AIAA Guidance, Navigation and Control Conference and Exhibit
18 - 21 August 2008, Honolulu, Hawaii

AIAA 2008-7236

# Feature Extraction of Low Dimensional Sensor Returns for Autonomous Target Identification

Christopher W. Lum,* and Rolf T. Rysdyk,†
*Autonomous Flight Systems Laboratory*
*University of Washington, Seattle, WA, 98105, USA*

**This work considers algorithms for maritime search and surveillance missions. During these type of missions, an agent searches for a target using its various sensors. Performing target identification and classification of sensor returns is a crucial component of the mission. This paper investigates a system to process returns from a low dimensional sensor and automatically classify the data. This system uses an algorithm that employs the sensor and motion model of the agent to augment the limited data from the sensor. Several differentiating features are then extracted and used to train various classifiers and machine learning algorithms.**

## Nomenclature

| | |
|---|---|
| $\Delta\psi$ | Change in heading |
| $\sigma_{sensor}$ | Standard deviation of sensor |
| $\chi_t$ | Particle filter set at time $t$ |
| $\psi$ | Heading |
| $C_t$ | Sum of particle filter weights at time $t$ |
| $f$ | Feature vector |
| $g()$ | Sampling function |
| $h()$ | Target magnetic signature function |
| $\tilde{h}_i()$ | False anomaly $i$ signature function |
| $M$ | Total number of particles |
| $p(A\|B)$ | Conditional probability of $A$ given $B$ |
| $T$ | Number of particle filter updates for a trajectory |
| $\overline{u}$ | Control vector |
| $V_a$ | Airspeed |
| $V_t$ | Variance of weights at time $t$ |
| $w_t^{[m]}$ | Weight on particle $m$ at time $t$ |
| $W_t$ | Weight set at time $t$ |
| $\overline{x}$ | State of agent |
| $z_t$ | Sensor measurement made by agent |
| $z_t^{[m]}$ | Sensor measurement expected by particle $m$ |

## I. Introduction

In many modern missions, unmanned system have shown potential and promise to greatly increase efficiency and success rates. One such mission is the search and surveillance type application which is a specialized type of Intelligence, Surveillance, and Reconnaissance (ISR) mission. Although Unmanned

*Research Assistant, Dept. of Aeronautics and Astronautics, lum@aa.washington.edu, AIAA student member
†Advanced Development, Insitu, rolf.rysdyk@insitu.com, AIAA member

Aerial Vehicles (UAVs) appear to be ideal for these operations, these missions typically require heavy human involvement. High operator involvement is required in logistics and operation; for example, distributing assignments such as which regions to search and coordinating and interpreting subsequent sensor measurements. Often times, the number of required operation crew for these autonomous systems exceed those of their manned counterparts. Therefore, the primary limitation to concurrent operation of multiple vehicles remains lack of autonomy of these vehicles.

A primary goal in a searching mission is typically to find a target in some region.[1] As an agent performs its search, it may encounter the target it is searching for or it may encounter an object which produces a sensor reading but is not the desired target (a false anomaly). The main challenge in this situation is to correctly identify or classify these encountered anomalies. Typically, the target identification process is handled by a human operator. It is the operator's responsibility to monitor sensor outputs and determine if readings correspond to the desired target, false anomalies, or extraneous noise. Many unmanned systems carry optical sensors such as cameras or infra-red imaging devices. Autonomously processing the output of these sensors and classifying readings requires significant computational resources and effort and in many cases, a human operator will outperform a computer generated algorithm in terms of classification accuracy and reliability. This can be mainly attributed to the fact that the output of the sensor is a complex multi-dimensional set of data with many features being explicit and more easily extracted and recognized by a trained human operator. As the number of agents involved in the search mission increases, so does the required number of human operators. A large team of expensive agents with sophisticated sensors may not be the most effective system. A more scalable and implementable team would be comprised of several smaller agents with less complex sensors. These more primitive sensors may output a smaller dimension signal which a human operator may have difficulty interpreting. Automating the target identification process may have more promise in this context. In a noisy environment, it becomes difficult for a human operator to classify sensor readings and assign confidence in these readings because the useful features are implicit in the signal. In this situation, the useful features in the signal can be extracted and then used to train a machine learning algorithm to perform the classification which may have better accuracy and reliability than the human operator.

The topic of machine learned classifiers is a large and well studied field. One of the most well known family of classification algorithms are the boosting algorithms generated by Schapire[2] and Viola et al.[3] Groups such as Lester et al.[4] have applied this idea and studied the problem of fusing data from a wearable unit with multiple sensors and extracting features from which to train boosting algorithms for automatic classification of human activities. Raj[5] and Fox[6] et al. addressed a similar problem using Rao-Blackwellized particle filters and hidden Markov models (HMMs) to incorporate temporal smoothness into the task of classifying human activities from the same wearable multi-sensor unit. Previous work[7] has investigated the potential for using particle filters for target identification on a basic level.

This work contributes to the large area of research in autonomous target identification. The main research focus is to develop a system which processes the output from a low dimensional sensor and extract relevant features from which to train a classifier. This paper investigates the use of particle filters instead of the traditional HMMs to incorporate temporal smoothness and regularity into the target identification system.

The example used in this paper is an agent searching for a target based on its magnetic sigature. The agent is equipped with a simple scalar magnetometer which can measure the magnetic field magnitude at its current location. Section II introduces the concept of a local total magnetic intensity (TMI) map. The use of this map in searching for magnetic anomalies is also explained. With this framework in place, a Monte-Carlo type simulation is performed to generated training data which is detailed in Section III. Section IV describes the extraction of relevant features from the training data to train a classification algorithm discussed in Section V. Finally, conclusions and continuing research directions are presented in Section VI.

## II. Aeromagnetic Data Surveys

### A. The Georanger autonomous aeromagnetic survey vehicle

For aeromagnetic surveys, the agent (UAV) is essentially a mobile sensor. The vehicle autonomy serves the engineering user who simply specifies an area of interest and, after some processing, receives a corresponding set of data. In this idealized perspective, the data-analyst is unconcerned with the method with which the data was obtained. To achieve such an objective, autonomy is required at several hierarchical levels. The current work focuses on target detection and classification tactics. The vehicle serving as the sensor platform

in this work is the Fugro Georanger, provided by The Insitu Group, and is shown in Figure 1.

