# Method based on a particle filter for UAV trajectory prediction under uncertainties[†]

**Roberto Conde*, José A. Cobano,***
**Aníbal Ollero*,****

**Escuela Superior de Ingenieros de la Universidad de Sevilla, 41092 Sevilla, Spain*
*e-mail: mamuso@gmail.com; jacobano@cartuja.us.es, aollero@cartuja.us.es.*
***Centro Avanzado de Tecnologías Aeroespaciales, Sevilla, 41309 Spain.*

**Abstract:** This paper presents a method based on particle filters to predict trajectories of unmanned aerial vehicles under uncertainties. The particle filter is able to cope with the influence of different sources of uncertainty such as the atmospheric conditions, the model of the UAV and the limitations of the sensors and control system on board the UAV. In the case of UAV the most important source during the flight is the wind. Several simulations that show the significance of the proposed method are presented. The obtained results also point out the interest of the resampling process of the particle filter.

## 1. INTRODUCTION

In the last ten years, important progress in aerial robotics has been achieved. Currently, the unmanned aerial vehicles (UAVs) have evolved until becoming aerial robots able to execute autonomously takeoff, transition between waypoints and landing. Also, systems that allow the cooperation of multiple aerial vehicles (Maza and Ollero, 2004) and the integration of these vehicles with sensors and actuators in the environment (Ollero *et al*., 2007) have been developed and implemented.

One of the most important problems that appears in the prediction of the trajectories described by an UAV is the uncertainty. This can be due to atmospheric conditions, the accuracy of the UAV model used to predict, and the limitations of the sensors and on board control system to cancel perturbations. In the case of the UAVs, the most important source of uncertainty during the flight is the changes in the atmospheric conditions.

Fig. 1 shows an UAV of the Robotics, Vision and Control Group (University of Seville) and Fig. 2 describes an experiment of autonomous flight of this UAV. In this experiment the UAV should follow the waypoints defined in the four vertexes of the polygon. The real trajectory followed by the UAV can be observed. Note the discrepancy due to the previously mentioned sources of uncertainty. In this experiment, the wind is the most important one (see Fig. 2).

The decreasing of the uncertainty in the trajectory prediction allows a better tracking of the flight plan. Furthermore, if the uncertainty is lower, the minimum required separation between UAVs could be also decreased allowing the increasing of traffic.

UAV conflict detection and avoidance requires a model to predict the UAV trajectories. In a probabilistic configuration, the model could be any empirical distribution of the predicted positions of the aerial vehicle (Paielli and Erzberger, 1997) or a dynamic model as a stochastic differential equation (Hu *et al*., 2003; Prandini *et al*., 2000) that describes the motion of the aerial vehicle and defines implicitly a distribution for predicted positions.

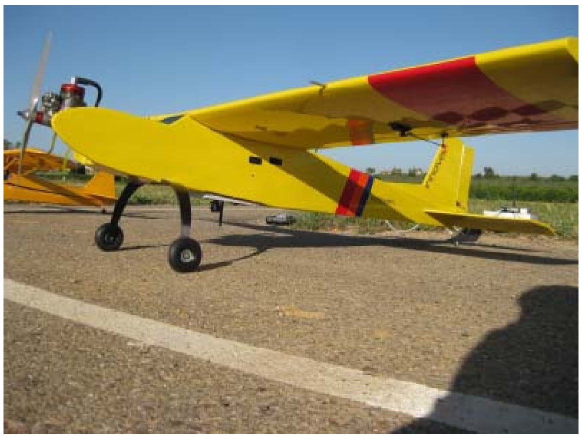
Fig. 1. Fixed wing UAV used in the experiments

This paper presents the implementation of a particle filter, based on atmospheric and vehicle models, to predict the trajectory of an aerial vehicle by considering the evolution of the uncertainty.

This paper is organized in four sections. Section 2 describes a particle filter basic algorithm and its implementation. Section 3 presents simulations showing the main features of the method. Conclusions are detailed in section 4.

[†] This work has been funded by the projects SADCON (2005/TEP-375 Andalusian Regional Government) and AWARE (European Commission IST 2006-33579).

