AA228/CS238 Final Report: Targeting Hostile Vehicle Modeled as a Partially Observable Marcov Decision Process with State-Dependent Observation Model

Joel Pazhayampallil Kyle Reinke

Fall Quarter 2014

Abstract

In this project, we simulate a hostile vehicle target with 2-dimensional continuous state which transitions from one discrete time-step to the next according to a known probabilistic model. The decision-making agent is described by a discrete 2-dimensional position which transitions state deterministically according to a discrete action model. The partially observable nature of the target is captured by a belief state over possible states. The probability model for the observations is defined as a function of the belief-state and deterministic agent state. The agent is only allowed to shoot at the target one time. A reward function was defined to capture the need to eliminate the target quickly, but also the need to establish a good enough estimate of the position of the target before shooting, lest the target survives.

Four such models are investigated to determine the effectiveness of the employed Partially Observable Markov Decision Process (POMDP) solution method. This project implements the Monte-Carlo Real-Time Belief Space Search method for online solution of the presented POMDP.

1 Introduction

There exists a large set of real-world problems in which an agent begins with good information about how the world works, but with very little information about the current state of the world. In order to perform a task, the agent must spend effort to both gather information about the world, and attempt to complete the goal task using gathered information. Ideally, a rational agent will optimally balance localization effort against the effort used to complete the task. Meaning the agent does not gather more information than necessary to perform satisfactorily, but also does not attempt the goal task if the risk of failure is unacceptably high. One such example would be that of a defensive agent that must engage a hostile vehicle encroaching on the border.

In order to effectively localize the target, observations that correlate with the state of the target must be made. In many cases the observations change depending on the state. The sensors may be obstructed in some states, or may be only reliably accurate in one axis in the agent's body-fixed frame. Thus, certain states may be beneficial for more rapidly localizing the target, but may be states that are counter-productive to directly achieving the goal.

2 Detailed Problem Description

Four distinct scenarios of the general problem were investigated in this project. Each problem setup can be described by the transition model, observation model, and reward function. In each of the scenarios, the agent has 6 actions: move ± 5 units along each axis, no movement, or engage the target. The agent moves deterministically, but the target position moves stochastically according to a known velocity model.

In the scenarios presented, the true target position moves sinusoidally between (10,-5) and (10,5). There are costs incurred at each time step and for movement. Additionally, engaging the target will end the game but will cost the infinite discounted sum of the time step cost multiplied by the probability of missing the target. In summary, the agent prefers engaging the target quickly to end the game and minimizing the probability of missing the target. In the sections below, the following conventions are used: x =agent position, s =target position, t =current time, a =action, Δt =time step, o =observation, $b = \{s_1 ... s_{N_p}\}$ = belief state represented by states of each particle.

2.1 Sensor with poor lateral localization and point target reward model

In this scenario the agent can localize the target well in the longitudinal direction but not in the lateral direction, relative to the agent. The reward model requires the target to be within a certain radius of the mean of the belief state.

(a) Transition model: The belief state transitions by integrating a known target velocity with additional noise. The target motion is independent of the agent's actions and the agent's motions are deterministic, so the agent's transition is handled separately from the target belief state transition.

Target transition

$\sigma = \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$

$v \sim \mathcal{N}(nominalVelocity(t), \sigma^2)$

$T(s, a) = v\Delta t$

Agent transition

$$T(x, a) = \begin{cases} x + \begin{bmatrix} 5 \\ 0 \end{bmatrix} & : a = \text{move right} \\ x + \begin{bmatrix} -5 \\ 0 \end{bmatrix} & : a = \text{move left} \\ x + \begin{bmatrix} 0 \\ 5 \end{bmatrix} & : a = \text{move up} \\ x + \begin{bmatrix} 0 \\ -5 \end{bmatrix} & : a = \text{move down} \\ x & : a = \text{no movement or engage} \end{cases}$$

(b) Observation model: The sensor is modeled as a Gaussian distribution with low variance in the direction along the vector to the target and high variance perpendicular direction.