The Georanger is a derivative of the ScanEagle vehicle which has the following performance specifications:

| | |
|---|---|
| Max Takeoff Weight | 41.9 lb / 19 kg |
| Payload | 15.4 lb / 7 kg |
| Endurance | 15 hours |
| Service Ceiling | 16400 ft / 5000 m |
| Max Level Speed | 70 knots / 36 m/s |
| Cruise Speed | 49 knots / 25 m/s |
| Wing Span | 10.2 ft / 3.1 m |
| Fuselage Diameter | 7.0 in / 0.2 m |
| Length | 4.9 ft / 1.5 m |

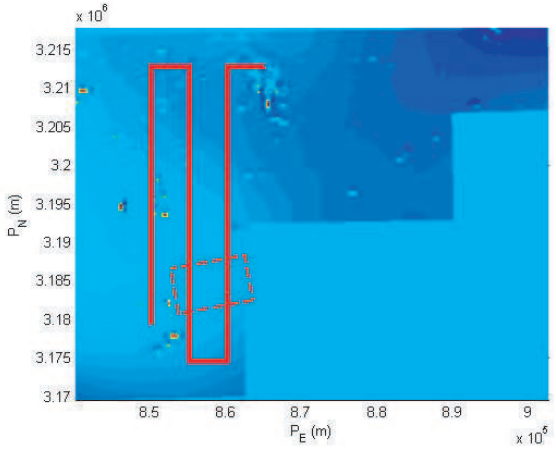
Each agent is equipped with a magnetometer to measure the total magnetic intensity at its current location. This data is relayed to a ground station. The crucial piece of information required by the ground station is a local magnetic map of the region where the search is taking place. This map of the TMI of the region may be acquired using analytical models such as the WMM-2000 or WGS-84 model.[8] However, since these models are coefficient-based analytical models, they do not capture temporal or small local variations in magnetic field strength. Therefore, a more accurate map is obtained by performing an actual survey over the area of interest to collect the necessary data.

**Figure 1. Image of the Fugro autonomous aeromagnetic survey vehicle, the 'Georanger I' by the Insitu Group.**

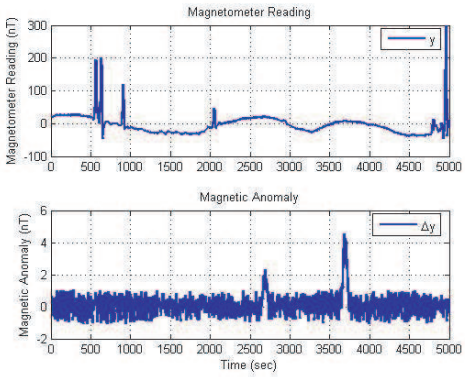
## B. Total Magnetic Intensity maps

When an actual search is executed, differences between the ground station map of the magnetic field and the actual magnetic field will appear as magnetic anomalies. In this work, to minimize the number of false anomaly encounters and to increase the accuracy of the evaluation, actual magnetic survey data is used as a local TMI map. This data is provided by Fugro Airborne Surveys. The data was collected by a manned aircraft equipped with a magnetometer to measure the TMI. This information, coupled with a GPS position, provides the TMI in "line data" form. This data can then be interpolated into a 100x100 meter grid. TMI readings at locations other than survey points are linearly interpolated from this grid. A magnetic map of a region in the Gulf of Mexico and a simple grid search trajectory are shown below in Figure 2(a). Here, the data is acquired in an approximate 60x50 km grid. The regions of uniform color denote areas where survey data is not available, creating the staircase appearance. Assuming that there are only permanent fixtures in the region when the map is acquired, this map now makes up the reference set of data on the ground station.

In addition to a local magnetic map, a magnetic model of the desired target is also required. In the following example, the magnetic signature of the target (an idealized submarine) is modeled as a simple two dimensional Gaussian distribution, shown in Figure 3. The magnetic signature of the target is a function




(a) Total magnetic intensity map.




(b) Associated magnetic traces.

**Figure 2. The total magnetic intensity map and trajectory over area with corresponding magnetometer readings.**

of many variables, namely depth of target, sensor altitude, etc. For current purposes, the target is assumed stationary and at a fixed depth, thereby rendering the magnetic signature static. Assuming that the magnetic signature of the target simply adds to the total magnetic intensity of the local region in a linear fashion,[9] anomalies can easily be identified by simply subtracting the magnetometer reading from the local reference map which is stored on the ground station.

The described approach can be used to compare magnetometer readings with the reference data to create a differential measurement. Large differential measurements imply the presence of a new magnetic anomaly and possible target. If the agent does not fly over any targets, the magnetic anomaly should be near zero. Small non-zero anomaly encounters can be attributed to temporal variations in magnetic field and sensor noise. A simple grid search pattern was shown previously in Figure 2(a). The location of the target is shown as a dashed red box and the trajectory of the agent is shown in the solid red line (starting in the lower left corner). The associated total magnetic intensity trace and differential measurement trace is shown in Figure 2(b). The total magnetic intensity reading as the agent flies over this trajectory is shown in the upper trace and the differential measurement is shown in the lower trace. As the agent flies this search trajectory, the sensor measurement is constantly compared to the reference data set to generate a differential measurement. As can be seen in Figure 2(b), given the differential magnetometer reading, the anomaly encounter can be easily detected (two spikes at approximately 2700 and 3700 seconds) even though the actual range of absolute measurements may be large.

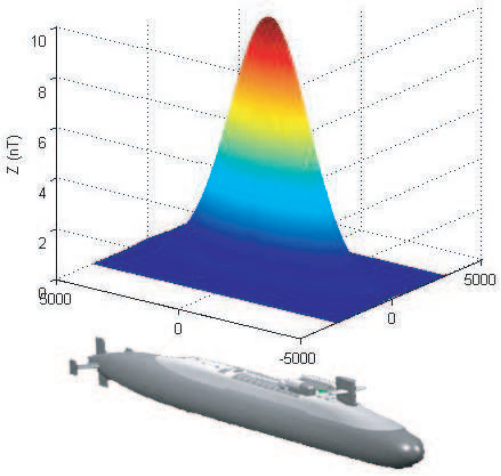



**Figure 3. Magnetic signature of target. Magnetic signature given by $z = h(x, y)$.**

Magnetic anomalies can be caused by many factors such as temporal variations in the magnetic field or false anomaly encounters (i.e. boats/vessels). Once a magnetic anomaly is encountered, it must be identified and classified. On a simplistic level, the overall goal is to either classify the anomaly as the desired target or a false reading. Obviously, it would be simple to identify the anomaly if the entire magnetic signature of the anomaly is obtained (the UAV flies over the entire boxed region in Figure 2(a)). However, this requires many passes over a potential target, and significant time to make the necessary measurements. If the anomaly is moving or evading, this may not be feasible. To further exacerbate the problem, if the agent is equipped with a simplistic sensor (like a simple scalar magnetometer as in this case), the data obtained from a single pass may be limited. The question now becomes, given only one or two passes over the anomaly, is it possible to correctly identify or provide a probability that this anomaly is indeed the target being sought after using

this limited sensor data?

## III. Generating Training Data

Generating a representative set of training data is crucial to the performance of the resulting classifier. This section describes how the training data is generated.

