A Ground-Based Vision System for UAV Pose Estimation

Nuno Pessanha Santos [1], Fernando Melício [2], Victor Lobo [3] and Alexandre Bernardino [4]

***Abstract*—**We present a vision system based on a single frame of standard RGB digital camera to estimate the pose of an unmanned aerial vehicle (UAV). The envisaged application is of ground-based automatic landing, where the vision system is located on the ship's deck and is used to estimate the UAV pose (3D position and orientation) during the landing process. Using a vision system located on the ship makes it possible to use an UAV with less processing power, decreasing its size and weight. The proposed method uses a 3D model based pose estimation approach that requires the 3D CAD model of the UAV. Pose is estimated in a particle filtering framework. The implemented particle filter is inspired in the evolution strategies present in the genetic algorithms avoiding sample impoverishment. Results show position and angular errors are compatible with automatic landing system requirements, even without temporal filtering. The algorithm is suitable for real time implementation in standard workstations with graphical processing units.

***Keywords*—** Computer Vision, Model Based Pose Estimation, Autonomous Vehicles, Military Systems, Particle Filters.

I. Introduction

In the past several years, research on Autonomous vehicles has augmented the number of possible civilian and military applications. The key requirement for most of these systems, especially in military environment, is reliability, guaranteeing a very low failure rate in normal operation.

Portugal has the 10th largest exclusive economic zone (EEZ) in the world and this makes it difficult to control the territorial waters (approximately 41335 km2) with limited resources. Nowadays, fast patrol boats (FPB) have an important role in this context, but their efficacy can be significantly improved by the support of unmanned aerial vehicles (UAVs). Because this kind of ships has a small crew and UAV pilots are not usually available, the automation of all the UAV operations that still require human intervention (e.g. landing) is a priority. The available landing site in this kind of ships is a small and irregular area with size of around 5x6m (stern section). A vision based system is also more robust in environments where GPS jamming can occur, increasing the system performance in real operation scenarios [1].

The available landing area limits the maximum UAV payload. Most of the research made in this area are based on on-board systems [2, 3], whereas ground systems [4, 5] are not usually considered. Using a system located on the ship's deck makes it possible to use an UAV with less processing power, decreasing its size and weight. The Portuguese navy has started first tests with low cost commercial off-the-shelf UAVs in 2005, testing several hardware and landing configurations [6].

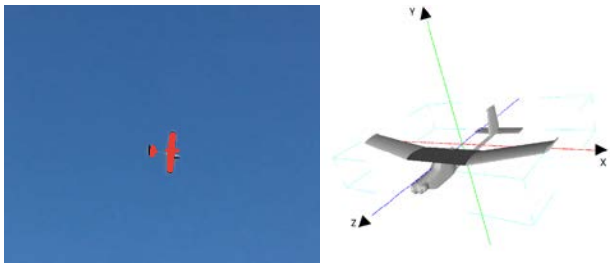

Figure 1. Pose estimation in a real scenario (left) and the used UAV Model (right).

For the vision based system we adopt a 3D model based approach (as seen in Figure 1). Nowadays all UAVs have their 3D CAD model available, so their pose (3D position and orientation) can be estimated using a class of methods for 3D model-based tracking based on particle filters. These represent the distribution of an object's 3D pose as a set of weighted hypotheses (particles) [7, 8]. Hypotheses are tested by explicitly projecting the object model in the image, with a certain rotation and translation, and comparing the actual image pixel information. The particle filter, in contrast with other methods like Kalman Filter, can be used in non-linear situations for instance in rotation estimation and multimodal distributions typical of pose estimation methods [9-11].

Initialization is a common problem in particle filters when no a-priori information of object location is known. In this case the particles are usually scattered randomly in space around a specific position [11]. This method is usually slow to converge, affecting the filter performance and robustness. Another solution may be to increase the particle number, but this increases the required processing power, which is usually limited. In this paper we propose a method to make a simple and efficient initialization, based on an initial database of known object poses.

The resampling strategies for the created particles are inspired in the evolution strategies present in the genetic algorithms [12-14], avoiding sample impoverishment [15]. Crossover and mutation operators are adopted, increasing the filter performance and decreasing the number of needed particles for object tracking.