$\Sigma = \begin{bmatrix} 0.1 & 0 \\ 0 & 5 \end{bmatrix}$

$\theta = \text{atan2}(s_y - x_y, s_x - x_x)$

$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

$\Sigma_R = R\Sigma R^T$

$O(o|s) = \mathcal{N}(o|s, \Sigma_R^2)$

(c) Reward model: The agent receives a penalty at each time step to encourage the agent to quickly engage the target. An additional small penalty is assigned for movement. Engaging the target will end the game, but missing the target will penalize the agent by the infinite discounted sum of the time step penalty. This is to capture the fact that the agent is limited to engaging only once and the target will remain indefinitely if the agent misses. In this case engaging the target will center the shot on the mean belief state of the target with the probability of success given by the proportion of the belief state within a radius of the shot.

$\gamma = 0.999$

$c_t = -100$: time penalty

$c_m = -1$: movement penalty

$r = 1$: hit radius

m_b : mean belief state

ρ : proportion of belief states enclosed within r of m_b

$$R(b, a) = \begin{cases} c_t & : a = \text{no movement} \\ c_t + c_m & : a = \text{any movement} \\ c_t + \frac{1}{1-\gamma}\rho & : a = \text{engage} \end{cases}$$

2.2 Sensor with poor lateral localization and angle target reward model

In this scenario the agent can localize the target well in the longitudinal direction but not in the lateral direction, relative to the agent as in section 2.1. The reward model requires the target to be within a certain angular distance of the mean of the belief state.

(a) Transition model: Identical to section 2.1

(b) Observation model: Identical to section 2.1

(c) Reward model: Identical to section 2.1 except that in this case engaging the target will aim the angle of the shot on the mean belief state of the target with the probability of success given by the proportion of the belief state within an angular distance of the shot.

$\gamma = 0.999$
$c_t = -100$: time penalty
$c_m = -1$: movement penalty
$\theta_{shot} = 1^\circ$: hit angle
m_b : mean belief state
ρ : proportion of belief states enclosed within θ_{shot} of m_b

$$R(b,a) = \begin{cases} c_t & : a = \text{no movement} \\ c_t + c_m & : a = \text{any movement} \\ c_t + \frac{1}{1-\gamma}\rho & : a = \text{engage} \end{cases}$$

2.3 Sensor with uniform localization, angle target reward model, and high noise in target transition model

In this scenario the agent can localize the target equally well in the lateral and longitudinal directions. The target transition model has high noise in the direction of the target's motion and low noise in the perpendicular direction. The reward model requires the target to be within a certain angular distance of the mean of the belief state as in 2.2.

(a) Transition model: Identical to section 2.1 except that in this case there is high noise in the target velocity.

Target transition
$\sigma = \begin{bmatrix} 0.15 & 0 \\ 0 & 7.5 \end{bmatrix}$
$v \sim \mathcal{N}(nominalVelocity(t), \sigma^2)$
$T(s,a) = v\Delta t$

(b) Observation model: The sensor model has uniform variance in both directions.

$\Sigma = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$
$O(o|s) = \mathcal{N}(o|s, \Sigma^2)$

(c) Reward model: Identical to section 2.2

2.4 Sensor with uniform localization, point target reward model, and agent state dependent observation model

In this scenario the agent can localize the target equally well in the lateral and longitudinal directions, but the sensor noise is dependent on the state of the agent. Sensor noise increases with the agent's euclidean distance from a given "vantage point". The reward model requires the target to be within a certain radius of the mean of the belief state as in section 2.1.

(a) Transition model: Identical to section 2.1

(b) Observation model: The sensor model has uniform variance in both directions, but the variance increases with the agent's distance from a given "vantage point".

$p_v = (-5, 10)$: Vantage point
$\Sigma = \|x - p_v\| \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$
$O(o|s) = \mathcal{N}(o|s, \Sigma^2)$

(c) Reward model: Identical to section 2.1

3 Solution Method

The overarching high level solution method implemented is Monte-Carlo Real-Time Belief Space Search (MC-RTBSS), similar to that described in Wolf[3]. The algorithms are described in the following sections.