### A. Simulated Trajectories

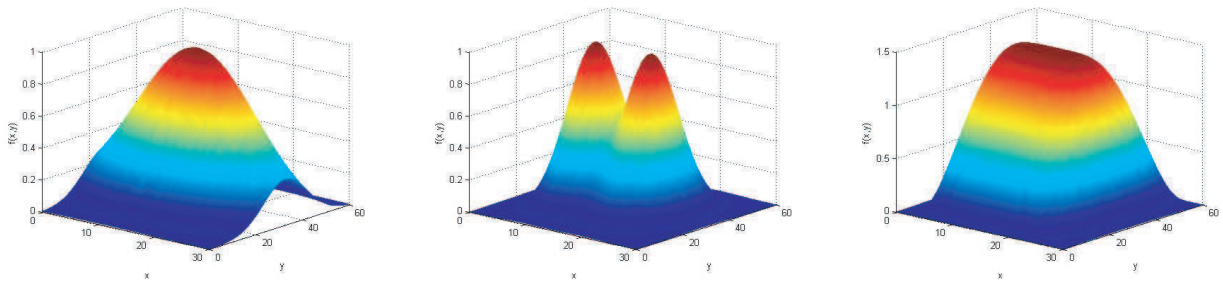
In practice, it is assumed that the agent's orientation in the earth frame is known via GPS. Because the anomaly's location and orientation is not known, the anomaly's frame of reference with respect to the earth frame is unknown. In practice, the agent will fly over a region and encounter an unexpected sensor reading (Figure 2(b)). It can be inferred from this reading that an anomaly was encountered, but it is unknown which part of the anomaly was flown over that generate this spurious measurement. One problem that comes to mind in this situation is how to estimate the state of the agent with respect to the target's frame of reference. In this context, the agent's state and control vectors are given by

$$\overline{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \\ \psi(t) \end{bmatrix} = \begin{bmatrix} x^{tgt}_{uav/tgt}(t) \\ y^{tgt}_{uav/tgt}(t) \\ \psi_{uav/tgt}(t) \end{bmatrix} \qquad \overline{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = \begin{bmatrix} V_{uav}(t) \\ \Delta\psi_{uav/tgt}(t) \end{bmatrix} \tag{1}$$

The agent is model as a simple planar kinematic vehicle with dynamics given by

$$\overline{x}(t+1) = u_1(t)\Delta T \begin{bmatrix} \cos(x_3(t) + u_2(t)) + x_1(t) \\ \sin(x_3(t) + u_2(t)) + x_2(t) \end{bmatrix} \tag{2}$$

A simulation environment was developed that allowed an agent to fly over a magnetic anomaly. The magnetic anomaly is either the true target with magnetic signature as described previously in Figure 3 or a false anomaly. To make the problem challenging, the false anomalies are similar to the true anomaly. An example of three false anomaly functions are shown below in Figure 4.



(a) $\tilde{h}_1(x, y)$ (b) $\tilde{h}_2(x, y)$ (c) $\tilde{h}_3(x, y)$

**Figure 4. Possible false anomaly signatures**

A single trajectory consists of an agent flying a path over an anomaly. As the agent flies over an environment, its sensor makes a scalar reading ($z(k) \in \Re$) at each time step. Because only the differential measurement is desired, the magnetic contribution by the environment is neglected. The sensor is modeled as

$$z(t) = \begin{cases} h(x_1(t), x_2(t)) + g(N(0, \sigma^2_{sensor})) & \text{if true target encounter} \\ \tilde{h}_i(x_1(t), x_2(t)) + g(N(0, \sigma^2_{sensor})) & \text{if false anomaly encounter} \end{cases} \tag{3}$$

In Eq. 3, $g$ is a sampling function which simply chooses a sample from a probability density function (in this case, a Gaussian distribution with zero mean and variance $\sigma^2_{sensor}$). The noise parameterized by $\sigma_{sensor}$

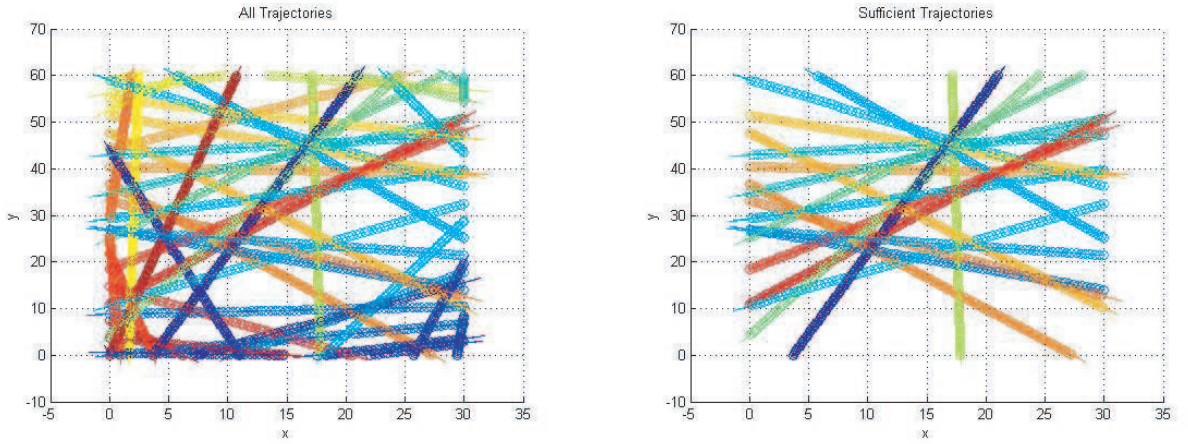
is used to model the variation in the magnetic field and also to model sensor noise. The anomaly function $h$ or $\tilde{h}_i$ is either the true target or one of the false anomalies described previously.

A Monte Carlo simulation is run with random, linear trajectories over the target. This generates many examples of trajectories and measurements of the agent flying over both the true target and the false anomalies. Of these runs, only ones that meet a certain requirement are selected to be used as training examples.

### B. Selecting Sufficient Runs

The rate of convergence of most learning algorithms is dependent on the quality of the training data. It is desirable to train a classifier using examples which are distinguishable and are good representations of actual trajectories and to eliminate ambiguous examples which may confuse the classifier. In this application, a run is deemed sufficient if it satisfies several criteria. It is unlikely that a measurement trace over a false anomaly will be able to be discerned from a trace over the true anomaly if there are only a small number of measurements made. Therefore, a run is deemed sufficient only if the agent makes a minimum number of measurements as it flies over an anomaly. Similarly, a run is deemed sufficient only if the magnitude of the differential sensor measurement exceeds a certain threshold. This eliminates runs where the agent flies over the edge of the anomaly parallel with the axis of the anomaly (where the anomaly magnetic signature is nearly zero).

The runs that are used as training examples must meet both requirements. This process helps ensure that the training examples that are fed to the learning algorithm are distinctive enough to train a decent classifier. An example of the total output of the Monte Carlo simulation and the selection of sufficient runs is shown in Figure 5.