In Section 2 is described the 3D model-based pose estimation system architecture, which is divided in the following major components: (1) Airship detection, (2) Particle initialization, (3) Particle evaluation and (4) Pose

[1]Nuno Pessanha Santos is researcher at the Portuguese Navy research center and MSc student at Instituto Superior de Engenharia de Lisboa (e-mail: nuno.pessanha.santos@marinha.pt).

[2]Fernando Melício is coordinator teacher at Instituto Superior de Engenharia de Lisboa (e-mail: fmelicio@deea.isel.ipl.pt).

[3]Victor Lobo is associated Professor at the Portuguese Naval Academy (e-mail: vlobo@isegi.unl.pt).

[4]Alexandre Bernardino is a tenured assistant Professor at the Instituto Superior Técnico, Lisboa (e-mail: alex@isr.ist.utl.pt).

optimization. In section 3 some experiments concerning synthetic and real scenarios are presented. Finally, in section 4 we present the conclusions of the paper and provide directions for further research work.

II. 3D Model-Based Pose Estimation System

This section introduces the proposed 3D model based pose estimation system. In this study, we consider UAV detection on an outdoor environment by a moving platform (ship). The state vector $\boldsymbol{P_t}$ is defined by the 3D position (X, Y, Z), and the object orientation represented by X, Y, Z Euler angles (γ, β, α). The main goal is to estimate the pose of the vehicle at each time $\{\boldsymbol{P_t}; \boldsymbol{t} \boldsymbol{\epsilon} \mathbf{N}\}$ and send this information to a control station. In this paper we do not consider information fusion between different time steps (tracking) but solely the performance evaluation of the detection and pose estimation algorithms. Tracking will be addressed in future work.

The proposed method is divided in 4 parts:

- **Airship detection** – In this stage, the image is scanned in the search for regions that may contain aerial vehicles with the appearance of the UAV to land;
- **Particle initialization** – In this stage we try to match the regions found in the previous stage to a pre-trained database of UAV bounding boxes in multiple poses. All poses with sufficient match score will initialize particles for the pose estimation procedure;
- **Particle evaluation** – Each of the particles generated in the previous stage will be ranked by likelihood, according to the distance of the particle two different metrics are used;
- **Pose optimization** – Based on the likelihood metrics, particles are resamples and optimized to best fit the image appearance.

A. Airship Detection

The initial region of interest (ROI) detection is very important since we will use feature points from the object to make the particle initialization. Since we are operating in outdoor environments it is very important to achieve luminosity invariance, guaranteeing a high object detection rate.

The initial UAV detection is made using a trained local binary pattern cascade classifier [16],. This initial image segmentation is very important since we are operating in the exterior and the presence of other objects in the frame (for instance clouds) affects the performance of the proposed system architecture.

B. Particle Initialization

The airship detection stage results in an oriented bounding box (BB) that indicates the image position and rough posture of the UAV. To initialize particles close to the real posture of the UAV, we compare the detected bounding box characteristics with synthetically generated bounding boxes of the UAV in many possible poses. This database can be created offline and indexed in an efficient way for fast run-time access. Since we use a perspective camera model, the database (training stage) can be made independent of object position in the image.

The detected bounding boxes are obtained applying the FAST [17, 18] feature detector on the observation frame, selecting the key points that are inside the obtained ROI. This ROI is simply obtained in the airship detection phase, making it possible to select the feature points belonging to the UAV.

The database is created by rendering images of the UAV 3D CAD model at a fixed position but varying rotation according to a Gaussian distribution with $\mathcal{N}(0,90)$ for the γ, β and α parameters. The created database represents 20000 orientation possibilities: 13620 possibilities (68.1%) are contained in the frontal semi sphere and the rest (6380) are contained in the back semi sphere. This gives an average sample difference of 1.23º for the front possibilities and 1.8º for the back semi sphere possibilities. We favour the front hemisphere because in a landing situation the UAV pose is more likely located on the frontal semi sphere.

