SENSITIVITY TO NOISE IN PARTICLE FILTERS FOR 2-D TRACKING ALGORITHMS [*]

DONG-HYEON PARK, STEPHANIE PORTER[†], SARAH WARKENTIN, AND FACULTY ADVISORS: ERIN BYRNE AND RACHEL LEVY

Abstract. A particle filter algorithm was used to simulate a Remotely Operated underwater Vehicle (ROV) tracking a moving target in 2-D space. The simulation modeled the behavior of a Sea Perch ROV modified with mounted cameras to perform blob-tracking on the target. Thirteen different noise levels were sampled for both distance and angle, with 100 trials per noise level. The angular noise demonstrated an exponential effect on the performance of the particle filter algorithm, while distance noise had minimal impact on the accuracy of the tracking.

Key words. particle filter, remotely operated underwater vehicles

AMS subject classifications. 60G, 65C, 93A

1. Introduction.

A Remotely Operated underwater Vehicle (ROV) is a robot used to navigate and explore aquatic environments. A popular problem in robotics is that of autonomous tracking, particularly involving ROVs. The process of tracking fish or sharks, for example, is greatly improved in convenience and efficiency with the development of an autonomous robotic tracking systems. Note that such autonomous tracking system can also have wide range of military applications. For example, VideoRay Military ROVs are used for the surveillance of enemy ships or submarines on Maritime ISR (Intelligence, Surveillance, Reconnaissance) missions [1].

For simplicity we worked in two dimensions, using a floating aquatic vehicle to track a floating target that emits a signal. Various types of signal transmission may be used, including blob-tracking via cameras or hydrophone pingers that send signals through the water. Cameras are generally more equipped to detect the angular position of an object, while pingers provide better distance data. With any given signal method however, a certain level of noise is generated in the signal data. The problem lies in how to anticipate and handle the noise while attempting to track the target. A flowchart of the general tracking process using this equipment can be seen in Figure 1.1.

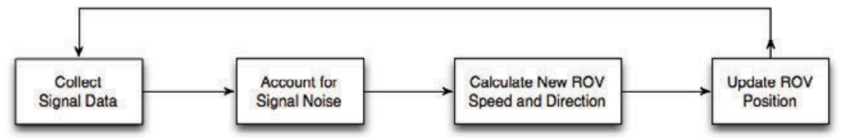

FIG. 1.1. *A flowchart of a general tracking algorithm using signal data and an ROV.*

We use a particle filter algorithm, described in Section 3, as one component of an ROV tracking system. This type of algorithm was chosen based on previous work conducted by Professor Christopher Clark at CalPoly, San Luis Obispo [2]. The particle filter uses signal data from receivers aboard the ROV to locate and track the

[*]Department of Mathematics, Harvey Mudd College, Claremont, California. This work has been conducted under the supervision of Profs. Erin Byrne and Rachel Levy.

[†]CORRESPONDING AUTHOR. QUESTIONS, COMMENTS AND SUGGESTIONS MAY BE SENT TO SPORTER@HMC.EDU.

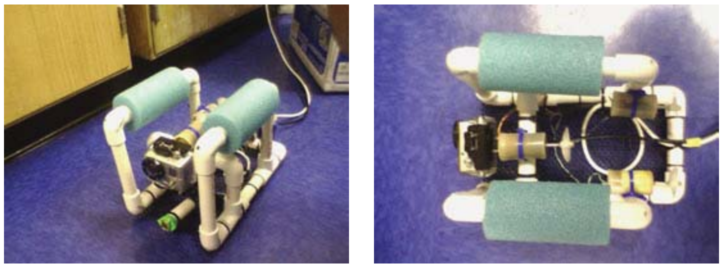

Fig. 2.1. *Configuration of the Sea Perch used in this paper. Picture on the right is a top-down view, which shows the position of the three motors and how they are oriented.*

Maneuver	Speed	Radius of Curvature
Forward	0.21 m/s	-
Right	0.44 rad/s	0.64 meters
Left	0.38 rad/s	0.46 meters
Sharp Right	0.23 rad/s	0.51 meters
Sharp Left	0.19 rad/s	0.46 meters
Backward	0.14 m/s	-

Table 2.1
Characteristics of Sea Perch's maneuvers

target. To get an accurate sense of how the particle filter algorithm will perform in a real-world situation, motion data of the Sea Perch ROV was gathered in a controlled environment. Then the particle filter algorithm was applied to a computer simulation of an ROV with movements characterized by this motion data. We will discuss both the positive and negative aspects of applying a particle filter to this problem, with a specific focus on how the algorithm handles noise to successfully direct the ROV to its desired target.

2. Sea Perch ROV. In order to simulate real-world behavior of an ROV, we focused on one specific ROV named Sea Perch. Sea Perch is a simple, inexpensive ROV developed by Massachusetts Institute of Technology Sea Grant (MITSG) College Program to promote STEM (Science, Technology, Engineering, Mathematics) education for high school students [3]. The basic frame of the Sea Perch is made of PVC pipes, and it has three propellers to help the ROV navigate underwater (See Figure 2.1). The propellers are positioned one on each side of the Sea Perch, and one at its center. All three propellers are oriented in the same direction so that the Sea Perch is propelled forward when all three are ON. Students often have the center propeller oriented vertically up or down to help the Sea Perch submerge, but we kept this propeller in same direction as the other two, as we are only interested in the two dimensional space. Each propeller can be set to either FORWARD or BACKWARD, so the side propellers are used to turn left or right, while the center propeller is used to move forward or backward. We also modified the Sea Perch to mount two cameras on its front; a GoPro HD camera and a small endoscope camera. Images from these two cameras are used to locate the target through blob-tracking software.

