+1}^m \otimes \delta\boldsymbol{q}_t^\eta \otimes \delta\boldsymbol{q}_t^{-1})$ where $\boldsymbol{q}_{t+1}$ is given by our pose estimation algorithm and $\boldsymbol{q}_t$ is the absolute orientation from the previous time step.

The standard Kalman Filter can be applied when both the measurement and the observation models are linear and the noises are additive:

$$\boldsymbol{x}_{t+1} = \boldsymbol{F}\boldsymbol{x}_t + \boldsymbol{\xi}_t \quad (24)$$

$$\boldsymbol{z}_t = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{\eta}_t \quad (25)$$

where $\boldsymbol{F}$ and $\boldsymbol{H}$ are matrices of appropriate dimensions.

Like the standard Kalman Filter, the UKF is composed of several phases:

- Propagation of the current state $\boldsymbol{x}_t$ and its covariance $\boldsymbol{P}_t$, using the process model (8) and the process noise covariance $\boldsymbol{Q}_t$, to compute the next time *a priori* state $\overline{\boldsymbol{x}}_{t+1}$ and covariance, $\boldsymbol{P}_{t+1}^{xx}$. In the case of standard Kalman Filter we have $\overline{\boldsymbol{X}}_{t+1} = \boldsymbol{F}\boldsymbol{X}_t$ and $\boldsymbol{P}_{t+1}^{xx} = \boldsymbol{F}\boldsymbol{P}_t\boldsymbol{F}^T + \boldsymbol{Q}_t$;
- Prediction of the measurement $\overline{\boldsymbol{z}}_{t+1}$ and associated covariance $\boldsymbol{P}_{t+1}^{zz}$ using the measurement model (10). In the case of standard Kalman Filter we have $\boldsymbol{P}_{t+1}^{zz} = \boldsymbol{H}\boldsymbol{P}_{t+1}^{xx}\boldsymbol{H}^T$;
- Computation of the cross correlation matrix $\boldsymbol{P}_{t+1}^{xz}$ relating the noise in the predicted state vector to the noise in the predicted measurements. In the case of standard Kalman Filter we have $\boldsymbol{P}_{t+1}^{xz} = \boldsymbol{P}_{t+1}^{xx}\boldsymbol{H}^T$;
- Computations of the innovation $\boldsymbol{v}_{t+1}$ as the error between the current measurement and the associated covariance. In the case of the standard Kalman Filter we have $\boldsymbol{v}_{t+1} = \boldsymbol{z}_{t+1} - \overline{\boldsymbol{z}}_{t+1}$ and $\boldsymbol{P}_{t+1}^{vv} = \boldsymbol{P}_{t+1}^{zz} + \boldsymbol{R}_{t+1}$;
- Computation of the Kalman gain $\boldsymbol{K}_{t+1} = \boldsymbol{P}_{t+1}^{xz}(\boldsymbol{P}_{t+1}^{vv})^{-1}$. This step is equivalent in the KF and UKF;
- Update of the *a posteriori* state and covariance estimates. In the case f the standard Kalman Filter we have $\boldsymbol{x}_{t+1} = \overline{\boldsymbol{x}}_{t+1} + \boldsymbol{K}_{t+1}\boldsymbol{v}_{t+1}$ and $\boldsymbol{P}_{t+1} = \boldsymbol{P}_{t+1}^{xx} - \boldsymbol{K}_{t+1}\boldsymbol{P}_{t+1}^{vv}\boldsymbol{K}_{t+1}^T$.