3.1 Particle Filter

The continuous nature of the target's states and non linear-Gaussian assumptions renders the state-space directly unsearchable, and the belief-space impractical to represent exactly. A particle model was chosen in order to discretely approximate the belief-state and reduce the continuous search space into a discretely searchable space.

At each time-step forward of the simulation the particles are propagated through the known stochastic transition model. An observation of the true underlying state is made, and each propagated particle is weighted proportionally to its new observation model. Once all weights are assigned, a new particle representation of the belief-state is sampled from the weighted particles. The process is summarized in Algorithm 1.

Algorithm 1 Updates belief state given an observation; returns an updated belief state

function UPDATEBELIEF(b, o)
  for state in b **do**
    $w_{\text{state}} \leftarrow O\,(o|\text{state})$
  end for
  $b' \leftarrow$ sample N_p particles from weighted b
return b'
end function

3.2 Projection of Belief State into Future

To project the belief state into the future for rational planning based on expected future outcomes, first the particles from the current leaf are transitioned according to the transition model. However, due to the partially observable nature of the problem, the resulting belief state will change depending on the observation received after the transition. The observation space is also continuous in this problem, so to expand and search all possible observations to determine an expected utility over all resulting belief states is infeasible. Again we will rely on weighted sampling N_o times to approximate the search over the observation space. Thus each time that the belief state is projected the search tree grows N_o new branches. The function that projects belief is outlined in Algorithm 2.

Algorithm 2 Projects the belief state into the next time step for a number of samples; returns set of projected next updated belief states

function ProjectBelief(b, a)
  $b' \leftarrow T(b, a)$
  for i in $1 : N_o$ **do**
    $s_{o_i} \leftarrow$ sample a particle from b
    $s'_{o_i} \leftarrow T(s_{o_i}, a)$
    $o_i \leftarrow$ sample observation from $O\left(o|s'_{o_i}\right)$
    $b'_{o_1} \leftarrow$ UpdateBelief(b', o_i)
  end for
  return $\left\{b'_{o_1}, b'_{o_2}, ..., b'_{o_{N_o}}\right\}$
end function

3.3 Search Tree Expansion: Branch and Bound

Now that we have introduced the forward projection of the belief state through the branching of a tree over observation samples, we can come up with a recursive expansion scheme that can search all sampled belief states. At depth 1 we first expand over the finite discrete action space to search which action will return the best result up to the search depth and quality of the heuristic at depth 0. Projecting forward, we must follow the projection algorithm shown in Algorithm 2 for each action. Thus at each depth the search tree will expand by a factor of $N_a \cdot N_o$, bringing the total complexity of calculation to the order of $(N_p + N_a) \cdot (N_a \cdot N_o)^D$.

With such a rapid expansion of complexity and the large number of particles required for reliable results, the ability to prune the search tree is extremely valuable in order to increase the search depth. Search depth is very important to the proposed problem, because without sufficient depth the algorithm will not be able to see ahead to realize that sacrificing an immediate increase in reward could result in a long-term increase in reward. In other words, our algorithm will become too greedy.