We have defined six distinct maneuvers for the Sea Perch: forward, normal right or left turns, sharp right or left turns, and backward. The normal turns can change Sea Perch's orientation quickly, but they have a large radius of curvature. On the other hand, sharp turns are slower but make smaller arcs as it turns. The long term behavior of each maneuver was observed in a controlled pool, with three trials per maneuver. The Sea Perch's movement was characterized as shown in Table 2.1. The positioning of the propellers have significant impact on the performance of the ROV, so the movement data of this particular configuration will not necessarily be valid for other Sea Perches. Note that on Table 2.1, the Left and Sharp Left turns had the same radius of curvature. This is due to the drag caused by the power cord interfering with the Sea Perch's movement. However, movement data is incorporated in the particle filter algorithm, and the tracker's movement within the algorithm will reflect the real-world movement of this particular configuration of the Sea Perch.

3. Particle Filters. A particle filter is an estimation model commonly used in tracking algorithms. Particle filters are best used for problems conducted in a nonlinear system, and work with either Gaussian or non-Gaussian noise. A typical particle filter algorithm generates a given number of "particles," each of which corresponds to a potential set of values for the set of variables under consideration. The particles are "filtered" by assigning each particle a weight based on its likelihood of containing the correct values. The algorithm then resamples from the set of particles with probabilities proportional to the assigned weights, generating a new set of particles. The process of resampling takes place after a specified amount of time (one time step) and then repeats itself for a given number of time steps. When the method is successful, the particles converge on the desired set of values.

In order to properly locate and follow the target in our problem, our ROV tracker incorporates a particle filter to determine the most likely location and velocity of the moving target. The process is aided by some form of signal data that indicates the relative angle of the target to the tracker, as well as the distance between the two. The algorithm then dictates how to move the tracker in such a way as to meet the target at some future step.

3.1. Comparison to Kalman Filters. Particle filters are often used as an alternative to Kalman filters, which estimate the parameters of unknown variables given a continuous input of noisy data. Kalman filters assume both a linear dynamical system and Gaussian distributions for error. This leads to poor tracking performance in environments which introduce nonlinear factors, such as occlusion or multipath propagation effects. Studies comparing the performance of Kalman and particle filter algorithms for tracking problems have demonstrated that Kalman filters are highly sensitive to levels of environment uniformity, while particle filters maintain consistent performance across varying scenarios [4]. While Kalman filters can be used in nonlinear systems, a particle filter is more likely to produce better results, since a particle filter makes no assumptions on linearity or uni-modal Gaussianess [4]-[6]. Due to the variable conditions in an underwater environment which have the potential of introducing nonlinear factors, we chose to implement a particle filter for our system.

3.2. Parameters. The tracking algorithm requires many parameter values to be determined empirically. The most important of these parameters are summarized in this section:

- **Time Step:** (denoted t) The algorithm is run in discrete time steps. Every time step, a new ping is received. The optimal time between pings must be determined empirically, but may be on the order of 5-10 seconds. The pings must be frequent enough such that the tracking algorithm receives enough data, but must also allow sufficient time for the ROV to respond at each time step. The time step is set to be 1 second in the current algorithm, for the sake of simplicity.

- **Tracker/Target speed:** (denoted m and s_s, respectively) These speeds will be determined by the number of position units that the tracker and target can move inbetween pings. Tracker speed is set to the experimentally determined values found in Table 2.1.
- **Number of particles:** A large number of particles increases robustness of the algorithm, but also increases runtime. Additionally, the estimation error of a particle filter decreases as the number of particles is increased (up to a threshold, as determined by the problem's other parameters) [7].
- **Noise parameters:** (denoted N_θ and N_v) The noise is modeled using normal random variables, one for distance and one for angle. These are included to increase the robustness of the algorithm, as described in Section 3.3. The algorithm has a parameter for the variance of each random variable. The effect of these noise parameters is discussed with the results of the simulation in Section 5.
- **Uncertainty parameters:** The weighting function used in the particle filter depends on the uncertainty in the predicted angle and distance. The uncertainty parameters are those inherent in the tracking equipment used by the ROV. These should therefore be as close as possible to the noise parameters, but in practice, these are properties of the tracking algorithm, and the noise parameters are meant to model reality. Thus, the algorithm should be tested for how much difference between noise and uncertainty parameters it can handle. The effect of these uncertainty parameters is illustrated in Figures 3.1 and 3.2. In Figure 3.1, the uncertainty is higher in the distance than the relative angle, so the particles converge more quickly to a particular relative angle than a particular distance. In Figure 3.2, the uncertainty is higher in the relative angle, so the opposite effect is seen. The particles are shown over several time steps, with the darker points indicating more recent particles.