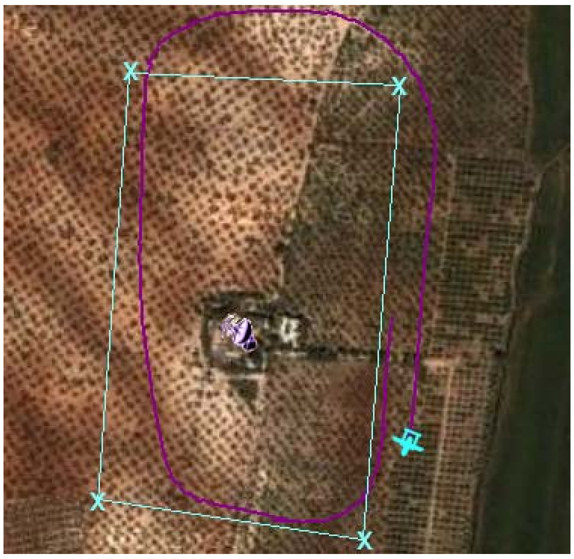

Fig. 2. Tracking of the desired trajectory (blue line) and the real trajectory (violet line) under wind effects

## 2. PARTICLE FILTER

Bayesian location is based on probabilistic techniques and provides the estimated position of a mobile robot from the observations of the robot and its control actions.

The Bayes rule is used to compute the posterior probability density function, that can be expressed as the probability $p(z_t|x_t)$, of the last measurement $z_t$, averaged by the prior probability density function of the robot position, $p(x_t|x_{t-1},a_{t-1})$ and the prior probability , $p(x_{t-1})$:

$$p(x_t) = \alpha p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, a_{t-1}) p(x_{t-1}) \qquad (1)$$

where $\alpha$ is a normalizing factor that ensures $\int_{x_t} p(x_t) = 1$

Depending on the technique used to compute the probability density function, hereafter p.d.f., several approaches of the bayesian location can be characterized. One of them is the particle filter. Particle filters allow solving the problem of the bayesian location representing the posterior p.d.f., which estimates the most probable positions of the robot. This is normally multimodal (Thurn *et al.*, 2005).

Particle filters solve this problem by approximating the posterior p.d.f. by random samples of a particle set that represent the most probable states of the robot. These location algorithms compute the probability that the robot is located in each of the particles using the Bayesian model, that is, the p.d.f. of each particle is computed (Thrun, 2002). The particles whose probability is negligible will not be considered, as they are not useful to describe the robot state.

### *2.1 Basic algorithm*

Particle filters represent the distribution of the expected state by a set of random samples drawn from the distribution. The main benefit of this kind of nonparametric distribution model is that it can represent many different types of unknown distributions, not only Gaussian, but also multimodal ones.

Particle filters are a nonparametric implementation of the Bayes filter, which can be used to estimate the p.d.f. of the state of a Markov process, given the previous outputs. Let $x$ be the state of the process, then the state space equations of the process are:

$$\begin{aligned} x_{k+1} &= f(x_k, v_k) + g(u_k, n_k) \\ z_k &= h(x_k, w_k) \end{aligned} \qquad (2)$$

where $z_k$ represents the current output, $u_k$ represents the current input, $v_k$, $w_k$ and $n_k$ are samples of a random variable, and *f, g* and *h* are known functions.

The particle filter will estimate the posterior p.d.f of $x_{0:k}$ , using the previous estimation obtained from the sampling of a set of particles that represent instances of the state trajectories. Therefore, the particle filter is a sequential Monte Carlo method, or recursive nonparametric Bayesian filter, with a belief function:

$$bel(x_{0:k}) = \eta p(z_k \mid x_k) p(x_k \mid x_{k-1}, u_k) bel(x_{0:k-1}) \quad (3)$$

where $\eta$ is a normalizer that ensures the probability of $x_{0:k}$ being in the space of the state trajectories is equal to 1.

As the space over the state sequence is dimensionally huge, and thus unfeasible to cover with particles, in practice the estimation will be done on $bel(x_k)$ instead of $bel(x_{0:k-1})$.