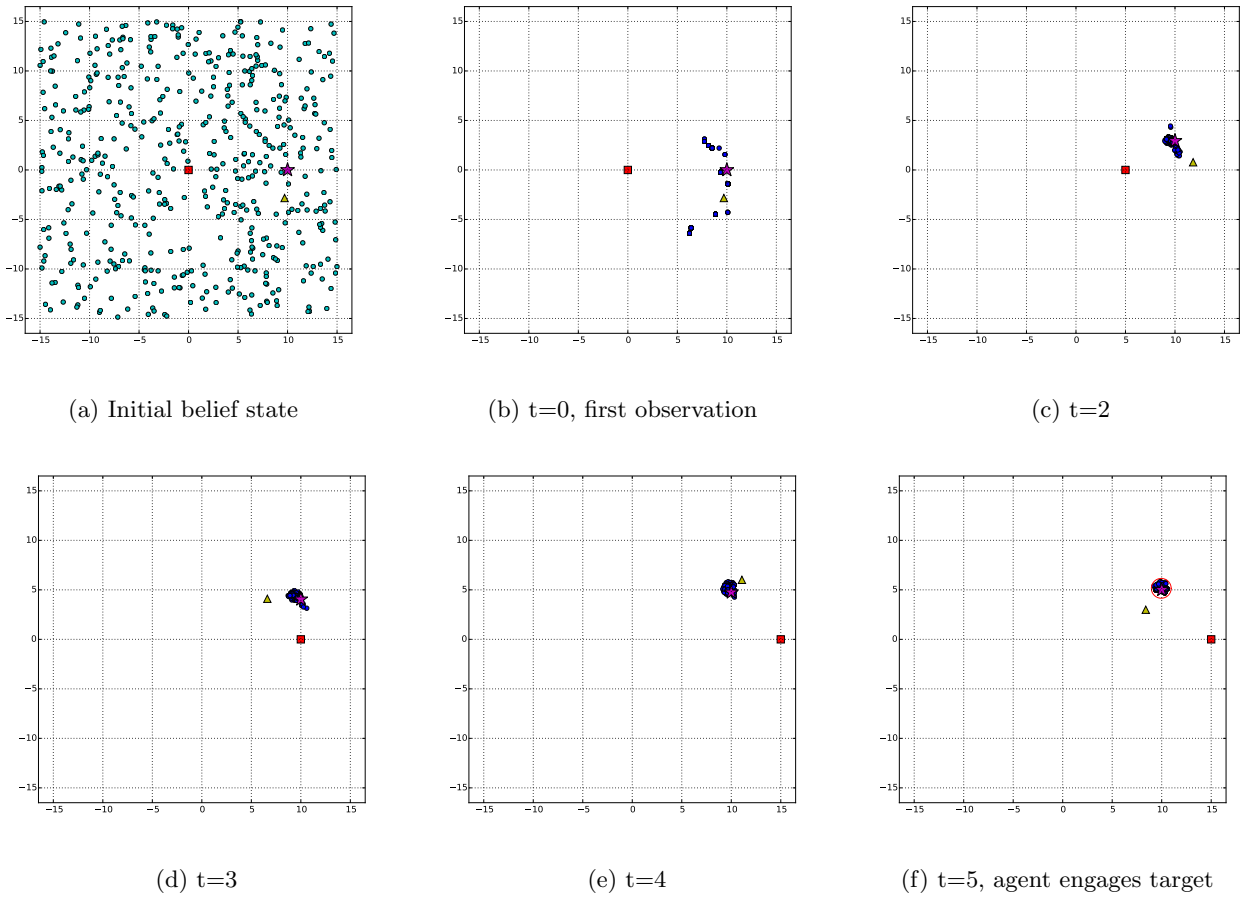
In order to safely prune branches from the search tree, a branch-and-bound technique was implemented. In this technique each expanding node first calculates a heuristic that represents the upper bound for the expected utility of it's entire sub-tree. In our case, the rewards are all monotonic negative, which means that a consistent upper bound on the reward is the immediate reward of the current belief-action pair because the total reward cannot increase as depth increases. This upper bound is then compared with the lower bound of the best expected reward as returned upwards from its sister trees. In our case, when the depth drops to 0 through recursion, there is a known reward possible at the end of the game because it will end when the agent chooses to shoot. A better solution may exist further down the tree, but we are guaranteed to at least be able to choose this zero-depth reward in the worst case. If the best possible reward of an unexpanded sub-tree is worse than the score we already guaranteed from a sister tree, then there is no reason to expand that tree, and we can save the computation time. The expanding and pruning algorithm implemented is illustrated in Algorithm 3