3.3. Filtering Algorithm. Our simulation of the particle filter algorithm generates a randomly placed stationary or linear-trajectory target. It determines the approximate position of the target relative to the tracker (which starts at the origin) and then uses the aforementioned particle filter to converge on the correct position of the target at a rate which depends on the predicted error in the measurements.

The particle filter itself uses a set of weighted possible target states, which are referred to as "particles." A target state consists of (x, y, θ, v), where (x, y) is the position of the target as viewed from above, θ is its heading, and v is its speed in that direction. At each time step, the particle filter updates the set of particles in such a way that the particles will ideally converge on the actual state of the target, given the information gleaned from the sensors.

The particle filter algorithm consists of two steps: a prediction step and a correction step. The prediction step introduces some randomness into the heading and speed of the particle according to

$$\theta_{\text{new}} = \theta_{\text{old}} + N_\theta \tag{3.1}$$

$$v_{\text{temp}} = v_{\text{old}} + N_v, \tag{3.2}$$

where N_θ and N_v are normal random variables with mean 0. Their standard deviations are the uncertainty parameters described in the Section 3.2, and affect the convergence

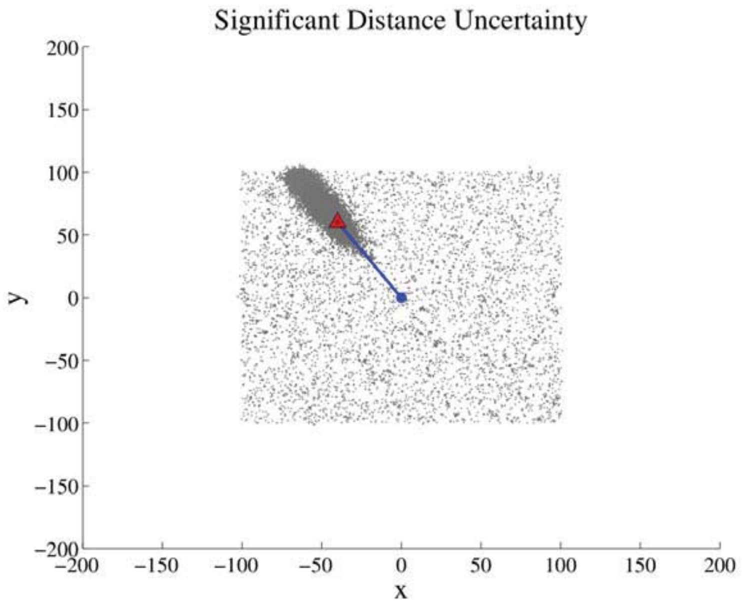

FIG. 3.1. *A time-collapsed figure of the tracking simulation under conditions of higher uncertainty in distance than relative angle. The trajectory of the simulated tracker is shown as the dark line (starting from the origin), and the target's position as the light triangle. The particles are shown as smaller points, progressing from grey to black with each time step. We can see that the particles converge more quickly in relative angle than distance. X and Y represent relative position in meters.*

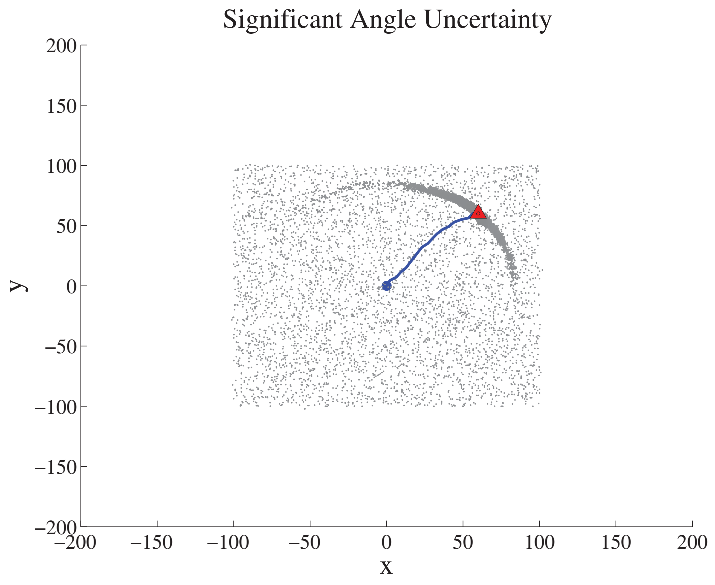

FIG. 3.2. *A time-collapsed figure of the tracking simulation under conditions of higher uncertainty in relative angle than distance. The trajectory of the simulated tracker is shown as the dark line (starting from the origin), and the target's position as the light triangle. The particles are shown as smaller points, progressing from grey to black with each time step. We can see that the particles converge more quickly in distance than relative angle. X and Y represent relative position in meters.*

of the particles in this algorithm. This randomness increases the robustness of the algorithm, as it helps to prevent the particles from converging on a local extremum. The algorithm then updates the position in each particle according to its heading and new speed:

$$x_{\text{new}} = x_{\text{old}} + v_{\text{temp}} * t * \cos\theta_{\text{new}} \tag{3.3}$$

$$y_{\text{new}} = y_{\text{old}} + v_{\text{temp}} * t * \sin\theta_{\text{new}}, \tag{3.4}$$

where t represents one time step. The speed is adjusted as a weighted average of its new speed and its speed in the previous time step:

$$v_{\text{new}} = \gamma * v_{\text{old}} + (1 - \gamma) * v_{\text{temp}}, \tag{3.5}$$

where γ is some quantity between 0 and 1. Finally, each particle is assigned a weight based on how close its coordinates are to the position predicted by the received signal data, which is computed according to certain signal strength functions described in Section 4. We explored a couple of possibilities for this weighting function, which are described in Section 3.4.