The algorithm of the basic Particle Filter is:

```
**ParticleFilter**$(X_{k-1}, u_k, z_k)$
1: $\tilde{X}_k = X_k = 0$
2: for $m = 1$ to $M$ do
3:     sample $x_k^{[m]} \sim p(x_k|u_k, x_{k-1}^{[m]})$
4:     $w_k^{[m]} = p(z_k|x_k^{[m]})$
5:     $\tilde{X}_k = \tilde{X}_k + \langle x_k^{[m]}, w_k^{[m]} \rangle$
6: for $m = 1$ to $M$ do
7:     draw $i$ with probability $\propto w_k^{[m]}$
8:     add $x_k^{[i]}$ to $X_t$
9: return $X_t$
```

This algorithm calculates $bel(x_k)$, and, when $M \to \infty$, it is equivalent to calculate $bel(x_{0:k})$ and extracting the last elements of the state sequence vector.

Inputs are the particle set in the time *t-1*, control $u_t$ and the measurement $z_t$ in the time *t*.

Line 3 of the algorithm represents the simulation of the motion of the aerial vehicle; that is, an estimation of the next vehicle state given the last known state and inputs. This will be done using the UAV and atmospheric models. In our case the aim is to estimate the future state of the process at a particular future time step $k_f$, i.e. we look for $bel(x_{kf})$.

Therefore, the process lie in having the particle filter running normally until the current time $k_c$, and then start a pure simulation of the particle set until the desired time $k_f$ without the use of any measurement. When a measurement is taken, the particle set can be modified at the measurement time if necessary by making a resampling.

### 2.2 Importance resampling

Importance resampling is a method to decrease the variance of the estimation process. It is based on the idea that some samples of the observable p.d.f. are more significant than others, and thus sampling them with a higher frequency will improve our estimator confidence.

In line 4 of the previously presented algorithm, we have set the weight $w_k$ of a particle as the probability of measuring $z_k$ (that in fact is what we have already measured) given the state $x^{[m]}_k$. This weighting function would give each particle the importance it deserves, making the particles that better represent the actual state more probable to be chosen and vice versa. The choice of the weighting function is a key for the accuracy of the algorithm, particularly when the size of the particle set cannot be large enough. Typical weighting functions are the squared distance to the measured state, or a Gaussian n-dimensional function centred on the state vector.

Next a particle filter algorithm with resampling process is shown:

```
**ParticleFilter**$(X_{k-1}, u_k, z_k)$
1:  $\tilde{X}_k = X_k = 0$
2:  **for** $m = 1$ **to** $M$ **do**
3:      sample $x_k^{[m]} \sim p(x_k | u_k, x_{k-1}^{[m]})$
4:      **if** exists $z_k$ **then**
5:          $w_k^{[m]} = \eta p(z_k | x_k^{[m]}) w_{k-1}$
6:          $\tilde{X}_k = \tilde{X}_k + \langle x_k^{[m]}, w_k^{[m]} \rangle$
7:  $N_{eff} \leftarrow 1 / \sum_{i=1}^{M} (w_k^{[i]})^2$
8:  **if** $N_{eff} < N_{th}$ **then**
9:      **for** $m = 1$ **to** $M$ **do**
10:         draw $i$ with probability $\propto w_k^{[m]}$
11:         add $x_k^{[i]}$ to $X_t$
12: **return** $X_t$
```

The resampling process is done at lines 9 and 10. The main idea is to build a new set using the last set and its weighting factors so that it is a better estimate of the posterior distribution.

As measurements are not available in each time step, we have to redefine the algorithm.

Also, as the resampling can sometimes be an unnecessary time consuming process, we can restrict it to some condition. A possible way is shown in line 8 of the previous algorithm. We have introduced here the concept of effective number of particles of a set, $N_{eff}$.

As $\eta$ is a normalizing factor that ensures that $\sum_{i=1}^{M} w_i = 1$, $N_{eff}$ is a real number between 1 and $M$ indicating the number of particles that are actually useful. Thus, $N_{eff}$ near 1 means that there is only a particle that has almost all the weight of the set, while values of $N_{eff}$ near to $M$ means that all particles have almost the same weight.