Algorithm 3 Expands the search tree at each depth D; returns the lower bound on largest reward for current sub-tree

function ExpandTree(b, D)
  if $D = 0$ **then**
    return $L(b)$
  else
    Queue actions in order of decreasing $U(b, a_i)$
    while $U(b, a_i) > L_T$ and (Action queue is not empty) **do**
      $a_i \leftarrow$ Dequeue action queue
      $\sigma \leftarrow 0$
      for b'_{o_i} in ProjectBelief(b, a_i) **do**
        $\sigma \leftarrow \sigma +$ ExpandTree($b'_{o_i}, D-1$)
      end for
      $L_T(a_i) \leftarrow$RewardModel(b, a_i)$+\gamma \frac{\sigma}{N_o}$
      if $L_T(a_i) > L_T$ **then**
        $a^\star \leftarrow a_i$
        $L_T \leftarrow L_T(a_i)$
      end if
    end while
  end if
  return L_T
end function

3.4 Top Level Simulation

With a POMDP solution method in place, a simple simulation structure was implemented to test the solver. The simulation simply makes an observation of the true state and applies the observation update model to the uniform initialization. Next the POMDP solver runs the ExpandTree function for the desired depth. The best action is returned and executed. Then the simulation time is ticked forward, the true state and belief states are transitioned through the transition model. A new observation is sampled from using the true state as a given, and the observation is used to again update the belief. This cycle continues until the game ends. The following parameters were used in the simulation.
$D = 2; N_p = 500; N_o = 3; N_a = 6$



(a) Initial belief state (b) t=0, first observation (c) t=2

(d) t=3 (e) t=4 (f) t=5, agent engages target

Figure 1: Progression of MC-RTBSS algorithm for the case where there is high lateral variance in the observation measurement, and the shooting mechanism is a point target. The agent state is the red square, the current observation is the yellow triangle, the true target position is the magenta star, the belief state particles are the blue circles, and the shooting mechanism is the red circle.

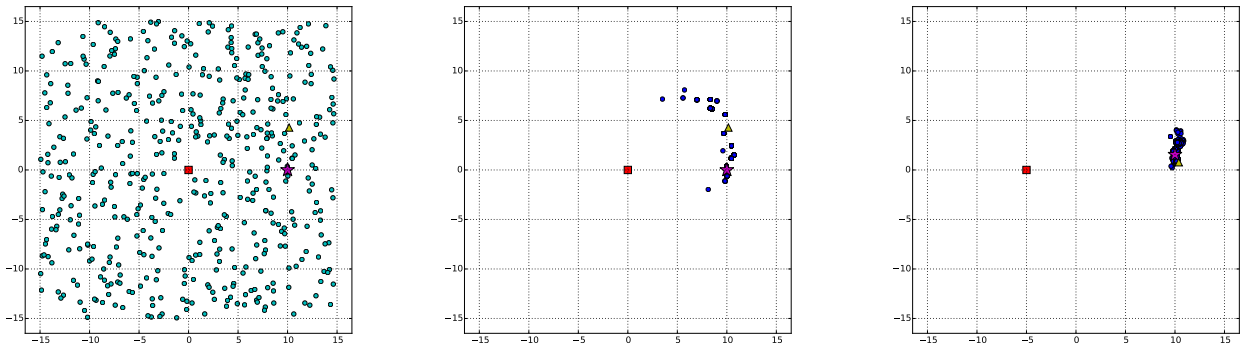
4 Results

4.1 High Observation Noise in Lateral Axis: Point Target

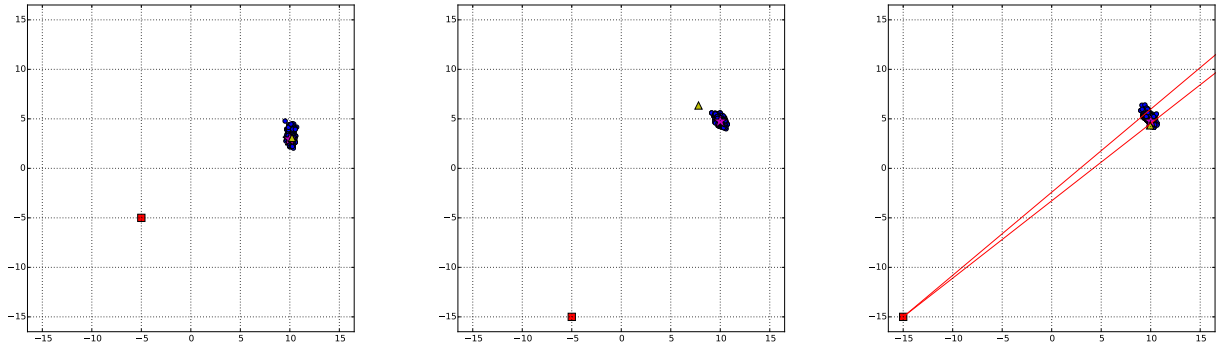
In this model we make the assumption that the sensor available to the agent is capable of precisely measuring in the longitudinal direction, along the displacement vector between the particle and the agent position. However, the measurement is quite noisy in the lateral direction. The results are shown graphically in Figure 1. At the first decision point, the signal is very noisy in the direction lateral to the particles. The algorithm determines that by measuring at different angles it can improve localization of the hostile vehicle. The agent continues to move around the target until there is no useful reduction of state uncertainty by moving or waiting available within the next D time steps. Thus the agent chooses to shoot, and successfully hits the target.