For each generated possibility is obtained the BB that better fits the projected object and is stored in a database indexed by angle (θ) and aspect ratio (AR). In order to estimate the initial object pose of the UAV detected in the real image (3D position and orientation), we compare the bounding box angle and BB aspect ratio between the object in the observation frame and the database.

$$AR = \frac{BB_{Widht}}{BB_{Height}} \tag{1}$$

The difference between the observation (obs) and the database (data) is calculated online using the Euclidean distance as in:

$$d(\theta, AR) = \sqrt{(\theta_{obs} - \theta_{data})^2) + (AR_{obs} - AR_{data})^2} \tag{2}$$

For the best possibilities 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.

$$Z = Z_{Database}\sqrt{\frac{Area_{Database}}{Area_{Observation}}} \tag{3}$$

The X and Y coordinates are calculated by the relation between the coordinate of the centre of the observation BB, the obtained Z coordinate and the camera intrinsic parameters – focal length and camera centre coordinates (obtained by calibration).

$$X = \frac{Z(BB_X - C_X)}{f_x} \tag{4}$$

$$Y = \frac{Z(BB_Y - C_Y)}{f_y} \tag{5}$$

It is important to guarantee a correct camera calibration to ensure precision in system performance. Figure 2 shows a block diagram of the particle initialization procedure.

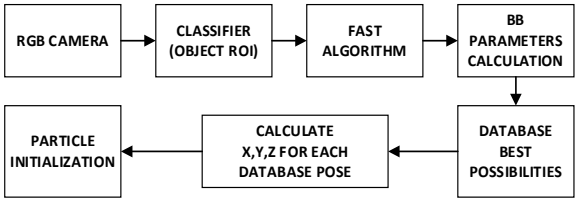

Figure 2. Particle initialization architecture.

C. Particle Evaluation

Each particle corresponds to a 3D pose. By projecting the 3D CAD model of a particle in the image frame, we obtain an expectation of its location (hypothesis). To check the quality of this hypothesis, we will sample the current image at the expected location and evaluate a likelihood function. To decrease the estimation error two different likelihood functions were used:

- Difference between the inner (object) and the outer histogram limited by a bounding box (above 25 meters estimation);
- Hybrid approach combining the likelihood function described before with a contour based method (under 25 meters estimation).

Above 25 meters the particle with the highest weight is the particle that maximizes the difference between the two histograms (See figure 3). Using this approach is possible to have a likelihood function invariant to illumination changes. This robustness is very important since this is an outdoor application, where the brightness variations are often unpredictable.

The histograms are obtained in the RGB colour space (12 bins for each colour – B = 12), and the distance between them are calculated using the Bhattacharyya similarity metric as in [19, 20]:

$$L_{texture} = 1 - \sum_{b=1}^{B} \sqrt{h^{inner}(b).h^{outer}(b)} \quad (6)$$

Where h^{inner} is the inner histogram, h^{outer} is the outer histogram and b is the respective histogram bin.

Figure 3. Example of object inner (object) and outer (between the object and the bounding box) regions where color histograms are computed for the particle likelihood metric.

The hybrid likelihood function (used in under 25 meters estimation) combines the likelihood function described before, with contour information. The set of visible edges (3D CAD model) are projected in a 2D plane according to the currently tested pose hypothesis. The edge line segments are identified and a sample point (v) in the middle is generated. Then a 1D perpendicular search (as seen in Figure 4) is made to match the sample points with the nearest edge (m). After calculating the matches the contour likelihood is calculated as in [21]:

$$L_{contour} = exp^{\left(-\lambda_v \frac{(p_v - p_m)}{p_v}\right)} \times exp^{(-\lambda_e \overline{e})} \quad (7)$$

where $\overline{e}$ is the arithmetic average distance between the sample points and the edge points, the 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.

To increase system performance when a frame is obtained the magnitude and orientation of the gradient (Sobel filter) is calculated and stored.

In order to obtain less error in the matching process, for each sample point the angle is calculated and compared to the gradient orientation at the matched point. If this angle diverges by 45 degrees the matched point is excluded minimizing the existing error.

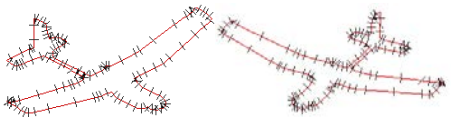

Figure 4. Sample points and 1D search (black lines).