(a) All trajectories

(b) Sufficient trajectories used as training examples

**Figure 5. Removing insufficient trajectories from Monte Carlo simulation**

Despite being selective about the runs to use as training data, at this point, the raw sensor measurements from these trajectories do not provide enough distinction to differentiate the class of anomaly. An example of the raw sensor measurements from both a true and false anomaly encounter are shown in Figure 6. In this example it can be seen why it is difficult to discern the class of the anomaly based solely on these raw sensor readings because the traces are so similar. It is difficult to determine the relevant features to look for in order to classify the anomaly. The task of extracting the relevant features from the sensor measurements is addressed in the next section.

## IV. Feature Extraction

This section discusses processing the raw sensor measurements in order to generate a feature vector which can be used to train a classification algorithm.

As evidenced by Figure 6, the raw sensor measurements offer limited amounts of useful data when it

comes to classifying the anomaly. One option is to simply use the the sensor measurement at each time step as an individual training example. Theoretically, it should be possible to create a classifier which is more successful than random guessing using this simple feature. For example, if the maximum sensor reading exceeds the maximum of the target signature, one can be fairly certain that the anomaly encountered is a false anomaly. However, this allows for multiple classifications within the same trace. Although this method provides an abundance of training data, it is virtually impossible to generate a reliable classifier using this simple a feature set.

Improvements to the accuracy can be made by using the assumption that during an anomaly encounter, the agent only encounters the true target or a false anomaly (it cannot encounter both together). This means that an entire trace should be classified as one or the other. Instead of using the sensor value at every time step, one can extract features like the maximum/minimum sensor values over the entire trace. This yields a single training example for each trajectory. This helps incorporate some temporal smoothness to the classification as this yields only a single classification per anomaly encounter.

Basic features such as maximum/minimum sensor readings and averages can be extracted fairly easily. These basic features help discern the classes somewhat. However, more ambiguous cases require additional features.

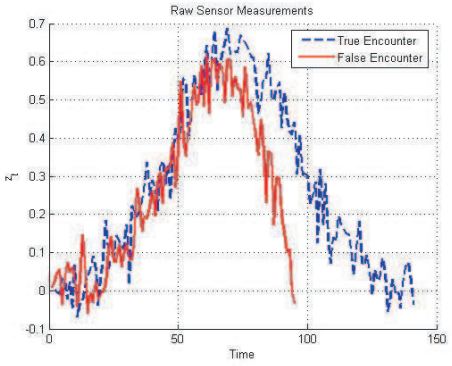



**Figure 6. Raw sensor measurements from both a true and false anomaly encounter.**

### A. Particle Filter Features

The previously mentioned features based on the raw sensor measurements still prove to be insufficient to train an accurate classifier. Further information can be gleaned by considering both the sensor model (Eq.3) and the dynamic motion model (Eq.6) of the agent. Since the data traces are generated using these models, a method which considers these temporal aspects has a much greater potential to yield useful features.

Typically, temporal smoothness is incorporated using Hidden Markov Models or Dynamic Bayesian Networks. These are useful to reduce classification noise in continuous traces. Some applications are classifying human activities[4,5] where the classification can switch between different affectivities in a reasonable manner (for example, humans might transition from walking to sitting but typically do not rapidly transition between sitting and riding a bike). In effect, this makes use of the agent's motion model to predict the state and thereby augment the classification. Although this method can be used in this application to smooth the data, it does not explicitly use information about how the sensor measurements are gathered (the sensor model in Eq. 3). In this application, the temporal aspects of both the motion and sensor model are addressed using particle filters.[10]

A particle filter is a recursive, non-parametric Bayes filter technique which estimates the states of a system using a finite number of state hypotheses.[11] In this situation, the state vector that is being estimated is the position of the agent with respect to the target, expressed in the target's frame of reference and the relative heading of the agent with respect to the target.

$$\overline{x}_t^{[m]} = \begin{bmatrix} x_{uav/tgt}^{tgt} \\ y_{uav/tgt}^{tgt} \\ \psi_{uav/tgt} \end{bmatrix} \tag{4}$$

Each individual state hypothesis, $\overline{x}_t^{[m]}$, is referred to as a particle, and together they make up the particle filter set, $\chi_t$.

$$\chi_t = \bigcup_M \overline{x}_t^{[m]} = \left\{ \overline{x}_t^{[1]}, \overline{x}_t^{[2]}, \ldots, \overline{x}_t^{[M]} \right\} \tag{5}$$

Historically, particle filters have been employed in this manner to perform tasks such as localization[12] and state estimation.[13] In this situation, the goal of the filter is to estimate the state of the agent (position and orientation with respect to the target, expressed in the target's frame of reference). The particle filter performs this estimate using two main steps, a prediction and correction step.

### 1. *Prediction*

In the prediction step, each particle is propagated forward in time using a motion model of the individual agent.

$$\overline{x}_t^{[m]} = g\left(p\left(\overline{x}_t^{[m]} | \overline{u}_t, \overline{x}_{t-1}^{[m]}\right)\right) \tag{6}$$

Each new particle is created from the old particle and the current control (applied to transition particle at time $t-1$ to time $t$). The term $p\left(\overline{x}_t^{[m]} | \overline{u}_t, \overline{x}_{t-1}^{[m]}\right)$ is a multi-dimensional probability density function of the new state given the old state and current control. Notice that in this formulation, the state transition is not a deterministic process. This stochastic aspect actually has important implications regarding the robustness of the particle filter.[11]

Although $p\left(\overline{x}_t^{[m]} | \overline{u}_t, \overline{x}_{t-1}^{[m]}\right)$ may be difficult to compute analytically, Eq. 6 is implemented in simulation by simply adding noise to the control and then propagating the state forward using a deterministic motion model in Eq. 6. In simulation, the noise added to each element of the control vector is obtained by sampling from a normal, Gaussian distribution with a variable standard deviation, $\sigma$. The standard deviation is a function of the actual control applied to the agent, $\overline{u}_t$. In effect, as $||\overline{u}_t||$ increases, so does $\sigma$. Physically, this translates into a model whose state transition becomes more uncertain as the agent moves faster or executes larger heading changes.

In addition to the control input at each time step, the actual sensor measurement observed by the agent, $z_t$, is made available to the particle filter. Each particle is then assigned a weight, $w_t^{[m]}$, based on how likely it is to make the same sensor measurement at its current state $\left(w_t^{[m]} \propto p\left(z_t | \overline{x}_t^{[m]}\right)\right)$. In effect, a higher weight should be assigned to particles whose states are close to the actual state, $\overline{x}_t$. Notice that this does not require a sampling function like Eq. 6 because $z_t$ and $\overline{x}_t^{[m]}$ are known at this point. This is another description of the sensor model of the agent. It allows for the fact that even though a particle's state may be vastly different than the true state of the agent, if the sensor is poor or unreliable, it has the possibility of still making the same sensor reading as the agent.