The correction step samples the set of particles with probabilities proportional to the weights assigned to them in the prediction step. These sampled particles are taken together with a small number of new, randomly generated particles to form the set of particles to be used in the next time step. The fresh, randomly-generated particles are introduced at each step as an additional effort to prevent the algorithm from converging on a local extremum, further improving the robustness of the algorithm.

After the new set of particles has been generated, the algorithm takes the weighted average of all particles to be the most likely position, heading, and velocity of the target. The weights used are the same weights stored in the particles themselves. The algorithm then predicts the distance between the tracker's current location and the future location of the target after one time step. Using this information, it updates the ROV's speed and heading to reach the target as quickly as possible, rather than simply driving toward its current location. The vector from tracker to target is (x, y) where

$$x = x_s + s_s \cos(\theta_s) - x_t \tag{3.6}$$

$$y = y_s + s_s \sin(\theta_s) - y_t. \tag{3.7}$$

Here, (x_s, y_s), θ_s, and s_s are the current predictions for position, heading, and speed of the target, and (x_t, y_t) is the current location of the tracker. The tracker's speed is then set to

$$\min\{\sqrt{x^2 + y^2}, m\}, \tag{3.8}$$

where m is the maximum tracker speed, and the heading is set to the angle between (x_t, y_t) and (x, y). In all simulations of the tracker's movement, tracker speed is taken to be consistent with the experimentally determined data found in Table 2.1. The transition time between any two ROV maneuvers is assumed to be negligible.

3.4. Weights. The way weights are assigned to particles is the most important process in the particle filter algorithm. The tracker should be able to determine the most likely position of the target using signal strength detected by the receiver, and then weight the particles according to how close they were to that predicted position. The tracking algorithm employs a Gaussian weighting function for relative angle and distance:

$$w = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{(S-p)^2}{2\sigma^2}}, \tag{3.9}$$

where w is the weight assigned to the particle, σ is the uncertainty parameter (described in Section 3.2) for either the distance or relative angle, S is the actual distance or relative angle predicted by the signal, and p is the distance or relative angle for that particular particle.

Given that the Gaussian distribution is not well-defined on a circle, the von Mises distribution is also good candidate in particle filters for problems involving the estimation of the probability distribution on a circle. This distribution is also known as the circular normal distribution, or Tikhonov distribution. It is defined on the range $w \in [0, 2\pi)$ with probability density function:

$$P(w) = \frac{e^{b\cos(w-a)}}{2\pi I_0(b)}, \tag{3.10}$$

where $I_0(x)$ is a Bessel function of order 0, $a \in [0, 2\pi)$ is the mean direction and $b > 0$ is a concentration parameter. The cumulative distribution function of the von Mises distribution is

$$D(w) = \frac{1}{2\pi I_0(b)} \left\{ wI_0(b) + 2\sum_{j=1}^{\infty} \frac{I_j(b)\sin[j(w-a)]}{j} \right\}, \tag{3.11}$$

which does not have a closed form. The von Mises distribution is the circular analog of the normal distribution with a mean of $\mu = a$ and circular variance $\sigma^2 = 1 - \frac{I_1(b)}{I_0(b)}$.

4. Signal Strength Functions. We assume that the signal amplitude decays exponentially with the distance between transmitter and receiver. The function we have currently implemented is

$$A = e^{-d} - \epsilon \tag{4.1}$$

where d is distance, A is amplitude, and ϵ is a small quantity to avoid division by zero in the inverse function. In Figure 4.1 we can see that as distance increases, the signal strength decays exponentially.

We also assume that the signal amplitude decays with angle in an approximately elliptical shape, based on the specifications for a similar transducer set[8]. The precise relationship between angle and signal strength is

$$A = \sqrt{\left(\frac{1}{2}\cos(\theta) + \frac{1}{2}\right)^2 + \left(b\sin(\theta)\right)^2}, \tag{4.2}$$

where θ is the angle, A is the amplitude, and b is a parameter determining the eccentricity of the ellipse which must be fit empirically. This relationship is shown in Figure 4.2. We can see that the signal strength decreases as the absolute value of relative angle increases.