### 2.3 Implementation

In order to use particle filters, models of both the UAV and the atmosphere are needed.

In this article a simple UAV model proposed in (McLain and Beard, 2005) is used. Nevertheless, in the proposed method it is also possible to use models of arbitrary complexity. The model describes the behaviour of a controlled aerial vehicle:

$$\dot{x}_i = v_i \cos(\psi_i) \tag{3}$$

$$\dot{y}_i = v_i \sin(\psi_i) \tag{4}$$

$$\ddot{h}_i = -\alpha_{\dot{h}} \dot{h}_i + \alpha_h (h_i^c - h_i) \tag{5}$$

$$\dot{\psi}_i = \alpha_\psi (\psi_i^c - \psi_i) \tag{6}$$

$$\dot{v}_i = \alpha_v (v_i^c - v_i) \tag{7}$$

where $(x_i, y_i, h_i)$ represents the 3D coordinates, $h_i$ is the altitude, and $\psi_i$ is the heading of the vehicle. $\alpha_v, \alpha_\psi, \alpha_h$ and $\alpha_{\dot{h}}$ are known parameters that depend on the characteristics of the vehicle. The following constraints with regard to $\dot{\psi}_i$ and the speed are used:

$$-c \le \dot{\psi}_i \le c \tag{8}$$

$$v_{\min} \le v_i \le v_{\max} \tag{9}$$

where $c$, $v_{min}$ y $v_{max}$ are positive constants that depend on the dynamic of the vehicle. Note that the constraints of minimum speed can be irrelevant in some aircrafts, such as helicopters, but have an important paper in fixed wing UAVs.

Equations (3)-(7) model the behaviour of the UAV taking into account the references in speed, altitude and heading.

The parameters of the above model can be estimated using real flight data of the UAV presented in Figure 1. It should be noted that the full model takes into account not only the above parameters but also high level control considerations about the waypoint tracking. This can be accomplished using the MATLAB nonlinear optimization tools over a large set of flight data from particular flight experiments carried out for that purpose.

The used atmospheric model solely indicates the wind vector speed, $\rho$, which in the simulations will only have values in the horizontal plane. The value of each component of $\rho$ is the average of a normal distribution with fixed standard deviation. Thus, each particle is affected by different wind disturbances.

In the implemented algorithm two relevant steps can be emphasized:

1.- Initialization

The initialization process gives a probable state of the aerial vehicle to each particle in the set. In the current simulations, this is done by assigning them random values in the neighbourhood of the measured vehicle state. It is also possible to use the knowledge about the sensor systems uncertainties.

2.- Resampling

In the current resampling process, each particle will be assigned a probability of being chosen that is proportional to its weight. Therefore the particles with a higher weight are resampled with a higher probability.

Next an implementation algorithm of the resampling step is presented:

```
**Random weighted resampling**(X,w)
1:  $min\_weight \propto 1/M$
2:  $set\_seq(0) \leftarrow w(0) * min\_weight$
3:  for $m = 1$ to $M-1$ do
4:      $set\_seq(m) \leftarrow w(m) * min\_weight + set\_seq(m-1)$
5:  for $m = 1$ to $M-1$ do
6:      $number \leftarrow set\_seq(M-1) * rand()$
7:      $chosen\_id \leftarrow 0$
8:      while $set\_seq(chosen) < number$ do
9:          $chosen\_id++$
10:     $X_{new}(m) \leftarrow X(chosen\_id)$
11:     $w(m) \leftarrow 1/M$
12: Return $X_{new}$
```

Inputs are $X$, the current particle set, and $w$, the corresponding vector of weights. Output is the new resampled set, $X_{new}$.

In this algorithm, *set_seq* is a sequence that represents the particle set. Its elements are the value of the previous element plus a proportional variable to the weight of the particle that it represents (see lines 1-4).

Then the particles are chosen randomly for the resampling process, but following a distribution that gives more probability to the ones with higher weights: a number between 0 and the last sequence value is generated using a random number generator, *rand()*; then the index of the first sequence value that exceeds that number will identify the chosen particle.

That process is repeated as many times as the size of the particle set, until a new set, $X_{new}$, is formed.