Differently from the KF, the UKF computes the mean and the covariance matrices using a sampling approach. The Gaussian distributions of the state and process noise are used to generate a set of points (sigma points) that are sufficient to represent their statistics. Then these points are propagated through the process and measurement models to obtain new sets of sigma points from which the means and covariance matrices of the state and measurement predictions are obtained, assuming Gaussian approximations.

a) Sigma Points

The process noise covariance $\mathbf{Q}_t$ (n dimensions vector) and state covariance $\mathbf{P}_t$ ($n\ x\ n$ matrix) are transformed into a $2n$ set of points $\boldsymbol{\delta}\mathbf{x}_t(i)$ that represent perturbations to the current state:

$$\boldsymbol{\delta}\mathbf{x}_t(i) = columns\left(\pm\sqrt{2n \cdot \gamma \cdot (\boldsymbol{P}_t + \boldsymbol{Q}_t)}\right), i = 1 \ldots 2n \quad (26)$$

The parameter γ is a scaling parameter given by:

$$\gamma = \alpha^2(n + k) \quad (27)$$

where α is a positive real ($0 \leq \alpha \leq 1$) parameter that controls the high order effects resulting from the existing non-linearity, k is another real parameter ($k \geq 0$) that will control the distance between the sigma points and their average [47, 50]. The matrix $\boldsymbol{P}_t$ is symmetric and positive definite, so it is possible to use the *Cholesky* decomposition to compute $\gamma(\boldsymbol{P}_t + \boldsymbol{Q}_t)$ [51]. The computation of the sigma points, is now done by adding directly $\boldsymbol{\delta}\mathbf{x}_t(i)$ to the state vector $\boldsymbol{x}_t$:

$$\boldsymbol{\mathcal{X}}_i = \boldsymbol{x}_t + \boldsymbol{\delta}\mathbf{x}_t(i), i = 1 \ldots 2n \text{ and } \boldsymbol{\mathcal{X}}_0 = \boldsymbol{x}_t \quad (28)$$

b) A priori state mean and covariance

The process model F is then applied to the obtained sigma points, generating the transformed sigma points:

$$\boldsymbol{\mathcal{Y}}_i = F(\boldsymbol{\mathcal{X}}_i, 0),\ i = 0 \ldots 2n \quad (29)$$

No additional noise is added at this step because the noise was already added at the sigma point's creation step (28).

The *a priory* state estimate is obtained calculating the mean of the transformed sigma points $\boldsymbol{\mathcal{Y}}_i$ according to:

$$\overline{\boldsymbol{x}}_{t+1} = \sum_{i=0}^{2n} w_i \boldsymbol{\mathcal{Y}}_i \quad (30)$$

The weights w_i come from the unscented transformation [52-54] and are given by ($i = 1, \ldots, 2n$):

$$w_0 = \frac{\lambda}{(n+\lambda)} \qquad w_i = \frac{\lambda}{2(n+\lambda)} \quad (31)$$

To estimate the *a priori state* covariance each propagated sigma point is removed from its mean to create the set of error vectors:

$$\delta\overline{\boldsymbol{x}}_{t+1}(i) = \boldsymbol{\mathcal{Y}}_i - \overline{\boldsymbol{x}}_{t+1} \quad (32)$$

Then:

$$\boldsymbol{P}_{t+1}^{xx} = \sum_{i=0}^{2n} w_i\, \delta\overline{\boldsymbol{x}}_{t+1}(i)\delta\overline{\boldsymbol{x}}_{t+1}(i)^T \quad (33)$$

where the scaling weights are given by (18), with exception of w_0 alternatively given by:

$$w_0 = \frac{\lambda}{(n+\lambda)} + (1 - \alpha^2 + \beta) \quad (34)$$

β is a non-negative term which incorporates knowledge of the higher order moments of the distribution [47, 50].

c) A priori measurement mean and covariance

The transformed sigma points are now projected into the measurement space according to:

$$\mathcal{Z}_i = H(\boldsymbol{\mathcal{Y}}_i, 0) \tag{35}$$

The measurement expected value is computed as:

$$\bar{\mathbf{z}}_{t+1} = \sum_{i=1}^{2n} w_i \boldsymbol{\mathcal{Z}}_i \tag{36}$$

and the measurement covariance estimative $\boldsymbol{P}_{t+1}^{zz}$ is given by:

$$\boldsymbol{P}_{t+1}^{zz} = \sum_{i=0}^{2n} w_i [\boldsymbol{\mathcal{Z}}_i - \bar{\mathbf{z}}_{t+1}][\boldsymbol{\mathcal{Z}}_i - \bar{\mathbf{z}}_{t+1}]^{\mathrm{T}} \tag{37}$$

