Multi-Agent Intercept

Justice Sefas

April 18, 2019

1 Introduction

Large numbers of threats homing in on an asset pose coordination problems for defending agents, especially under degraded sensing conditions. Relaying information to a centralized coordinator is costly in short timeline scenarios and vulnerable to attacks on the single point of failure. Developing a system of decentralized coordination among agents is crucial to responding readily to incoming threats. This project aims to implement a solution to the problem of distributed two-dimensional multi-agent intercept.

The problem in question consists of n red team agents, each with its own trajectory toward a known point p, and n blue team agents whose roles are to intercept the red agents before they reach p. Furthermore, each blue agent only has a limited

radius r within which to communicate with and detect other agents. The goal is to develop a scheme by which blue agents decide where to move by detecting and communicating with neighboring agents in order to maximize the number of red agents intercepted.

2 Perception-Action-Cycle[1]

This algorithm is divided into several steps called the perception-action cycle (PAC) wherein agents move, sense, and update state estimates. In particular, at each time step, the following phases occur as in *Multi-Agent Surveillance and Tracking using Cyclic-Stochastic Gradient* by Botts et al:

1) TRUTH: Red agents move along their predetermined trajectory by updating their state where the state of red agent i is $[y_i^E(t), y_i^N(t), \dot{y}_i^E(t), \dot{y}_i^N(t)]$.

2) DECIDE: Each blue agent minimizes its own loss function and updates its velocity using a stochastic gradient. Blue agents move in the direction of the predicted next location of the nearest estimated red agent. Velocity updates affect heading by the following relation: $\theta_i(t) = \tan^{-1}(\dot{x}_i^N/\dot{x}_i^E)$

3) MOVE: Each blue agent updates its position by $x_i^E(\delta_t + t) := x_i^E(t) + \delta_t \dot{x}_i^E(t)$ and $x_i^N(\delta_t + t) := x_i^N(t) + \delta_t \dot{x}_i^N(t)$.

4) SENSE: Each blue agent detects all red agents within its detection radius with a

probability that drops off with distance from the blue agent.

5) COMMUNICATE: Each blue agent b broadcasts the states of the red agents it successfully detected, and any other blue agent within the detection radius of b receives the state.

6) INFER: Each blue agent updates its red agent state estimates using an extended Kalman filter.

3 Assumptions

Several simplifying assumptions were made in this formulation. Firstly, the red agents never deviate from their fixed trajectories even in the presence of a blue agent. Second, when agents of opposite teams collide, they immediately annihilate each other from the simulation. Furthermore, agents of the same team can pass through each other. Another assumption is that two blue agents can communicate with each other perfectly as long as they are in each other's radius of detection. Moreover, blue agents can differentiate between red and blue team agents without fault.

4 Algorithm

First we describe the physical layout of the scenario. The point p that the red agents are moving toward is set to be $(0, 0)$; all states and velocities are with respect to this origin point. The starting points of the n blue agents are placed at x-coordinate 0 and y-coordinate chosen uniformly randomly between $[-y_{max}, y_{max}]$ for some $y_{max} > 0$. The red agents are placed at x-coordinate x_{red_start} and y-coordinate randomly between $[-y_{max}, y_{max}]$.

Next we describe the movement of the red agents. Before the simulation begins, each red agent chooses a number d randomly from $\{2, 3, 4\}$. This number becomes the degree of a polynomial representing the agent's trajectory toward $(0, 0)$. Because $d+1$ points fully determine a polynomial of degree d and the start and end points of each red agent is already known, each agent chooses $d - 1$ more y-coordinate values randomly between $[-y_{max}, y_{max}]$ whose corresponding x-coordinate values partition $[0, x_{red\_start}]$ evenly. These $d + 1$ points create the polynomial trajectory of one red agent. During the TRUTH phase, each red agent updates its x-coordinate by moving west along a line whose slope is equal to the derivative of its trajectory at its current position. Its updated y-coordinate is chosen to be the value of the polynomial at its new x-coordinate. Note that the velocity of a red agent is never used in updating its state.