The sensor model used in simulation calculates the weights by creating an error between the particle sensor measurement and the true sensor measurement and then using this as the argument of a Gaussian distribution.

$$w_t^{[m]} = (2\pi\sigma_{sensor}^2)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\frac{(z_t - z_t^{[m]})^2}{\sigma_{sensor}^2}\right) \tag{7}$$

In Eq. 7, $z_t^{[m]}$ is the predicted sensor measurement made by particle $m$ computed using the same sensor model in Eq. 3. As stated previously, $\sigma_{sensor}$ is a measure of the sensor's accuracy. A larger $\sigma_{sensor}$ implies an unreliable sensor; therefore, particles that do not make the same measurement as the true agent still receive high weights. Note that the weight is not a probability, but this still achieves the goal of assigning high weights to particles that are more likely to have states which are similar to the true agent state. Similar to the particle set, $\chi_t$, the weight set at a given time step $t$ is given by

$$W_t = \bigcup_M w_t^{[m]} = \left\{w_t^{[1]}, w_t^{[2]}, \ldots, w_t^{[M]}\right\} \tag{8}$$

### 2. *Correction*

Now that each particle has been propagated forward and assigned a weight, it becomes necessary to correct the particle filter set so that it comes closer to representing the actual state of the agent. This process is known as resampling.

As stated before, the particle filter's estimate of the state is made up of all the particles. Currently, the particle filter set contains particles which have both high and low weights. As more and more sensor measurements are acquired, it is desired that high scoring particles are replicated and kept in the next generation population whereas low scoring particles are discarded. The important feature in this evolutionary process is that the particles are resampled with replacement so that the total number of particles remains

constant at each cycle. Any type of evolutionary scheme, such as survival of the fittest, can be used to evolve the current population to the next.

In simulation, a roulette wheel method is used. In this method, $M$ bins are created (one for each particle). The size of each bin is directly proportional to the weight of the associated particle. The bins are placed next to each other and a random number is then generated. The bin in which the random number falls then has its associated particle included in the next population. This process is repeated $M$ times and is synonymous to spinning a roulette wheel $M$ times where the number and size of the slots on the wheel are directly proportional to $M$ and the weights, respectively.

Using the roulette wheel method yields resampling proportional to the weights. This allows for a particle to be copied multiple times in the next generation. This also generates a small probability that particles with low weights have the possibility to survive to the next generation as well.

One important feature of the particle filter is the ability to use different motion and sensor models. This allows for a team of agents to be comprised of different types of vehicles and sensors. This simply requires modifying the motion and sensor models of each particle filter for each member of the heterogeneous team.

### *3. Execution*

When an agent encounters an anomaly whose magnitude exceeds the noise threshold (approximately 1 nT in this case), the particle filter is started in an attempt to estimate the state of the agent with respect to the target. The particle filter's progression as the agent flies diagonally over the target is displayed over a top down view of the target signature (Figure 3) and is shown below in Figure 7.

In this sequence, the large red circle represents the actual location of the agent and the solid red line represents the agent's trajectory over the target. The smaller purple dots represent the particle filter's many different hypotheses of the possible state of the agent (position north, position east, and heading). The actual agent crosses over the target starting in the lower left corner and flies over it to the upper right corner. Also note that the initial distribution of particles is not simply random over the domain. Since the algorithm is recursive, the number of iterations before convergence is based on its initial condition. Incorporating a priori knowledge that the particle filter is started when the anomaly magnitude exceeds 1 nT suggests that the particles be clustered along the level curves where the target signature is 1 nT.

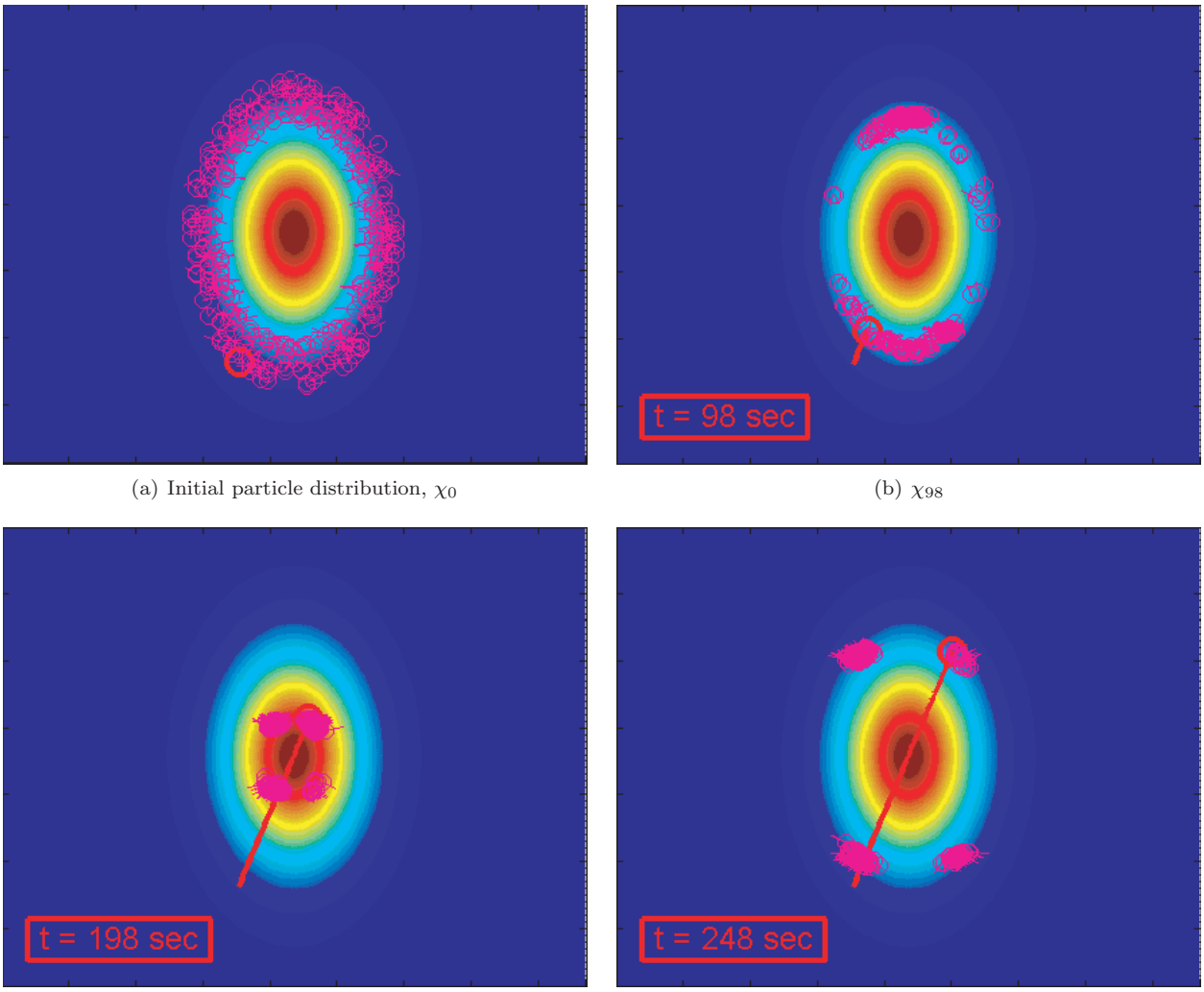
As the agent obtains more and more sensor measurements (at a simulated rate of 1 Hz), the particle filter is able to eliminate particles which are inconsistent with the current measurement and resample these particles to regions which have a higher probability of producing the actual sensor reading, $z_t$. This is why as time progresses, the particles become concentrated around the actual UAV location. Near the end of the simulation, there are four distinct groups of particles. This is due to the symmetry of the underlying target signature. Each of these four groups of particles are equally likely because each group would produce the correct actual sensor readings. In effect, $z_t^{[m]} \approx z_t \, \forall m$. Because of this symmetry, the particle filter is not able to uniquely identify the position of the agent with respect to the target. This would require multiple passes over the target and more sensor measurements.