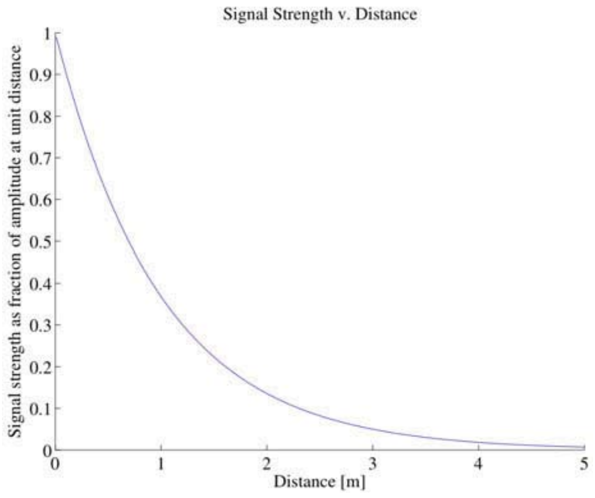

Fig. 4.1. *One example of a relationship between signal amplitude and distance between transmitter and receiver. As distance increases, signal strength decays exponentially.*

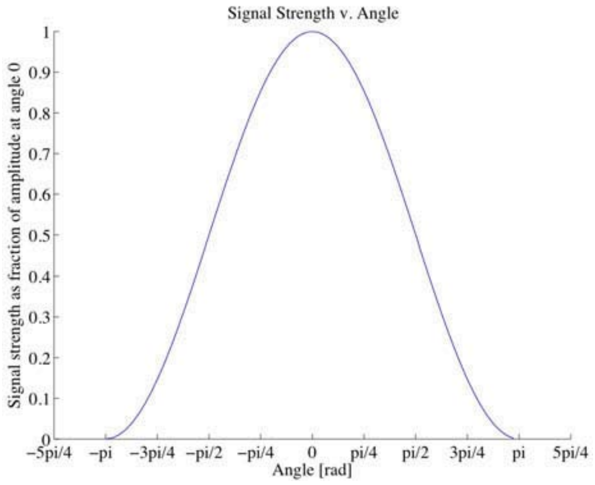

Fig. 4.2. *One example of a relationship between signal amplitude and relative angle from transmitter to receiver. Signal strength decreases as relative angle moves away from zero.*

5. Results. Sample output of the simulation with the particle filter is shown in Figures 5.1 - 5.3. In Figure 5.1, the stationary target is shown as the lighter triangle and the tracker's trajectory (starting from the origin) is shown as the darker line. The smaller black points represent the particles used in the particle filter, each of which corresponds to a possible target state. The particles converge on the location of the target, and the tracker drives directly to it.

In Figure 5.2, the target has a linear trajectory. As in the stationary case, the tracker succeeds in locating the target, though now it successfully predicts the targets motion so that it can follow the target.

In Figure 5.3, the effects of simulated noise are shown. As before, the tracker successfully tracks the target, though now with a slightly less direct path. We can

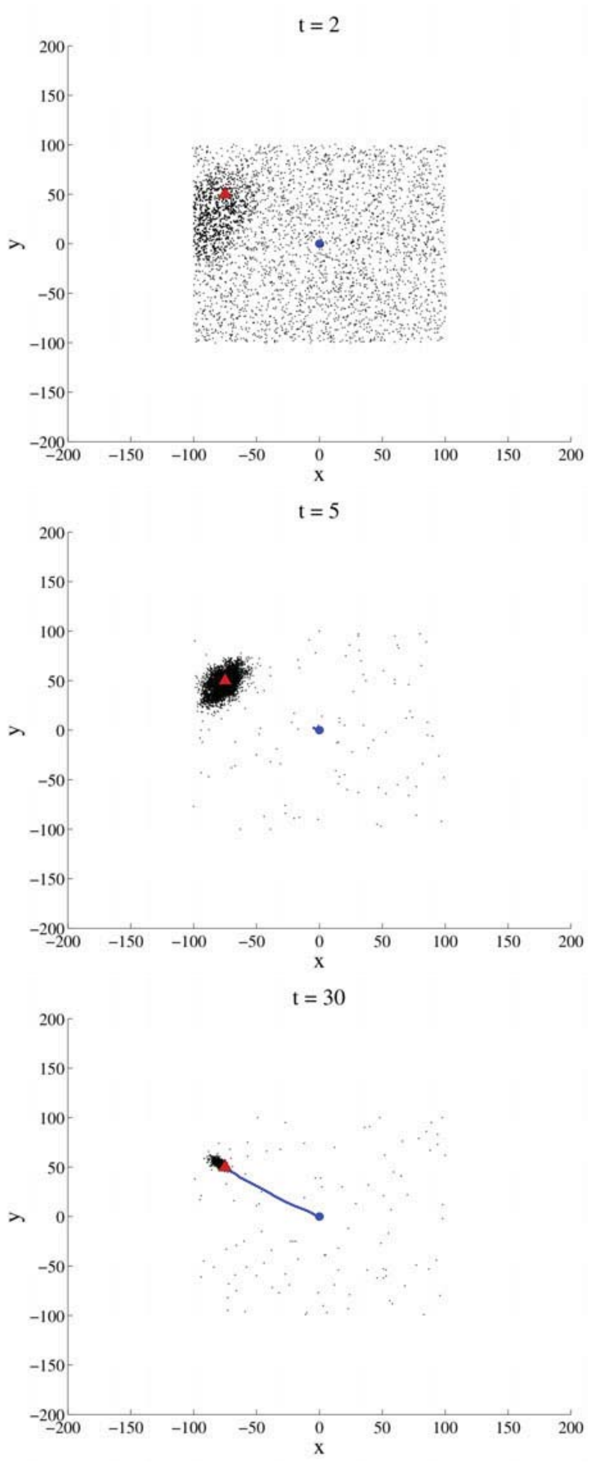

FIG. 5.1. *The output of the tracking simulation with a stationary target. The trajectory of the simulated tracker is shown as the dark line (starting from the origin), and the target's position as the light triangle. The particles are shown as smaller black points. The tracker is able to successfully locate the target. X and Y represent relative position in meters, and t is measured in seconds.*