The hybrid likelihood is created by the junction of the two likelihood functions described before in equations (6) and (7), and is calculated as in:

$$L_{total} = L_{contour} + A \times L_{texture} \quad (8)$$

Where A is a relative sensitivity term used to fine tune the hybrid likelihood function. The simplified schematic of this calculation can be seen in Figure 5.

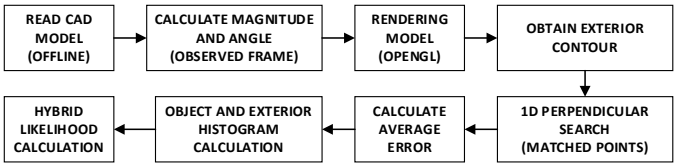

Figure 5. Hybrid likelihood calculation – simplified schematic.

D. Pose Optimization

Given a set of particles and a likelihood function, the optimization process (Figure 6) operates in three phases:

- Bootstrap;
- Coarse Optimization;
- Fine Optimization.

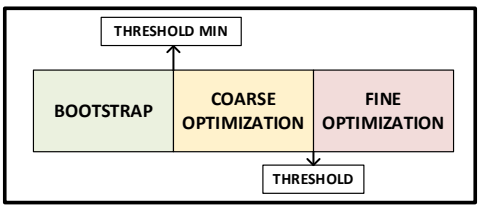

Figure 6. Particle Filter optimization Phases.

In the Bootstrap phase the best 100 possibilities obtained by comparison with the database are collected in a list (Top 100). The likelihood of each particle is evaluated and stored in the list. The best M particles are stored in an auxiliary buffer (Top M). The particles with weight very close to zero (below $\delta = 0.01$) are eliminated and replaced with a random particle selected from the Top M buffer, added with Gaussian noise of covariance Σ_B. At this point, all particles have likelihood above δ.

Then, we run up to 10 improvement steps. In each step all particles are evaluated and compared to those in the Top M, if the obtained weight is higher the Top M is updated. If there are at least two particles in the Top M with likelihood bigger than "Threshold min", the bootstrap phase ends. 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 bootstrap process is restarted up to a maximum of 3 restarts. In our experiments we have noticed that the occurrence of restarts is very rare.

The coarse optimization phase begins when at least two particles have a weight higher than “Threshold min”. At any stage of the coarse and fine optimization phases, the best two 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. The approach works as follows.

Each particle in the Top 100 list coming from the bootstrap process is analyzed. If the particle is the best one, it is perturbed with some Gaussian noise. If the particle weight is smaller, the best 2 particles are combined by crossover to create a new particle. The crossover operation consists in random selection of attributes (X, Y, Z, γ, β, α) of the original particles. To half of the particles generated by crossover is applied a soft mutation by adding Gaussian noise to the result. Together these rules allow a focused particle diversity, simultaneously converging to the best solution and avoiding possible local minima. The process stops when at least two of the particles are above value “Threshold”. If this does not happen in 10 iterations, the pose optimization filter returns to the bootstrap phase automatically.

The fine optimization phase is analogous to the coarse phase but the Gaussian noise variance applied in mutation is lower, in order to make a fine-tuning to the estimated pose. The fine optimization phase ends after 5 iterations.

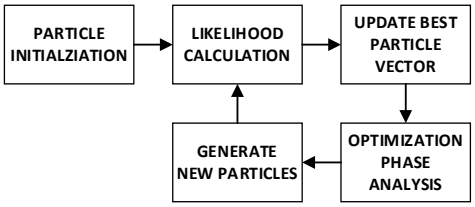

Figure 7. Pose optimization architecture.

III. Experimental Results

In this section we show results from UAV detection, particle initialization and pose estimation on real images. Because with real images we do not have ground truth information, results are qualitatively evaluated through observation of the CAD model projection on the images. To quantitatively evaluate the performance of our method we also show a statistical analysis of pose estimation on a large number of synthetically generated images with ground truth. The method was implemented in C++ on a 2.40 GHz Intel i7 CPU and NVIDIA GeForce GT 750M. All computational times and results presented in this section refer to this platform.