d) The Innovation

The innovation vector $\boldsymbol{\upsilon}_{t+1}$ is obtained comparing the actual measurement $\mathbf{z}_{t+1}$ to the measurement estimate $\bar{\mathbf{z}}_{t+1}$:

$$\boldsymbol{\upsilon}_{t+1} = \mathbf{z}_t - \bar{\mathbf{z}}_{t+1} \tag{38}$$

The innovation covariance $\boldsymbol{P}_{t+1}^{\upsilon\upsilon}$ is obtained adding the measurement noise $\boldsymbol{R}_{t+1}$ to the measurement covariance $\boldsymbol{P}_{t+1}^{zz}$:

$$\boldsymbol{P}_{t+1}^{\upsilon\upsilon} = \boldsymbol{P}_{t+1}^{zz} + \boldsymbol{R}_{t+1} \tag{39}$$

e) The Kalman Gain

The cross correlation matrix $\boldsymbol{P}_{t+1}^{xz}$ is obtained from $\boldsymbol{\mathcal{Z}}_i$ and $\boldsymbol{\mathcal{Y}}_i$, according to:

$$\boldsymbol{P}_{t+1}^{xz} = \sum_{i=0}^{2n} w_i [\boldsymbol{\mathcal{Y}}_i - \bar{\mathbf{x}}_{t+1}][\mathcal{Z}_i - \bar{\mathbf{z}}_{t+1}]^{\mathrm{T}} \tag{40}$$

The Kalman gain is then computed from:

$$\boldsymbol{K}_{t+1} = \boldsymbol{P}_{t+1}^{xz} (\boldsymbol{P}_{t+1}^{\upsilon\upsilon})^{-1} \tag{41}$$

f) State Update

Finally the *a posteriori* state estimate is obtained according to:

$$\mathbf{x}_{t+1} = \bar{\mathbf{x}}_{t+1} + \boldsymbol{K}_{t+1}\boldsymbol{\upsilon}_{t+1} \tag{42}$$

and the state covariance $\boldsymbol{P}_{t+1}$ is given by:

$$\boldsymbol{P}_{t+1} = \boldsymbol{P}_{t+1}^{xx} - \boldsymbol{K}_{t+1}\boldsymbol{P}_{t+1}^{\upsilon\upsilon}\boldsymbol{K}_{t+1}^{T} \tag{43}$$

III. Experimental Results

In this section we show results from the implemented framework. Because with real images we do not have ground truth information, results are qualitatively evaluated using synthetically generated (Fig. 11) landing sequences with ground truth. The method was implemented in C++ on a 2.40 GHz Intel i7 CPU and NVIDIA GeForce GT 750M. All results presented in this section refer to this platform.

Fig. 11. Synthetic frame example.

A. Target Detection, Pose Estimation and Temporal filtering

Temporal filtering is an essential step, allowing the use of the object behaviour information between frames. A constant linear and angular velocity model is used to filter the individual pose estimate at each frame and decrease the obtained estimation error. For performance illustration two different video sequences were analysed, quantifying the obtained errors between the ground truth, the measurements and the estimated values of pose, applying the detection, pose estimation and temporal filtering phases. The parameters used in the target detection and pose estimation phase have been reported in [16, 17, 19, 20].

The UAV is initialized in a centered position (as seen in Fig. 11), according to the values described in Table I. The orientation is represented here in Euler angles (Rotation in the X, Y and Z axis ⇒ $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ respectively) due to the use of the OpenGL library for object rendering, which uses this rotation representation.