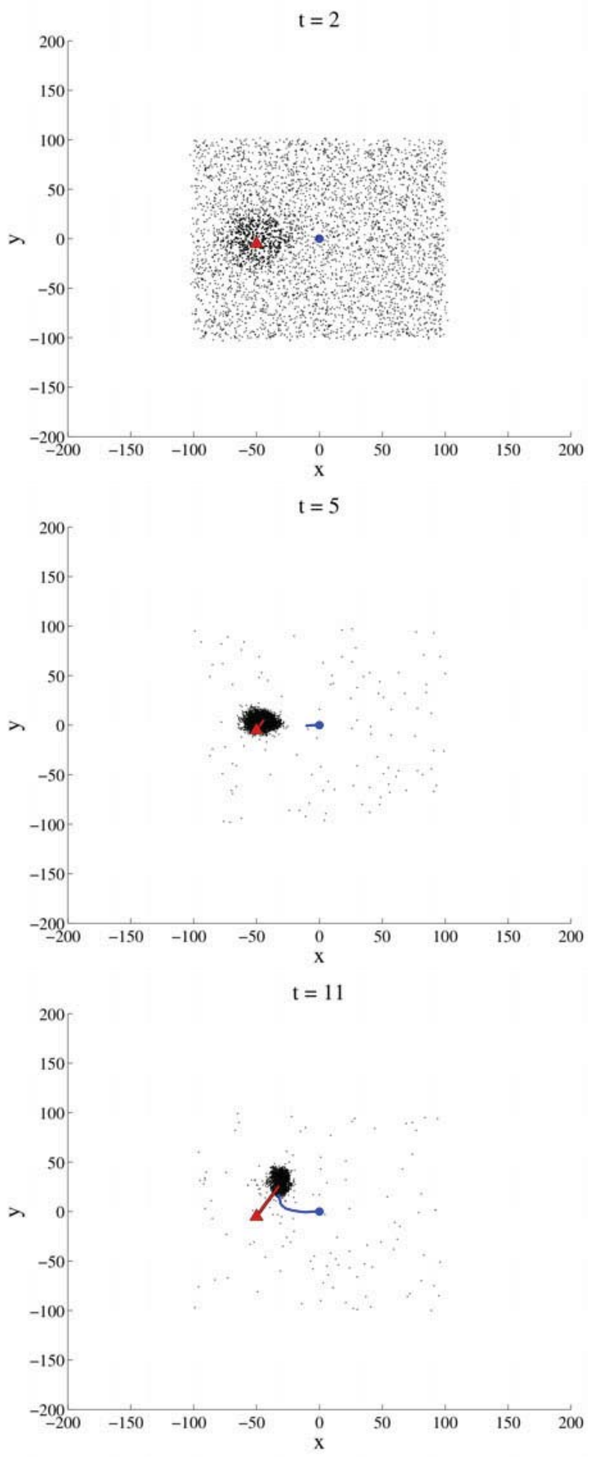

Fig. 5.2. *The output of the tracking simulation with a linear target trajectory. The trajectory of the simulated tracker is shown as the darker, curving line (starting from the origin), and the target's trajectory as the lighter, straight line (starting at the triangle). The particles are shown as the smaller black points. We can see that the tracker is able to follow a moving target. X and Y represent relative position in meters, and t is measured in seconds.*