A. Object Detection

The initial object detection is made using a cascade classifier that identifies the object localization ROI as seen in Figure 8. The average running time in a 1280x720 frame is about 59 ms.

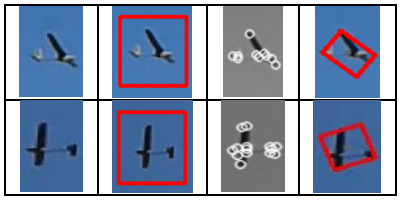

Figure 8. Observation frame Bounding Box detection. From left to right: (i) original image; (ii) detected ROI by sliding window cascaded classifier trained on LBP features; (iii) FAST features detected inside previous ROI; and (iv) oriented bounding box enclosing detected features.

After the ROI was obtained, the FAST algorithm is applied to the frame and the points that are inside that ROI are used to calculate the object BB. The obtained average running time of the FAST algorithm in a 1280x720 frame is less than 1 ms.

B. Particle Initialization

Figure 9. In red we can see the projection on the real image of randomly selected possibilities for particle initialization from the best database matches.

For the obtained bounding box the angle and aspect ratio is calculated and compared with the database, as explained in Section II.D. Since the database is entirely loaded in the beginning of the program, the average running time for all database is less than 1 ms.

The best possibilities (as seen in Figure 9) are used to initialize the pose optimization procedure. There are some possibilities with the same relation of BB and angle that do not correspond to the real pose of the UAV, but those possibilities are filtered in the first iteration. This filtering is done replacing all the particles with very low likelihood with a best particle adding some Gaussian noise to guarantee particle diversity.

C. Particle Optimization

Figure 10. Best Particles obtained in the first 3 iterations for two different poses (Particles = 100).

The described method was applied in real scenarios (as seen in Figure 10). We found that 4 iterations with 100 particles are enough for a good detection performance. This is only possible because the particle initialization procedure gives already good approximations to the pose, improving the convergence rate of the optimization phase.

The average likelihood for the real scenario in Figure 10 was calculated (Figure 11) changing the particle number (N) and the database poses (D) used for the initialization. When D is lower than N the remaining possibilities are created adding Gaussian noise to the best possibilities. In the plots the

two horizontal lines correspond to the selected “Threshold” (upper line) and “Threshold min” (lower line) that control the coarse and fine phases of the optimization algorithm depicted in Figure 6. Bellow “Threshold Min” we are at the bootstrap phase (best database possibilities and some Gaussian noise depending of the selected parameters), in the middle we are at the coarse optimization phase and above “Threshold” we are at the fine optimization phase.

As we can see from the Figure 11 the convergence to the coarse optimization is very fast (in average 2 iterations), demonstrating the importance of the initialization process in the convergence to the final result. It is also shown that more database possibilities do not necessarily correspond to better results, for example the 200 particles used in Figure 11 for 150 and 200 database possibilities. The “Threshold” parameter must be set to a value that corresponds to acceptable particle pose likelihood, being a compromise between speed and accuracy. The best speed vs performance parameters obtained are 100 particles with 100 poses used from the database.

The average computational time for each iteration is shown in Table I. To calculate the magnitude and angle of the gradient of each frame (used for the contour likelihood calculation) an average running time of 25 ms must be added. Thus, for 100 particles, we get an average time of 790 ms (approximately 1.26 fps).

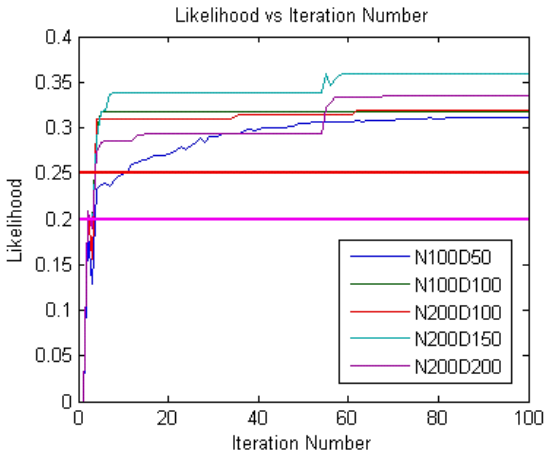

Figure 11. Average likelihood vs Iteration number.