TABLE I. Initial UAV Position Attitude

Initial State Vector	
X	0 m
Y	0 m
Z	25 m
Alfa (α)	0°
Beta (β)	0°
Gama (γ)	190°

1) Landing Sequence I

In the first sequence the UAV makes an approach at a constant speed of 9.45 m/s in the Z axis (after several real sea tests this is the ideal UAV approach velocity for a successful landing), maintaining a constant attitude. As it is possible to analyse from the Fig. 12, Fig. 13 and Fig. 14 (X, Y and Z motion respectively) there is some measure error smoothing on the UAV trajectory estimation through the use of the temporal filtering framework. This filtering framework allows a better pose estimation than the direct use of the taken measurement. The obtained errors in this landing sequence are quantified in Table II.

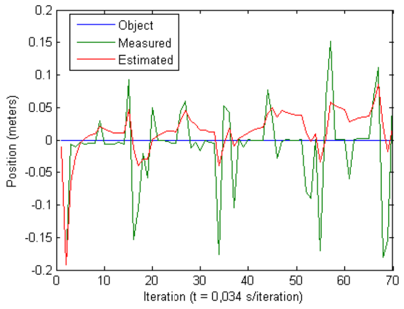

Fig. 12. Temporal filtering – Sequence I: Translation in X.

The attitude error is obtained via the error quaternion between the measurement/estimation and the ground truth. The error between the two rotations is given by [32, 49]:

$$\theta_{error} = 2\arccos(q_{error_w}) \tag{44}$$

where $\boldsymbol{q}_{error_w}$ corresponds to the scalar part of the obtained error quaternion. As observed in Fig. 15, the filter is effective in smoothing out many angular outlier observations on the pose estimation method. These outliers arise from the observation process due to the ambiguities in appearance of the aircraft around symmetries (frontal/backward, up/down).

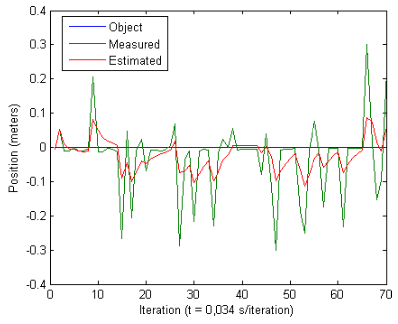

Fig. 13. Temporal filtering – Sequence I: Translation in Y.

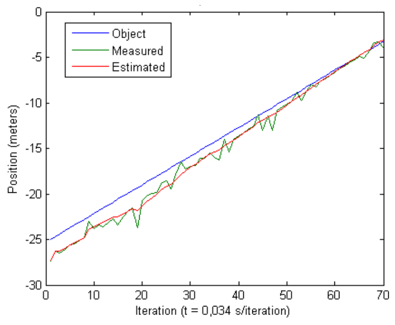

Fig. 14. Temporal filtering – Sequence I: Translation in Z.

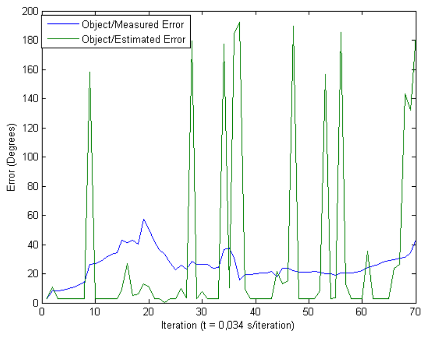

Fig. 15. Temporal filtering – Sequence I: Rotation errors between the ground truth and the measurements (*green*) and between the ground truth and the estimated values (*blue*).

TABLE II. ERROR QUANTIFICATION – SEQUENCE I

	Measured		**Sequence I (Estimative)**	
	Mean Value	***Median value***	***Mean Value***	***Median value***
X (m):	0.012	0.002	-0.012	-0.013
Y (m):	0.029	0.007	0.023	0.019
Z (m):	32.39	1.069	1.099	1.096
Attitude error Obj. vs Meas. (degrees)	32.39	2.89		
Attitude error Obj. vs Esti. (degrees)			24.87	23.91

2) *Landing Sequence II*

For a better analysis of the temporal filtering advantages, a new landing sequence is generated with random pose variations, representing large disturbances (e.g. wind) according to two normal distributions. one for the translation X and Y ($\mathcal{N}(0, 0.25)$ m) and other for the Euler angles alpha, beta and gamma ($\mathcal{N}(0,5.00)$ deg). The approach velocity (Z axis) remains the same of the previous sequence.