## 3. SIMULATIONS AND EXPERIMENTS

In this section, two simulations are presented with two different trajectories to demonstrate the particle filter operation using the aerial vehicle and atmospheric model described in section 2.3.

Besides, a simulation that uses a tracking control model of the UAV and real data instead of simulated nominal trajectories is also presented.

In the first simulation, the reference angle, $\psi^c$, is set to 0 radians and $\rho$ equal to [1, 1]m/s. The results obtained in the first simulation are shown in Fig. 3. A deviation in the trajectory angle can be observed due to wind influence. The evolution of the particle set around the nominal trajectory can also be observed, and the reduction of the uncertainty when a resampling is done.

In Figures 4 and 5 a detailed view of the evolution of the particle set is shown. In Fig. 4 the first iterations of the simulation can be observed. The particle set is gathered around the initial location of the vehicle (first point cloud) following a distribution that takes into account the error sources coming from the on-board sensors. The next point clouds are computed from particle set location samples each second. The expansion of the point clouds is due to the uncertainties and the different initial conditions of each particle.

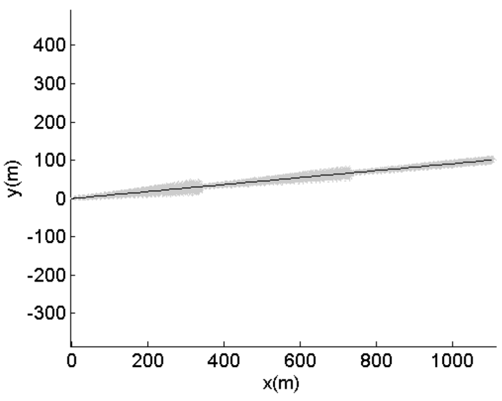

Fig.3. Evolution of the particle set (grey points) around the nominal trajectory (black line) with $\psi^c$ =0º and $\rho$=[1, 1]m/s

When $N_{ef}$ is less than $N_{lim}$, the resampling process is carried out. Therefore, the uncertainty of the trajectory prediction decreases (see Fig. 3) by ruling out the less significant particles (lower weights $w$) and replacing them by others with higher weights. From that moment, each particle will evolve independently until the next resampling.

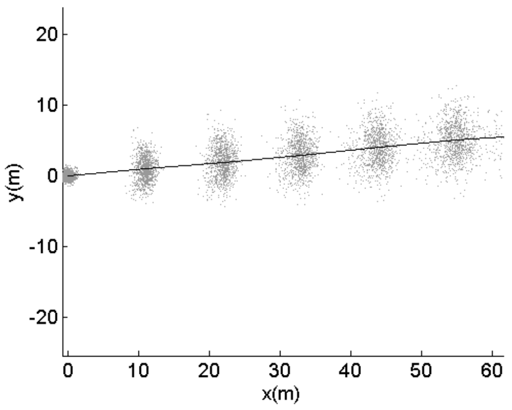

Fig. 4. First iterations of simulation 1

In the second simulation, $\psi^c$ varies to describe a circular trajectory centered on the origin (see Fig. 6). In this simulation $\rho$ is set to [0, 1]m/s.

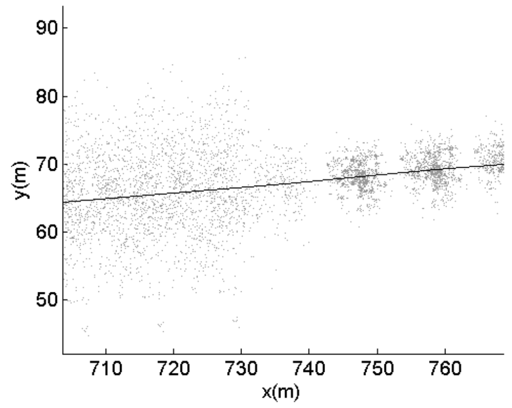

Fig. 5. Resampling process in simulation 1

In order to show the effect of resampling, the evolution of a particle set without carrying out this process is shown in Fig. 6. The consistent progressive growth of the uncertainty is translated into a gradual growth of the point cloud.