TABLE I

COMPUTATIONAL TIME

Particle Number (N)	Time (ms)	Frames per second (fps)
1	7.5	133.3
10	72	13.8
20	148	6.7
50	374	2.6
100	765	1.3
200	1659	0.6

D. Quantitative Performance Evaluation

A test set was created for several UAV distances: 5, 10, 15, 20, 25, 30, 35, 40, 45 and 50 meters, using 850 synthetic frames for each distance. The final pose estimate was obtained from the most likely particle running the proposed algorithm with 100 particles and 100 database poses.

As we can see from the Figure 12 the translation error decreases with proximity, obtaining a mean value at 5 meters (as seen in Table II) of 0.27 meters. The landing area is an irregular area of 5x6 meters, so we need to guarantee a minimum translation error of 1 meter in order to ensure a reliable landing.

As we are operating in an outdoor environment and with moving platforms, a sudden variation in the atmospheric conditions can lead to failure. This resolution in translation is clearly achieved, guaranteeing a UAV good estimation in 3D position across the range of distances covered in this test. We expect to further increase the accuracy of the system and reduce the amount of outliers (outliers are currently less than 5% of the cases) by using the UAV dynamic model in a temporal filtering and data association framework.

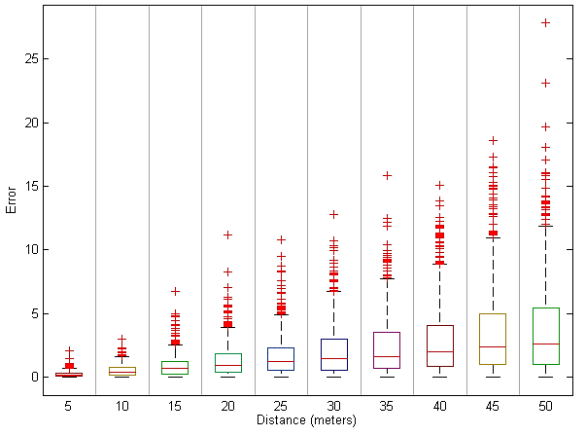

Figure 12. Translation Error (meters).

TABLE II

TRANSLATION ERROR

Distance (meters)	Mean value (meters)	Median value (meters)
5	0.27	0.19
10	0.54	0.40
15	0.92	0.70
20	1.32	0.92
25	1.68	1.28
30	2.1	1.45
35	2.4	1.66
40	2.9	1.99
45	3.5	2.43
50	3.8	2.61

The rotation error at 5 meters also decreases with proximity but less markedly than with translation. It has a median value of 9.4 degrees (as seen in Table III). This error values can be achieved by the combination of the hybrid likelihood formula as described in section II.C for distances under 25 meters. The higher error for distances above 25 meters happens mainly because the UAV model has the majority of its pixels concentrated in its wings and the used likelihood formula is based on the maximization of the difference between two pixel areas, originating a lower sensibility in variations that happen in the rest of its body. The rotation error becomes

particularly evident in poses where the UAV body is partially occluded by its wings, generating some situations where the particle is shifted around 180 degrees from the observed pose. This ambiguity will be tackled in future work by using dynamic constraints between pose and velocity direction in a temporal filtering framework.

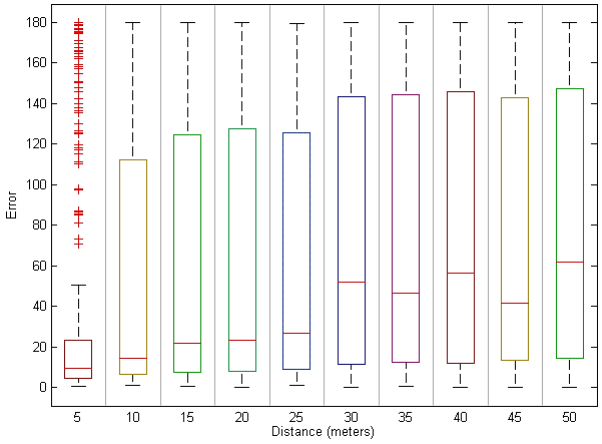

Figure 13. Rotation Error (Degrees).