Although the goal of the particle filter is to estimate the position of the agent with respect to the target in the target frame of reference, in the larger picture, the location of the target with respect to the agent in the agent frame of reference is more useful because it then becomes simple to locate the target in the earth frame of reference (agent's position and orientation in the earth frame of reference is known from GPS). Each particle can be transformed using Eq. 9.

$$\begin{bmatrix} x_{tgt/uav}^{uav} \\ y_{tgt/uav}^{uav} \\ \psi_{tgt/uav} \end{bmatrix} = \begin{bmatrix} -\cos(\psi_{uav/tgt}) & \sin(\psi_{uav/tgt}) & 0 \\ -\sin(\psi_{uav/tgt}) & -\cos(\psi_{uav/tgt}) & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_{uav/tgt}^{tgt} \\ y_{uav/tgt}^{tgt} \\ \psi_{uav/tgt} \end{bmatrix} \tag{9}$$

When each particle is transformed in this fashion, the distribution of the target location with respect to the agent in the agent's frame of reference becomes as shown in Figure 8.

As shown in the first two plots in Figure 8(b), it appears that the particle filter now has a somewhat unique estimate of the location of the target relative to the agent as shown by an approximate unimodal distribution in $x_{tgt/uav}^{uav}$ and $y_{tgt/uav}^{uav}$ centered at approximately 0 and -2250, respectively. However, notice that the distribution of $\psi_{tgt/uav}$ is obviously a multimodal distribution. This distribution is actually the sum of four peaks which should ideally be centered at $\pm 11.3$ degrees and $\pm 168.7$ degrees. Since the number of




(a) Initial particle distribution, $\chi_0$

(b) $\chi_{98}$

(c) $\chi_{198}$

(d) $\chi_{248}$

**Figure 7. Particle filter progression during a target encounter. The solid line indicates actual aircraft position relative to target signature, while the particles concentrate about possible positions**

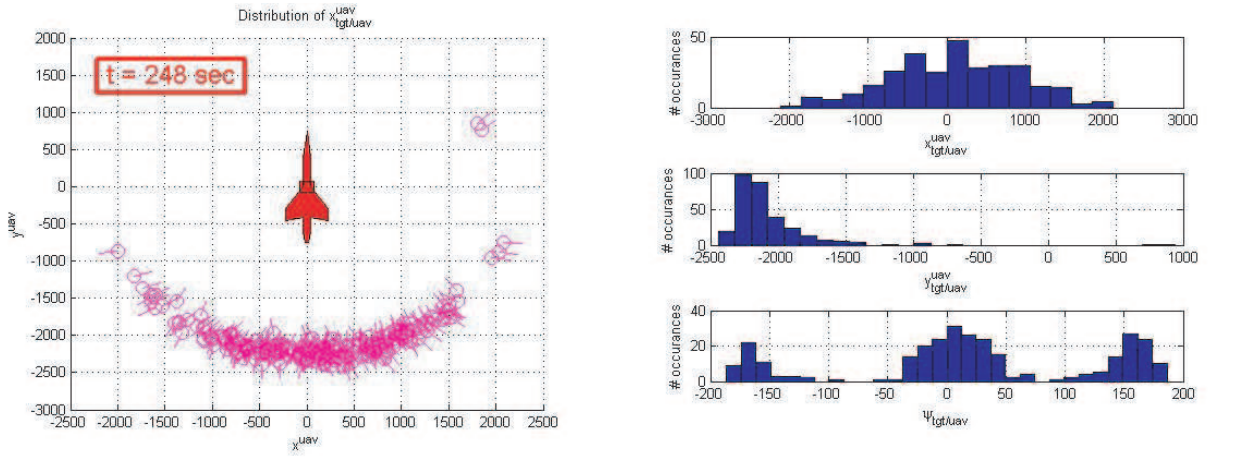
particles was not large enough and since the motion and sensor models of the particle filter were not highly accurate, the two peaks centered at $\pm 11.3$ degrees appear as a single peak at 0 degrees.

This multimodal distribution in $\psi_{tgt/uav}$ reflects the four distinct state hypotheses shown previously in Figure 7(d). However, if the orientation of the target is not desired, then by transforming the particles, it is possible to obtain a unique estimate of simply $x^{uav}_{tgt/uav}$ and $y^{uav}_{tgt/uav}$. Note that this is only the case when the agent happens to fly directly over the target as shown in this example. In a more general case where the agent passes over the target off-centered, then even with the transformation of the particles, the location of the target cannot be determined uniquely (but the number of possible locations may be reduced).

## B. Generating a Feature Vector

The majority of this section has discussed addressing the state estimation problem using the particle filter method. Deviating from the standard usage of the filter, a closer look at the weights, $w_t^{[m]}$, is warranted in the context of target identification.

A scalar quantity which collectively measures the overall accuracy of the particle filter can be obtained by simply summing all the weights. If most of the particles are in locations that are similar to the true state, then the sum of the overall weights should be large.