The resulting position estimates (Fig. 16, Fig. 17 and Fig. 18) present an obvious smoothing in the UAV trajectory far more evident in this example. In the attitude analysis (Fig. 19) there is a clear decrease in the error from the measurements to the filtered estimates, despite the high number of pose outliers.. The obtained errors in this sequence are quantified in Table III.

In this work we have not introduced coupling between the dynamics in the translation and rotational models yet because the dynamics of the UAV is currently being more thoroughly analysed. These constraints would further improve the dynamics model and thus the precision of the estimates. Also the observation model could benefit from the state covariance estimates to weight the sampling process. These points will be subject to further research.

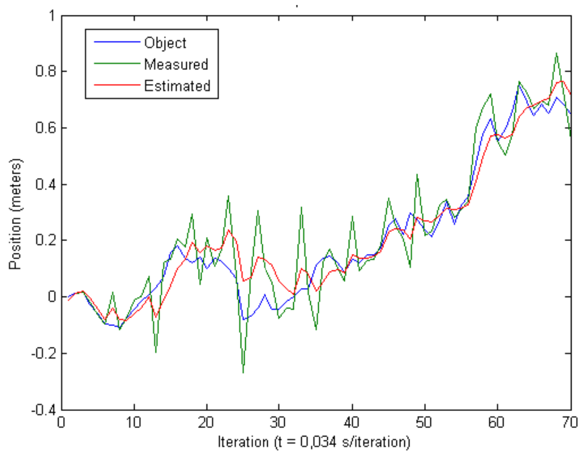

Fig. 16. Temporal filtering – Sequence II: Translation in X.

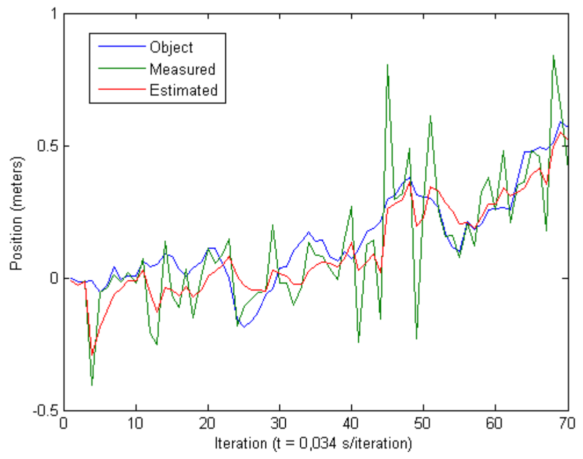

Fig. 17. Temporal filtering – Sequence II: Translation in Y.

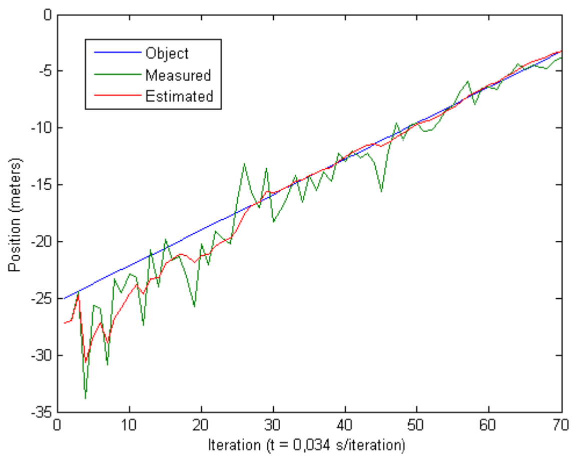

Fig. 18. Temporal filtering – Sequence II: Translation in Z.