4.2 High Observation Noise in Lateral Axis: Angle Target

We make the assumption in this model that the sensor available to the agent is capable of precisely measuring in the longitudinal direction, along the displacement vector between the particle and the agent position. However, the measurement is quite noisy in the lateral direction. The results are shown graphically in Figure 2. At the first decision point, the signal is very noisy in the direction lateral to the particles. The algorithm determines that by measuring at different angles it can improve localization of the hostile vehicle. Additionally by moving away from the vehicle the beam width at target intersection is much larger. The moves mostly down at first, as this is the most effective way to change measurement angle, which will most rapidly reduce the uncertainty of the belief state. Once the agent reaches the game boundary in



(a) Initial belief state (b) A t=0, first observation (c) t=1



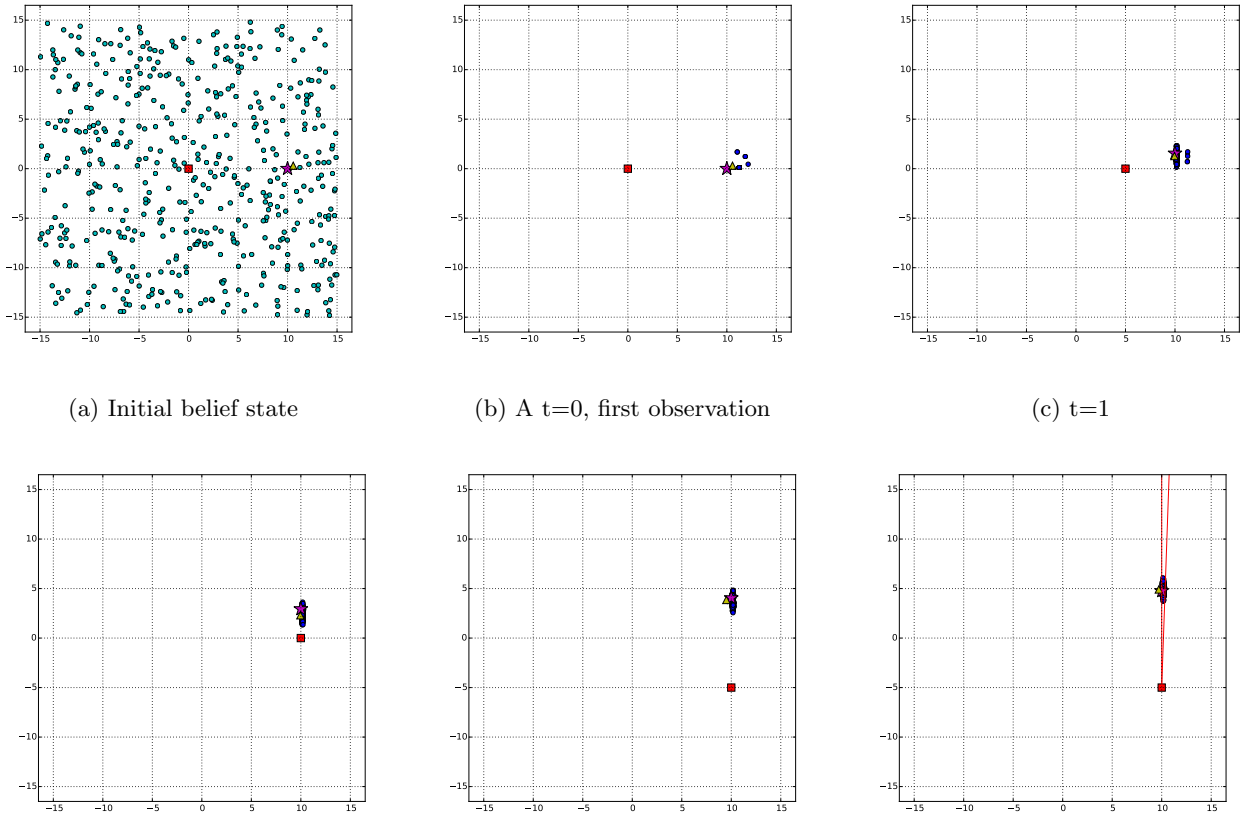
(d) t=2 (e) t=4 (f) t=6, agent engages target