(a) Distribution of transformed particles

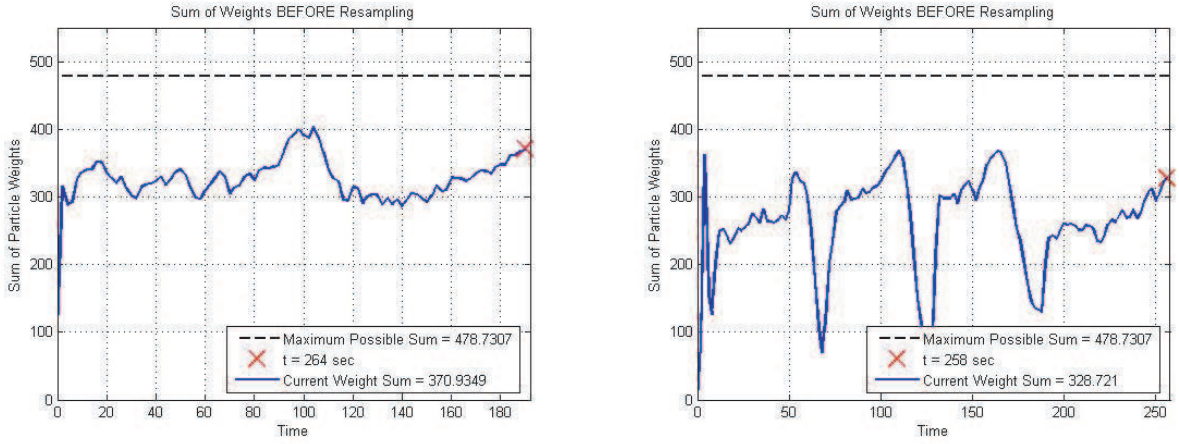
(b) Histogram distribution of $x^{uav}_{tgt/uav}$, $y^{uav}_{tgt/uav}$, and $\psi_{tgt/uav}$

**Figure 8. Particles now represent position and orientation of target with respect to the agent in the agent's frame of reference.**

$$C_t = \sum_{m=1}^{M} w_t^{[m]} \tag{10}$$

This trace of a $C_t$ vs. $t$ might be considered a side-effect of estimating $\overline{x}_t$, but as will be shown later, this is the main piece of information that will be used to address the target identification problem.

The particle filter will attempt to estimate the agent's state regardless of whether the anomaly encountered is the actual target or a false anomaly. A method to identify the target is now required. The sum of all the particle weights, $C_t$, provides a quantitative measure of how confident the particle filter is that the anomaly encountered is the actual target. If all or most of the particles are resampled to areas which are near the actual state of the agent, then most of the weights will be fairly high. The sum of the particle weights for an encounter with the actual target and an encounter with a false anomaly is shown below in Figure 9.




(a) True target encounter

(b) False anomaly encounter

**Figure 9. Sum of all particle weights during a true target encounter and a false anomaly encounter.**

In Figure 9, the difference between a true target encounter and a false anomaly encounter is fairly clear. In the situation where the agent encounters the true target, the confidence measure increases initially as the particles are quickly resampled to locations which are consistent with the actual sensor measurements and

then stays fairly constant. However, in the case where the agent encounters a false anomaly, the particle filter regularly "loses confidence" as inconsistent sensor measurements are obtained. This is characterized by the sharp drops in the sum of the particle weights.

The variance of the particle weights also is a good indicator of filter confidence. If the variance is high, this implies that the weight of the particles are spread out. Since a high weight indicates a good state estimate, if the filter is not very confident in its state estimate, the variance is high. The variance at each time step of a single trajectory is recorded to be used in generating features later.

$$V_t = \mathrm{Var}(W_t) \tag{11}$$

Using the raw sensor readings and the particle filter outputs, an eleven dimensional feature vector is generated

$$f = \begin{pmatrix} T \\ M(2\pi\sigma^2)^{-1/2} \\ \max_{t=1,\ldots,T}(z_t) \\ \frac{1}{T}\sum_{t=1}^{T} z_t \\ \frac{1}{T}\sum_{t=1}^{T} V_t \\ \frac{1}{T}\sum_{t=1}^{T} C_t \\ \frac{1}{T}\sum_{t=1}^{T} \frac{C_t}{M(2\pi\sigma^2)^{-1/2}} \\ \max_{t=1,\ldots,T}(V_t) - \min_{t=1,\ldots,T}(V_t) \\ \max_{t=1,\ldots,T}(C_t) - \min_{t=1,\ldots,T}(C_t) \\ \max_{t=1,\ldots,T-1} |V_{t+1} - V_t| \\ \max_{t=1,\ldots,T-1} |C_{t+1} - C_t| \end{pmatrix} \tag{12}$$

The first two features are constants which are functions of the particle filter execution. The first feature is simply the number of measurements made by the agent. The second feature is the maximum sum of the weights possible at any point during the particle filter execution. The third feature is simply the maximum sensor reading encountered and is included for reasons discussed previously. The next three features are the average sensor reading, variance of weights, and sum of weights, respectively. The seventh feature is the average weight proportion which is somewhat redundant but is helpful when visualizing data. The next two are the changes in variance of the weights and sum of the weights over the entire particle filter execution. Finally, the last two features are the maximum change in variance of the weight and sum of the weights in a single step, respectively.

A single feature vector is generated for each measurement trace and represents a single example for training a classifier. Using these features with various learning algorithms are discussed in the next section.

## V. Learning a Classifier

### A. Training Classification Algorithms

The flow chart for the classification system is shown in Figure 10. A simulated trajectory over an anomaly is generated using the specified sensor and motion model. If this trajectory is deemed sufficient, the particle

filter is run using the controls and measurements. Features are then extracted from these controls and measurements and the outputs from the particle filter algorithm.

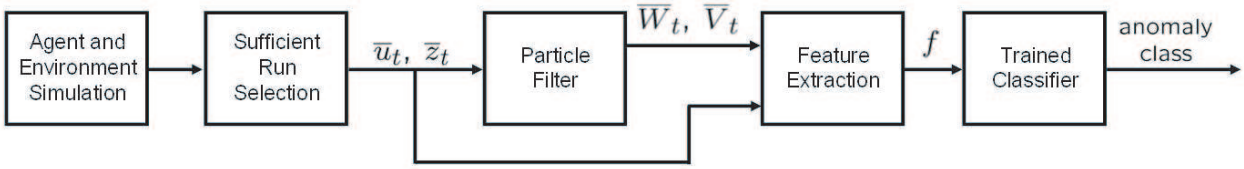



**Figure 10. Flow diagram for classification system.**

To generate training examples, 1700 random trajectories were flown over both the true target and various false anomalies (roughly equal proportions). Of these 1700 trajectories, 836 were deemed sufficient for feature extraction. Once the features are extracted from these sufficient runs, a set of 836 training/testing examples are generated. This data set is evaluated with several different learning schemes as described in the next section.

### B. Results

Several different classifiers are trained and tested using these examples. This is done using the Weka toolkit.[14] Each classifier was trained and tested using a stratified 10-fold cross-validation scheme. The results for four learning methods are displayed in Table 1

**Table 1. Classification results for various learned algorithms.**

| | C4.5 Revision 8 Classified As | | Decision Stump Classified As | | Alternating Decision Tree Classified As | | Multi-Layer Perceptron Classified As | |
|---|---|---|---|---|---|---|---|---|
| | True | False | True | False | True | False | True | False |
| Actual True | 232 | 40 | 0 | 272 | 197 | 75 | 247 | 25 |
| Actual False | 70 | 494 | 0 | 564 | 57 | 507 | 59 | 505 |
| Accuracy | 86.84% | | 67.46% | | 84.21% | | 89.95% | |

As can be seen from Table 1, most of the learning algorithms perform very well with an average accuracy of 87% (excluding the overly simplistic decision stump method). These results are encouraging considering that there are only 11 features in each example. Often times, hundreds of features are extracted for each example in order to obtain these accuracies. Of course, of these large feature sets, only several features are useful for the classification. This leads to the belief that these 11 features are valuable in this application. Extracting more features might serve to increase accuracy but probably not by a large margin.