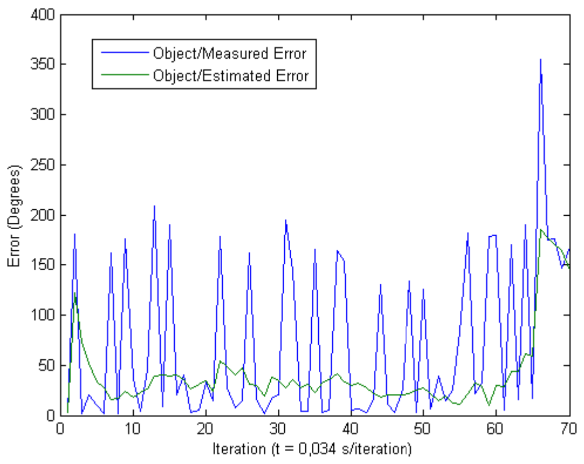

Fig. 19. Temporal filtering – Sequence II: Rotation errors between the ground truth and the measurements (*green*) and between the ground truth and the estimated values (*blue*).

TABLE III. ERROR QUANTIFICATION – SEQUENCE II

	Measured		**Sequence II (Estimative)**	
	Mean Value	***Median value***	***Mean Value***	***Median value***
X (m):	-0.021	-0.009	-0.011	-0.009
Y (m):	0.026	0.018	0.037	0.034
Z (m):	1.015	0.621	0.904	0.212
Attitude error Obj. vs Meas. (degrees)	72.79	23.30		
Attitude error Obj. vs Esti. (degrees)			31.94	29.54

CONCLUSIONS

A method was introduced and tested for tracking of a known UAV, for the purpose of automatic landing. The presented algorithms features a UAV detection method based on a cascaded classifier; a particle initialization methodology with a pre-trained database; a particle optimization stage inspired in evolutionary methods that has shown interesting convergence properties; and a temporal filtering using an unscented Kalman filtering. The implemented architecture allows better pose estimation between successive frames decreasing the obtained position and attitude errors. We consider the achieved precision levels suitable for automated landing. In the following stages of the work, we will focus now real data acquisition and on implementing the methods using a GPU for processing time decrease.

ACKNOWLEDGMENT

This work was funded by the EU Commission within the National Strategic Reference Framework (QREN) under grant agreement 23260 (AUTOLAND), by the Portuguese government through FEDER funds under project SEAGULL (SI IDT 34063) and by the FCT project [UID/EEA/50009/2013].

REFERENCES

[1] A. D. Wu, E. N. Johnson, M. Kaess, F. Dellaert, and G. Chowdhary, "Autonomous flight in GPS-denied environments using monocular vision and inertial sensors," *AIAA J. of Aerospace Information Systems (JAIS),* vol. 10, p. 14, April 2013.

[2] A. Grant, P. Williams, N. Ward, and S. Basker, "GPS jamming and the impact on maritime navigation," *Journal of Navigation,* vol. 62, pp. 173-187, 2009.

[3] C. S. Sharp, O. Shakernia, and S. S. Sastry, "A vision system for landing an unmanned aerial vehicle," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, 2001, pp. 1720-1727.

[4] S. Saripalli, J. F. Montgomery, and G. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in *Robotics and automation, 2002. Proceedings. ICRA'02. IEEE international conference on*, 2002, pp. 2799-2804.

[5] Y. J. Zhao and H. L. Pei, "Improved Vision-Based Algorithm for Unmanned Aerial Vehicles Autonomous Landing," *Applied Mechanics and Materials,* vol. 273, pp. 560-565, 2013.

[6] S. Saripalli, "Vision-based autonomous landing of an helicopter on a moving target," in *Proceedings of AIAA Guidance, Navigation, and Control Conference, Chicago, USA*, 2009.

[7] W. Xiang, Y. Cao, and Z. Wang, "Automatic take-off and landing of a quad-rotor flying robot," in *Control and Decision Conference (CCDC), 2012 24th Chinese*, 2012, pp. 1251-1255.

[8] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle," *Journal of intelligent & robotic systems,* vol. 61, pp. 221-238, 2011.

[9] S. Lange, N. Sünderhauf, and P. Protzel, "Autonomous landing for a multirotor UAV using vision," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2008)*, 2008, pp. 482-491.