When the resampling is done, the uncertainty decreases in a similar way as in the first simulation (see Fig. 7). In this case it has to be noted that due to the differences in $\rho$ and the trajectory, the distribution of the clouds have different shapes.

In order to analyze the effect of the wind in over the trajectory prediction, four different simulations with different $\rho$ have been carried out. All simulations consider the same control law in $\psi^c$.

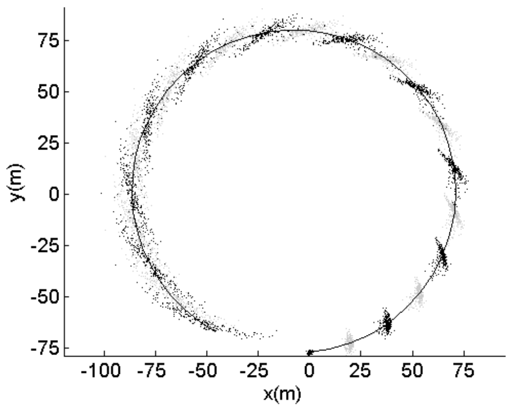

Fig.6. Evolution of the particle set around the nominal trajectory (black line) with $\psi^c$ following a circular law and $\rho$=[0, 1]m/s. Without resampling

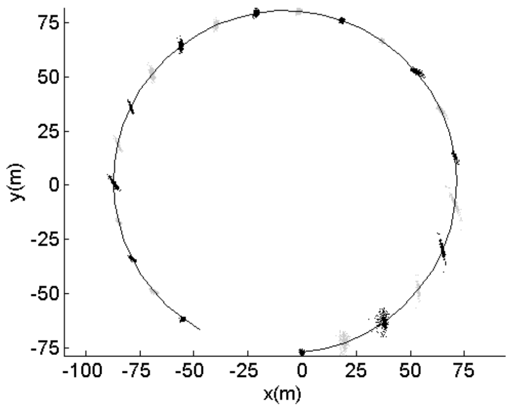

Fig. 7. Evolution of the particle set around the nominal trajectory (black line) with $\psi^c$ following a circular law and $\rho$=[0, 1]m/s. With resampling

Mean and maximum errors and mean standard deviation for each of the four simulations are shown in Table 1. These errors are computed regarding the mean, maximum and standard deviation over each point cloud for the entire trajectory.

**Table 1. Trajectory prediction error in four different scenarios**

| Error (m) | Wind speed vector $\boldsymbol{\rho}$ (m/s) | | | |
|---|---|---|---|---|
| | [0, 1] | [1, 0] | [-1, 0] | [0, -1] |
| Mean | 1.3121 | 1.2153 | 1.3475 | 1.3737 |
| Maximum | 13.3903 | 11.0736 | 10.3345 | 9.5199 |
| Std. dev. | 0.8614 | 0.8732 | 0.9082 | 0.8502 |

Several simulations have been carried out using the algorithm described in section 2 and data extracted from real flights of the UAV presented in section 1. The model of the UAV motion is based on the one presented in section 2.3. Its behaviour can be seen in Fig. 8.

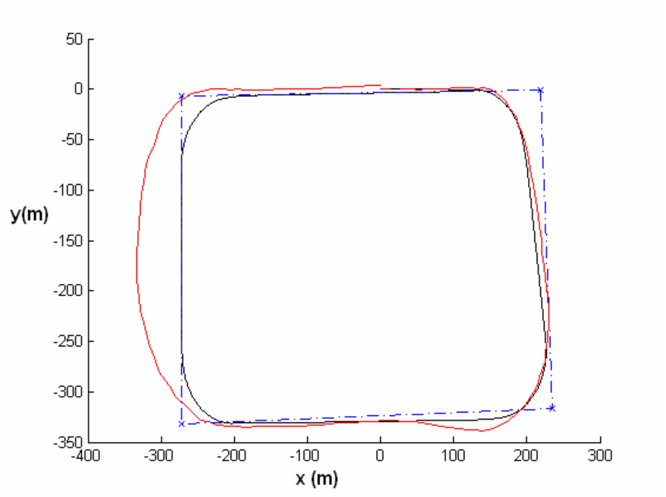

Fig. 8. Real trajectory (red line) influenced by the wind, simulated trajectory using UAV model without wind (black line), and waypoints linked with segments (dotted blue line)

In the simulation in Fig. 9, the particle filter achieves a drastic reduction of the uncertainty of the predicted trajectory. Note that these predictions are substituted on every resampling, so a long-term prediction will result on a Montecarlo simulation from the current state on, but with a more accurate model of the wind inferred by the particle filter.