Figure 2: Progression of MC-RTBSS algorithm for the case where there is high lateral variance in the observation measurement, and the shooting mechanism is an angle target. The agent state is the red square, the current observation is the yellow triangle, the true target position is the magenta star, the belief state particles are the blue circles, and the shooting mechanism is the red beam.

the vertical axis, it resumes moving away to the left. Once it is as far from the target as possible, the algorithm returns that there is no useful reduction of state uncertainty or increase of beam width at target intersection within the next D time steps. Thus the agent chooses to shoot, and successfully hits the target.

4.3 High Transition Velocity Noise: Angle Target

In this model we assumed that the hostile vehicle is not able to control the magnitude of its velocity very precisely. Thus, as particles are transitioned or projected forward, the position uncertainty is much larger in the direction of travel. The results are shown in the plots of Figure 3. In this case, the agent begins by moving to the right. This is because the reward function will most quickly increase by reducing the angle between the displacement vector and the major principle axis of the velocity noise. Once the agent is directly down from the target vehicle, it moves further downward. The algorithm is able to determine that the total game reward can be further increased by using another time step to reduce the angular spread of the target, and thus increase the proportion of belief states covered by the beam. After that final adjustment, the algorithm does not detect any improvement in chance to hit the target that will be worth the additional time cost. In this example the agent shoots and hits the target.