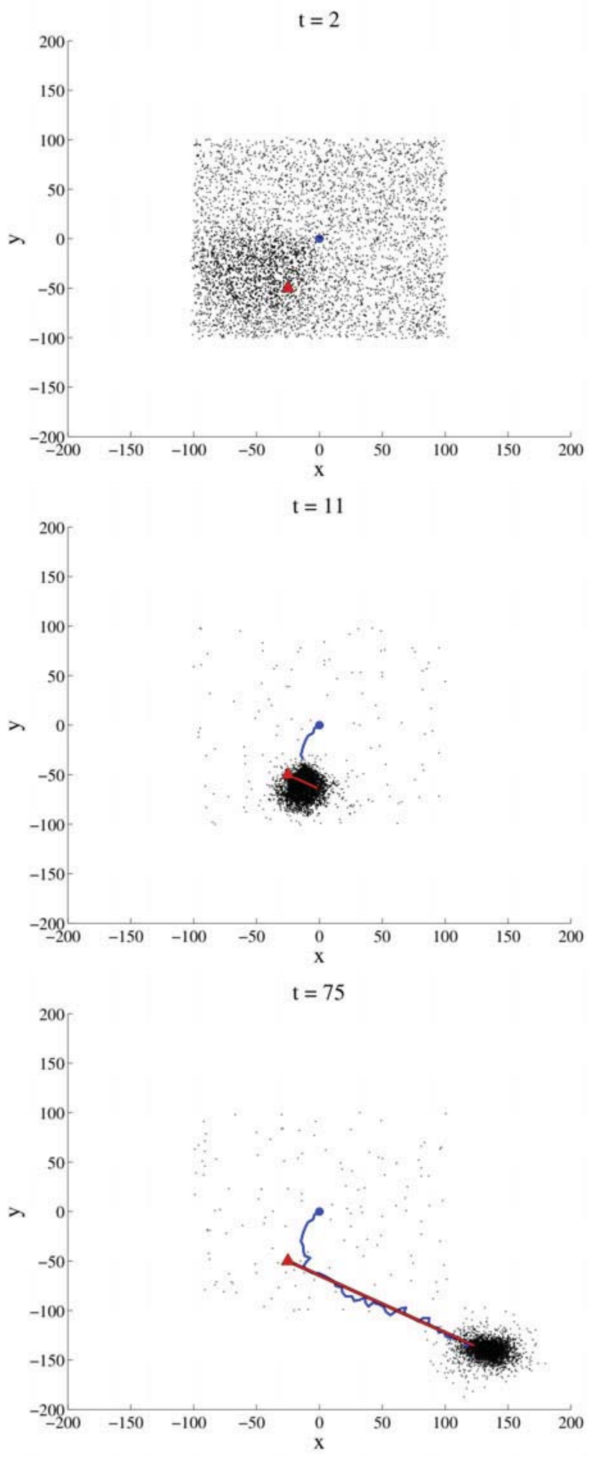

Fig. 5.3. *The output of the tracking simulation under conditions of simulated signal noise and with a linear target trajectory. The trajectory of the simulated tracker is shown as the darker, curving line (starting from the origin), and the target's trajectory as the lighter, straight line (starting from the triangle). The particles are shown as smaller black points. We can see that even in the presence of noise, the tracker is able to successfully follow the target. X and Y represent relative position in*

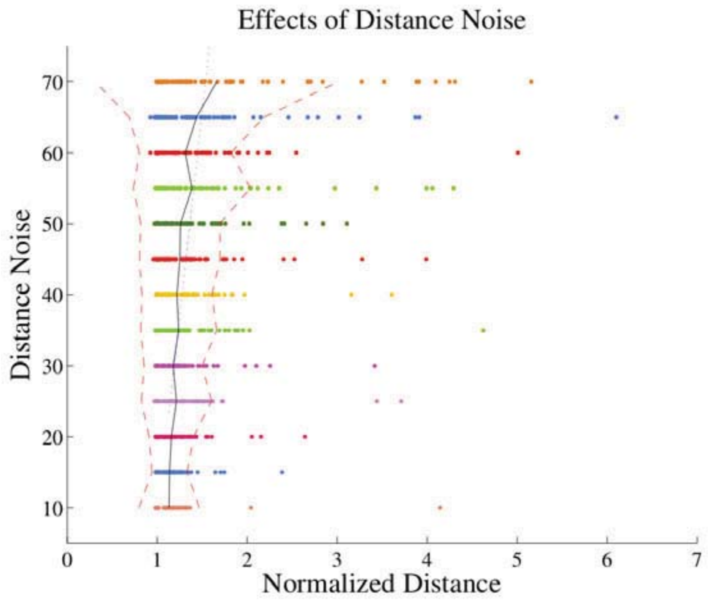

Fig. 5.4. *A representation of how varying noise levels for distance affect the distance the ROV must travel to reach a randomly-placed stationary target, normalized against the Euclidean distance between the ROV's origin and the target. Angle noise is fixed at 0, and distance is measured in meters. The solid line is a piece-wise connection between mean-value normalized distance travelled to reach stationary target at each given noise level. The dotted line is the fitted curve of these data points, and dashed lines represent one standard deviation on either side of the mean line (piece-wise). 100 trials were run for each noise level. It can be noted that increasing distance noise has minimal effect on the success of the ROV in locating and navigating to a stationary target.*

see that the particles begin to converge to a position slightly to the left of the target, but ultimately find the correct position and trajectory.

Figures 5.4 and 5.5 show how various noise parameters affect the success of the ROV in locating and navigating to a stationary target, according to signal strength functions (4.1) and (4.2). In both figures, either angle noise or distance noise is held at a constant of 0, while the other parameter is assigned values over a range ($\pi/63$ to $\pi/2$ for angle noise, and 5 to 70 meters for distance noise). The simulation is run for 100 trials for each given combination of noise values, plotting the normalized distance the ROV must travel to reach a randomly-placed target. The solid line represents piece-wise connection between mean-value normalized distance travelled to reach stationary target at each given noise level. The dotted line is the fitted curve of these data points, and dashed lines represent one standard deviation on either side of the mean line (piece-wise).

In Figure 5.4 we can see that increasing the amount of noise given in distance has minimal effect on the success of the ROV. However, Figure 5.5 demonstrates an exponential curve as angle noise is increased. It appears that the impact on the general success of the ROV tracking occurs at about $\pi/5$.