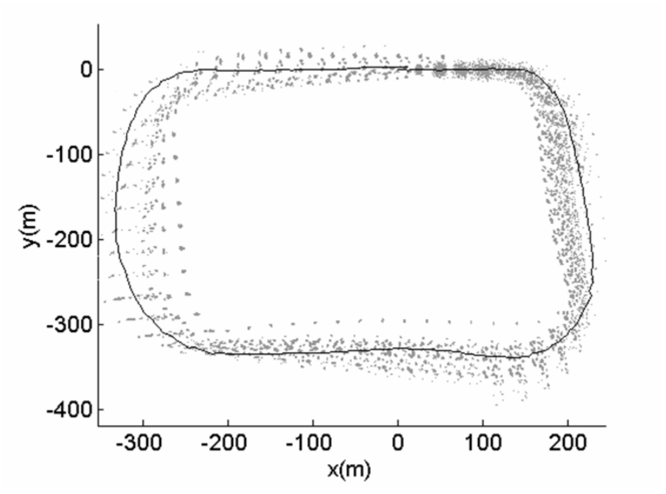

Fig. 9. Simulation of the particle set (grey points) around the real trajectory (black line) with resampling. A wind model is inferred by the filter, and the uncertainty decreases.

## 4. CONCLUSIONS

The estimation of the uncertainty in the trajectory prediction is a main issue for many UAV applications. Particularly, these estimations are required for obstacle detection and avoidance in both single UAV and multi-UAV systems.

This paper has presented a method for the estimation of this uncertainty by using particle filters. The method uses UAV and atmospheric models of arbitrary complexity, although this complexity should be limited for the real-time application of the method.

The obtained results are coherent and point out the interest of the resampling process of the particle filter.

Furthermore, the results with the real flight data lead us to the same conclusion. Further improvements of the model will provide a better estimation of the uncertainty.

The future work will include the real-time implementation of the proposed method for the estimation of the uncertainty in UAV collision avoidance.

## REFERENCES

Hu, J., Prandini, M. and Sastry, S, (2003). Aircraft conflict detection in presence of spatially correlated wind perturbations. *Proceedings of the AIAA Guide, Navigation and Control Conference,* Austing, TX.

McLain T. W. and Beard, R. W. (2005). Coordination variables, coordination functions and cooperative timing missions. *Journal of Guidance, Control and Dynamics ,* **vol. 28,** no. 1, pp. 150-161.

Ollero, A., Bernard, M., La Civita, M., van Hoesel, L., Marrón, P., Lepley, J., and de Andrés, A. (2007). AWARE: Platform for Autonomous self-deploying and operation of wireless sensor-actuator networks cooperating with unmanned Aerial Vehicles. *Proceedings of the IEEE Workshop on Safety, Security and Rescue Robotics.* Rome.

Maza, I. and Ollero, A. (2004). Multiple UAV Cooperative Searching Operation Using Polygon Area Decomposition and Efficient Coverage Algorithms *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic System. Toulouse, France.* DARS 2004, pp. 211-220.

Paielli, R., and Erzberger, H. (1997). Conflict probability estimation for free flight. *Journal Guid. and Control Dyn.* **vol. 20**, no. 3, pp. 588-596.

Prandini, J., Hu, J., Lygeros, J., and Sastry, S. (2000). A probabilistic approach to aircraft conflict detection. *IEEE Trans. Intell. Transp. Syst.* **vol. 1**, no. 4, pp. 199-220.

Thurn, S. (2002). Particle Filters in Robotics. *Proceedings of Uncertainty in Intelligence Artificial.*

Thurn, S., Burgard, W., and Fox, D. (2005). Probabilistic Robotics. MIT Press.