(a) Initial belief state

(b) A t=0, first observation

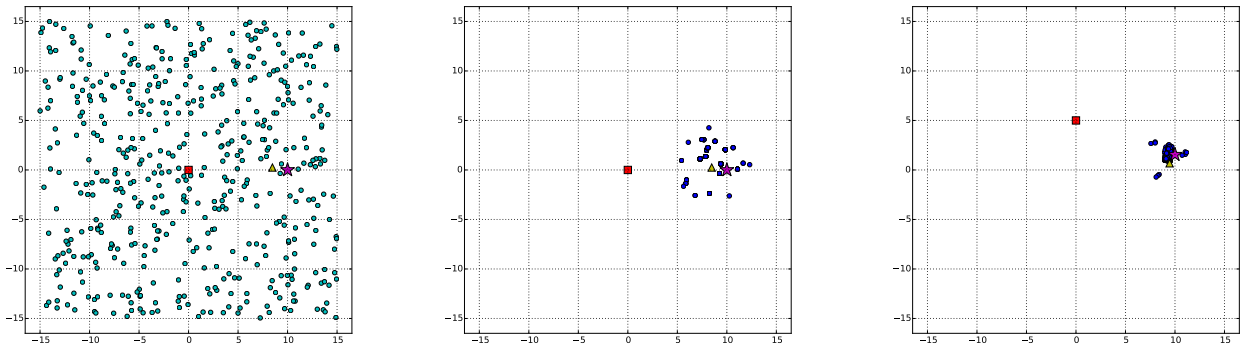
(c) t=1

(d) t=2

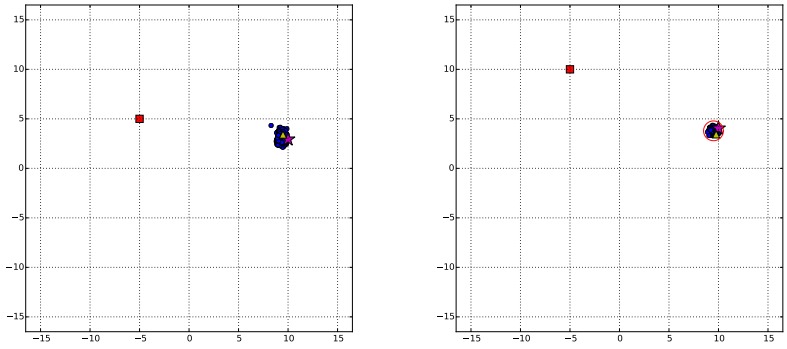
(e) t=3

(f) t=4, agent egages target

Figure 3: Progression of MC-RTBSS algorithm for the case where there is high velocity noise in the state transition function of the hostile vehicle, and the shooting mechanism is an angle target. The agent state is the red square, the current observation is the yellow triangle, the true target position is the magenta star, the belief state particles are the blue circles, and the shooting mechanism is the red beam.



(a) Initial belief state (b) A t=0, first observation (c) t=1



(d) t=2 (e) t=3, agent engages target

Figure 4: Progression of MC-RTBSS algorithm for the case where an improved vantage point exists at $(-5, 10)$, and the shooting mechanism is a point target. The agent state is the red square, the current observation is the yellow triangle, the true target position is the magenta star, the belief state particles are the blue circles, and the shooting mechanism is the red circle.

4.4 Fixed Improved Vantage Point Model: Point Target

In this model we had assumed that there is a fixed location that gives the smallest variance in the observation model, and that the variance increases as the agent is farther away from that location. The case presented has the improved vantage point at $(-5, 10)$. The visualization of the results can be seen in Figure 4. In this case the MC-RTBSS algorithm chooses to move toward the vantage point because the uncertainty of the target's location can only be reduced in this way. The algorithm has computed that the time cost taken to move towards the vantage point will result in a higher score at the end of the game. This is the behavior that one would expect from an agent behaving rationally in this situation. As soon as the cost of waiting or moving to improve localization cannot be offset by the expected reward of hitting the target within depth D time steps, the algorithm shoots. In this example the agent has succeeded in hitting the true target.

5 Conclusion

While in these simple test cases the MC-RTBSS algorithm performs satisfactorily with a lookahead depth of 2, more interesting cases exist where this was not sufficient. However, if the lookahead were increased sufficiently to capture the proper behavior, the algorithm run time was orders of magnitude too slow for online use.

The bounds used in Algorithm 3 to prune the search tree could be made to be more tight in order to prune even more of the tree. In fact, it may be beneficial to employ a non-consistent heuristic that will further prune the tree at the expense of an optimality guarantee. Practically, with a good heuristic the result will be only slightly suboptimal over the full tree, while the increased tree depth will reveal higher rewards that lead to more sophisticated behaviors.

Another improvement that may contribute to more efficient updates of the belief state is a smart dynamic resampling scheme. When the belief state is sparse, more particles are necessary to accurately represent it. However, once the belief state is concentrated, less particles are necessary. If the number of particles is dynamically selected to produce an optimal particle density, then a large increase in computation time could be gained at a negligible degradation in accuracy of the belief state.

6 Acknowledgements

Additional resources used in developing the accompanying Julia Language implementation of the methods described are Decision Making Under Uncertainty,by Kochenderfer [1], and Probabilistic Robotics, by Thrun [2]

References

[1] Mykel Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. The MIT Press, 1 edition, 2014. Pre-publishing printing for Stanford course.

[2] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[3] Travis B. Wolf and Mykel J. Kochenderfer. Aircraft collision avoidance using Monte Carlo real-time belief space search. *Journal of Intelligent and Robotic Systems*, 64(2):277–298, 2011.