[10] A. Cesetti, E. Frontoni, A. Mancini, P. Zingaretti, and S. Longhi, "A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks," *Journal of Intelligent and Robotic Systems,* vol. 57, pp. 233-257, 2010/01/01 2010.

[11] G. Xu, Y. Zhang, S. Ji, Y. Cheng, and Y. Tian, "Research on computer vision-based for UAV autonomous landing on a ship," *Pattern Recognition Letters,* vol. 30, pp. 600-605, 4/15/ 2009.

[12] W. Kong, D. Zhang, X. Wang, Z. Xian, and J. Zhang, "Autonomous landing of an UAV with a ground-based actuated infrared stereo vision system," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 2963-2970.

[13] C. Martínez, P. Campoy, I. Mondragón, and M. A. Olivares-Méndez, "Trinocular ground system to control UAVs," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009, pp. 3361-3367.

[14] A. Hazeldene, A. Sloan, C. Wilkin, and A. Price, "In-flight orientation, object identification and landing support for an unmanned air vehicle," in *Proceedings of the IEEE International Conference on Autonomous Robots and Agents*, 2004, pp. 13-15.

[15] R. Moore, S. Thurrowgood, D. Bland, D. Soccol, and M. V. Srinivasan, "A stereo vision system for UAV guidance," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009, pp. 3386-3391.

[16] N. P. Santos, F. Melício, V. Lobo, and A. Bernardino, "A Ground-Based Vision System for UAV Pose Estimation," *International Journal of Mechatronics and Robotics (IJMR) - UNSYSdigital International Journals,* vol. 1, p. 7, 2014.

[17] N. P. Santos, F. Melício, V. Lobo, and A. Bernardino, "A Ground-Based Vision System for UAV Pose Estimation," presented at the The 10th International Conference on Intelligent Unmanned Systems, Montreal, Quebec, Canada, 2014.

[18] N. P. Santos, F. Melício, V. Lobo, and A. Bernardino, "A Ground-based vision system for UAV autonomous landing," presented at the Poster presented at the 3rd Workshop on European Unmanned Maritime Systems, Porto, Portugal, 2014.

[19] N. P. Santos, "Sistema de Visão para Aterragem Automática de UAV," presented at the Jornadas do Mar 2014 - "Mar: Uma onda de progresso", Escola Naval, Alfeite, Almada, 2014.

[20] N. P. Santos, "Sistema de visão para aterragem automática de UAV," MSc, Instituto Superior de Engenharia de Lisboa (ISEL), 2014.

[21] A. M. Gonçalves-Coelho, L. C. Veloso, and V. J. A. S. Lobo, "Tests of a light UAV for naval surveillance," presented at the IEEE/OES Oceans'2007, Aberdeen, UK, 2007.

[22] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*: Springer, 2001.

[23] A. J. Haug, *Bayesian Estimation and Tracking: A Practical Guide*: Wiley, 2012.

[24] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*: Pearson Education, Limited, 2011.

[25] M. Boli, P. M. Djuri, and S. Hong, "Resampling algorithms for particle filters: a computational complexity perspective," *EURASIP J. Appl. Signal Process.,* vol. 2004, pp. 2267-2277, 2004.

[26] S. Park, J. P. Hwang, E. Kim, and H.-J. Kang, "A new evolutionary particle filter for the prevention of sample impoverishment," *Trans. Evol. Comp,* vol. 13, pp. 801-809, 2009.

[27] N. M. Kwok, G. Fang, and W. Zhou, "Evolutionary particle filter: re-sampling from the genetic algorithm perspective," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, 2005, pp. 2935-2940.

[28] A. Carmi, S. J. Godsill, and F. Septier, "Evolutionary MCMC particle filtering for target cluster tracking," in *Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop, 2009. DSP/SPE 2009. IEEE 13th*, 2009, pp. 262-267.

[29] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*: Wiley, 2006.