TABLE III

ROTATION ERROR

Distance (meters)	Mean value (Degrees)	Median value (Degrees)
5	37.2	9.4
10	52.3	14.6
15	60.5	22.1
20	62.5	23.3
25	63.3	27.0
30	75.4	52.2
35	75.1	46.6
40	77.2	56.5
45	74.9	41.8
50	79.3	61.7

IV. CONCLUSIONS

A method was introduced and tested for estimating the pose of a known UAV in images acquired onboard a ship, for the purpose of automatic landing. The presented algorithms features a UAV detection method based on a cascaded classifier; a novel particle initialization methodology with a pre-trained database of images indexed by bounding box properties; and a particle optimization stage inspired in evolutionary methods that has shown interesting convergence properties. The implemented architecture allows a very accurate position estimation (about 4% median error at 5 m) and a reasonable attitude error (about 10º median error at 5m). We consider these precision levels suitable for the following stages of the work, which will focus on using temporal filtering frameworks and dynamic constraints to complete the tracking system.

ACKNOWLEDGMENT

This work was funded by the EU Commission within the National Strategic Reference Framework (QREN) under grant agreement 23260 (AUTOLAND) and by the Portuguese government through FEDER funds under project SEAGULL (SI IDT 34063) and by the FCT project [PEst-OE/EEI/LA0009/2013].

REFERENCES

1. Wu, A.D., 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), 2013. **10**(4): p. 14.
2. Cesetti, A., 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, 2010. **57**(1-4): p. 233-257.
3. Xu, G., Y. Zhang, S. Ji, Y. Cheng, and Y. Tian, *Research on computer vision-based for UAV autonomous landing on a ship.* Pattern Recognition Letters, 2009. **30**(6): p. 600-605.
4. Kong, W., 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. IEEE.
5. Martínez, C., 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. Ieee.
6. Gonçalves-Coelho, A.M., L.C. Veloso, and V.J.A.S. Lobo, *Tests of a light UAV for naval surveillance*, in *IEEE/OES Oceans'2007*. 2007: Aberdeen, UK.
7. Doucet, A., N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. 2001: Springer.
8. Haug, A.J., *Bayesian Estimation and Tracking: A Practical Guide*. 2012: Wiley.
9. Lepetit, V. and P. Fua, *Monocular Model-Based 3D Tracking of Rigid Objects: A Survey*. Foundations and Trends® in Computer Graphics and Vision, 2005. **1**(1): p. 1-89.
10. Challa, S., *Fundamentals of Object Tracking*. 2011: Cambridge University Press.
11. Forsyth, D.A. and J. Ponce, *Computer Vision: A Modern Approach*. 2011: Pearson Education, Limited.
12. Boli, M., P.M. Djuri, and S. Hong, *Resampling algorithms for particle filters: a computational complexity perspective.* EURASIP J. Appl. Signal Process., 2004. **2004**: p. 2267-2277.
13. Park, S., J.P. Hwang, E. Kim, and H.-J. Kang, *A new evolutionary particle filter for the prevention of sample impoverishment.* Trans. Evol. Comp, 2009. **13**(4): p. 801-809.
14. Kwok, N.M., 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. IEEE.
15. Simon, D., *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. 2006: Wiley.
16. Pietikäinen, M., A. Hadid, G. Zhao, and T. Ahonen, *Computer Vision Using Local Binary Patterns*, in *Computer Vision Using Local Binary Patterns*. 2011, Springer London. p. E1-E2.
17. Rosten, E. and T. Drummond, *Machine Learning for High-Speed Corner Detection*, in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Editors. 2006, Springer Berlin Heidelberg. p. 430-443.
18. Rosten, E., R. Porter, and T. Drummond, *Faster and Better: A Machine Learning Approach to Corner Detection.* IEEE Trans. Pattern Anal. Mach. Intell., 2010. **32**(1): p. 105-119.
19. Taiana, M., 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, 2010. **58**(6): p. 784-795.
20. Taiana, M., J.C. Nascimento, J.A. Gaspar, and A. Bernardino. *Sample-Based 3D Tracking of Colored Objects: A Flexible Architecture*. in *BMVC*. 2008.
21. Choi, C. 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. IEEE.