The decision tree generated by the C4.5 Revision 8 algorithm is shown in Figure 11. Although there is not much to be learned by looking at the individual nodes and leaves of the tree, the structure illustrates why this method of extracting a low number of features is desirable. The computational cost of executing the particle filter (and extracting the feature vector) is relatively high. However, once the features are extracted, the actual classification is fairly simple. The resulting decision tree is small enough to write down and illustrate. Therefore, this tree can be implemented easily for a real-time application. Once the features are extracted from the data, the actual classification is not computationally intensive (in this case requiring at most eight if-statements).

## VI. Conclusion and Further Research

The main challenge in searching for a target in a noisy environment is identifying the anomalies that are encountered by the agent. This is challenging because in many cases, the agent is equipped with a simple sensor which captures a limited amount of data. The particle filter method gracefully captures the temporal aspects of how the data was captured by using both the motion and sensor model of the agent

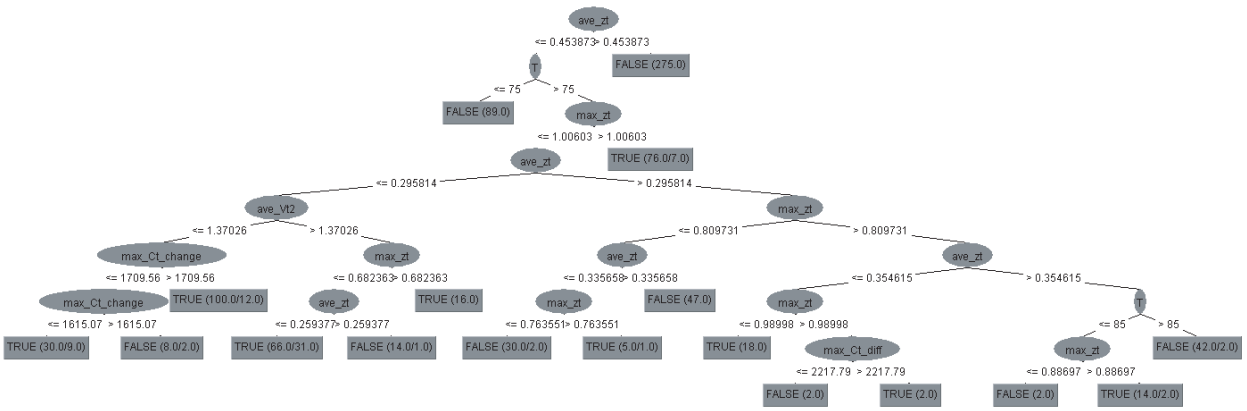



**Figure 11. Decision tree generated by C4.5 Revision 8 algorithm.**

and maximizes the amount of information gained from the sensor returns. The outputs from the particle filter and the sensor data are then used to generate a low dimensional feature vector. This feature vector is able to distinguish between the true target and a false anomaly very effectively and can be used to train any number of machine learning algorithms for classification.

Training and learning a classifier off line saves computational resources which can instead be directed towards feature extraction during a real time mission. Current research is directed towards optimizing the particle filter routine or finding other less computationally intensive methods for incorporating temporal smoothness and consistency into the feature vector. The high classification accuracies are very encouraging and efforts to find other features which may increase the accuracy even further are also being investigated.

## VII. Acknowledgements

This work is sponsored in part by the Washington Technology Center (WTC), the Osberg Family Trust Fellowship, and the Washington NASA Space Grant Consortium. Juris Vagners at the University of Washington also contributed to this work.

## References

[1]Lum, C. W., Rysdyk, R. T., and Pongpunwattana, A., "Occupancy Based Map Searching Using Heterogeneous Teams of Autonomous Vehicles," *Proceedings of the 2006 Guidance, Navigation, and Control Conference*, Autonomous Flight Systems Laboratory, Keystone, CO, August 2006.

[2]Schapire, R. E., "A Brief Introduction to Boosting," *IJCAI*, 1999.

[3]Viola, P. and Jones, M., "Rapid Object Detection using a Boosted Cascade of Simple Features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1, 2001.

[4]Lester, J., Choudhury, T., Kern, N., Borriello, G., and Hannaford, B., "A Hybrid Discriminative/Generative Approach for Modeling Human Activities," *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[5]Raj, A., Subramanya, A., Fox, D., and Bilmes, J., "Rao-Blackwellized Particle Filters for Recognizing Activities and Spatial Context from Wearable Sensors," *Experimental Robotics: The 10th International Symposium, Springer Tracts in Advanced Robotics*, 2006.

[6]Fox, D., Hightower, J., Liao, L., and Schulz, D., "Bayesian Filtering for Location Estimation," *IEEE Pervasive Computing*, July 2003, pp. 23–33.

[7]Lum, C. W., Rysdyk, R. T., and Pongpunwattana, A., "Autonomous Airborne Geomagnetic Surveying and Target Identification," *Proceedings of the 2005 Infotech@Aerospace Conference*, Autonomous Flight Systems Laboratory, Arlington, VA, September 2005.

[8]NIMA, "National Imagery and Mapping Agency Technical Repport TR8350.2: Department of Defense World Geodetic System 1984, Its Definition and Relatinships with Local Geodetic Systems, 3rd Edition," Tech. rep., National Imagery and Mapping Agency, 2000.

[9]Dransfield, M., Christensen, A., and Liu, G., "Airborne Vector Magnetics Mapping of Remanently Magnetised Banded Iron-Formations at Rocklea, Western Australia," *Proceedings of the ASEG 16th Geophysical Conference and Exhibition*, Adelaide, Australia, February 2003.

[10]Kwok, C., Fox, D., and Meila, M., "Real-Time Particle Filters," *IEEE Special Issue on Sequential State Estimation*, 2004.

[11]Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics*, MIT Press, 2005.

[12]Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Haehnel, D., Rosenberg, C., Roy, N., Schulte, J., and Schulz, D., "Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva," *International Journal of Robotics Research*, 2000.

[13]Ferris, B., Hahnel, D., and Fox, D., "Gaussian Processes for Signal Strength-Based Location Estimation," *Proceedings of Robotics: Science and Systems*, 2006.

[14]Witten, I. H. and Frank, E., *Data Mining*, Morgan Kaufmann Publishers, San Francisco, CA, 2nd ed., 2005.