[30] R. Faragher, "Understanding the basis of the Kalman filter via a simple and intuitive derivation," *IEEE Signal Processing Magazine,* vol. 29, pp. 128-132, 2012.

[31] B. D. Anderson and J. B. Moore, *Optimal filtering*: Courier Dover Publications, 2012.

[32] E. Kraft, "A quaternion-based unscented Kalman filter for orientation tracking," in *Proceedings of the Sixth International Conference of Information Fusion*, 2003, pp. 47-54.

[33] F. L. Lewis, L. Xie, and D. Popa, *Optimal and robust estimation: with an introduction to stochastic control theory* vol. 26: CRC, 2008.

[34] F. Janabi-Sharifi and M. Marey, "A kalman-filter-based method for pose estimation in visual servoing," *Robotics, IEEE Transactions on,* vol. 26, pp. 939-947, 2010.

[35] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *Computer Vision – ECCV 2006*. vol. 3951, A. Leonardis, H. Bischof, and A. Pinz, Eds., ed: Springer Berlin Heidelberg, 2006, pp. 430-443.

[36] G. M. Marakas, *Decision support systems in the 21st century* vol. 134: Prentice Hall, 2003.

[37] E. Turban, R. Sharda, D. Delen, and T. Efraim, *Decision support and business intelligence systems*: Pearson Education India, 2007.

[38] V. Lepetit and P. Fua, "Monocular Model-Based 3D Tracking of Rigid Objects: A Survey," *Foundations and Trends® in Computer Graphics and Vision,* vol. 1, pp. 1-89, 2005.

[39] K. S. Dhindsa and G. Babbar, "Development of an Improved Feature based Algorithm for Image Matching," *Development,* vol. 14, 2011.

[40] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, 2002, pp. I-900-I-903 vol. 1.

[41] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, pp. 886-893.

[42] E. Rosten, R. Porter, and T. Drummond, "Faster and Better: A Machine Learning Approach to Corner Detection," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 32, pp. 105-119, 2010.

[43] M. Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino, and P. Lima, "Tracking objects with generic calibrated sensors: an algorithm based on color and 3D shape features," *Robotics and autonomous systems,* vol. 58, pp. 784-795, 2010.

[44] M. Taiana, J. C. Nascimento, J. A. Gaspar, and A. Bernardino, "Sample-Based 3D Tracking of Colored Objects: A Flexible Architecture," in *BMVC*, 2008, pp. 1-10.

[45] C. Choi and H. I. Christensen, "Robust 3D visual tracking using particle filtering on the SE (3) group," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 4384-4390.

[46] J. L. Crassidis and F. L. Markley, "Unscented filtering for spacecraft attitude estimation," *Journal of guidance, control, and dynamics,* vol. 26, pp. 536-542, 2003.

[47] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan, "The unscented particle filter," in *NIPS*, 2000, pp. 584-590.

[48] J. Zhou, Y. Yang, J. Zhang, and E. Edwan, "Applying quaternion-based unscented particle filter on INS/GPS with field experiments," *Proceedings of the ION GNSS, Portland,* pp. 1-14, 2011.

[49] Y.-J. Cheon and J.-H. Kim, "Unscented filtering in a unit quaternion space for spacecraft attitude estimation," in *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, 2007, pp. 66-71.

[50] A. Doucet, N. De Freitas, and N. Gordon, "An introduction to sequential Monte Carlo methods," in *Sequential Monte Carlo methods in practice*, ed: Springer, 2001, pp. 3-14.

[51] N. J. Higham, "Analysis of the Cholesky decomposition of a semi-definite matrix," 1990.

[52] Y. Rui and Y. Chen, "Better proposal distributions: Object tracking using unscented particle filter," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, pp. II-786-II-793 vol. 2.

[53] P. Li, T. Zhang, and A. E. Pece, "Visual contour tracking based on particle filters," *Image and Vision Computing,* vol. 21, pp. 111-123, 2003.

[54] S. J. Julier, "The scaled unscented transformation," in *American Control Conference, 2002. Proceedings of the 2002*, 2002, pp. 4555-4559.