As previously stated, both hydrophone tracking and blob tracking are methods by which a signal can be relayed from target to tracker. Hydrophone tracking introduces a moderate amount of noise into both distance and angle parameters. Blob tracking features minimal angle noise but significant distance noise, as the size of the blob is

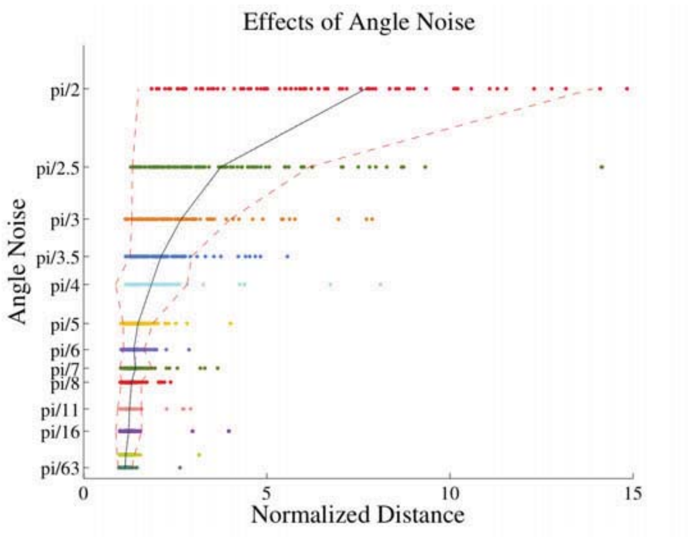

FIG. 5.5. *A representation of how varying noise levels for angle affect the distance the ROV must travel to reach a randomly-placed stationary target. normalized against the Euclidean distance between the ROV's origin and the target. Distance noise is fixed at 0, and angle is measured in radians. The solid line is a piece-wise connection between mean-value normalized distance travelled to reach stationary target at each given noise level. The dotted line is the fitted curve of these data points, and dashed lines represent one standard deviation on either side of the mean line (piece-wise). 100 trials were run for each noise level. It can be noted that increasing angle noise has an exponentially negative effect on the success of the ROV in locating and navigating to a stationary target. However, noise levels up to $\pi/5$ are more or less comparable in term of success.*

used to approximate how far away the target is. Considering that the particle filter appears to perform relatively consistently regardless of distance noise and is more affected by angle noise, blob tracking appears to be a preferable method for collecting position data from the target.

6. Conclusion. Remotely Operated underwater Vehicles (ROVs) have been highly successful in exploring aquatic environments and tracking fast-moving organisms such as sharks. A particle filter algorithm can be highly effective in such automated tracking operations. To investigate a particle filter's sensitivity to noise, we simulated an ROV tracking a moving object in two dimensional space at different distance and angle noise levels. The simulations demonstrated that angle noise has an exponential effect on the tracker's success in locating the target, while distance noise has a constant and therefore trivial effect. This suggests that a particle filter algorithm is well-suited for an ROV equipped with a video camera, as this can provide high accuracy with regards to the relative angular position of the target.

The next step of this research would be to implement the particle filter algorithm in a real-world ROV system. This should help us understand how various sensor noise or fluid drag can affect the performance of the algorithm. Other future work would be to investigate how the particle filter algorithm performs at different tracker speeds. Finding the relationship between tracker speed and the effect of angular noise can help determine the optimal sensor quality for a particular ROV.

7. Acknowledgements. The authors would like to thank the Office of Naval Research (ONR Grant Number N00014-10-1-0641) for the funding of this project. We wish to acknowledge the contributions of previous DYNAR teams. We would also like to thank our advisors, Professors Erin Byrne and Rachel Levy, for their guidance and help.

REFERENCES

[1] Example of military applications for ROVs, `http://www.videoray.com/applications/militaryapp/maritime-isr.html`, 2013.

[2] C. M. Clark. Teams from Cal Poly, CSU Long Beach track sharks using underwater robotics, `http://calpolynews.calpoly.edu/news_releases/2011/August/underwater.html`, 2011.

[3] Information on the Sea Perch ROV, `http://www.seaperch.org/`

[4] K. Lee, A. Oka, E. Pollakis, and L. Lampe. "A comparison between unscented Kalman filtering and particle filtering for RSSI-Based tracking," in *7th Workshop on Positioning Navigation and Communication (WPNC).* IEEE. March 2010, pp. 157-163.

[5] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, A Tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Bayesian Tracking, IEEE Transactions on Signal Processing, vol. 50, no. 2, pp. 174-188, Feb 2002.

[6] J. Carpenter, P. Clifford, and P. Fearnhead. An Improved Particle Filter for Non-linear Problems. *Proc. Inst. Elect: Eng., Radar, Sonar, Navig.,* 1999

[7] M. Bolic, S. Hong, and P. M. Djuric. Performance and Complexity Analysis of Adaptive Particle Filtering for Tracking Applications. Proceedings of the Asilomar Conference on Signals, Systems, and Computers, Asilomar, CA, 2002.

[8] Spec sheet for the ultrasonic transducers, `http://www3.hmc.edu/~watkins/classes/e155/projects2010/FlomStreshinsky.pdf`, 2010.

[9] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. Nordlund.