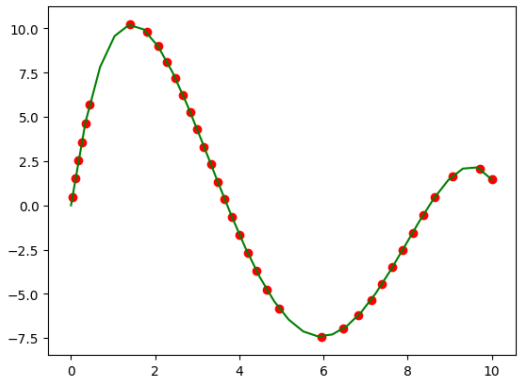

Figure 1: Example of a red agent's trajectory (right to left)

Subsequent phases of the algorithm deal solely with the actions of the blue agents. First, the DECIDE phase dictates that each blue agent b determine the red agent closest to it. If it is not aware of any red agents, then b does nothing and waits until the MOVE phase. Otherwise b attempts to minimize the following loss function $L_j(\dot{x}_j^E(t), \dot{x}_j^N(t)) = \frac{||[x_j^E(t+\delta t), x_j^N(t+\delta t)] - [y_i^E(t+\delta t), y_i^E(t+\delta t)]||^2}{2}$ where x_j is the state of b and y_i is the state of the estimated closest red agent to b. Note that to minimize this loss function requires knowledge of y_i at a future time step, $t + \delta t$. Therefore this loss function is estimated by $\hat{L}_j(\dot{x}_j^E(t), \dot{x}_j^N(t)) = \frac{||[x_j^E(t+\delta t), x_j^N(t+\delta t)] - [\hat{y}_i^E(t+\delta t|t), \hat{y}_i^E(t+\delta t|t)]||^2}{2}$, so the stochastic gradient is

$$\hat{g}_j(\dot{x}_j^E(t), \dot{x}_j^N(t)) = [x_j^E(t), x_j^N(t)] + \delta t[\dot{x}_j^E(t-\delta t), \dot{x}_j^N(t-\delta t)] - [\hat{y}_i^E(t+\delta t|t), \hat{y}_i^N(t+\delta t|t)]$$

and the update[2] is $[\dot{x}_j^E(t+\delta t), \dot{x}_j^N(t+\delta t)]^T := [\dot{x}_j^E(t), \dot{x}_j^N(t)]^T - a(\delta t)^{-2}\hat{g}_j(\dot{x}_j^E(t), \dot{x}_j^N(t))$

where $a > 0$ is a constant and therefore $a(\delta t)^{-2}$ is the gain sequence. Note that $a(\delta t)^{-2}$ can be held constant due to the fact that the agents are moving in time[2]. Because each blue agent attempts to minimize its own loss function individually, this approach is considered a cyclic stochastic optimization whose input vector is the concatenation of the state vectors of all blue agents and whose updates occur independently on the constituent subvectors.

The blue agents then progress to the MOVE phase where they update their positions using their new velocity vectors as described earlier. An intercept between blue and red agents b and r occurs if b moves to a position within its collision radius of r: $||x_j(t) - y_i(t)||_2 \leq c\_r_{x_j}$ where the collision radius $c\_r_{x_j}$ is taken to be much smaller than the detection radius $d\_r_{x_j}$. If an intercept occurs, then both b and r are immediately removed from the simulation. Therefore, a blue agent can only intercept one red agent regardless of the number of them within its collision radius.

After each blue agent has moved, the agents transition to the SENSE phase. At this step, each blue agent iterates through all red agents within its detection radius: $||x_j(t) - y_i(t)||_2 \leq d\_r_{x_j}$ and successfully detects with probability $1 - \dfrac{\rho(\mathbf{x}_j(t), \mathbf{y}_i(t))}{d\_r_{x_j}}$ where $\rho(\mathbf{x}_j(t), \mathbf{y}_i(t)) := |||[y_i^N(t) - x_j^N(t), y_i^E(t) - x_j^E(t)]||_2$. Note that the farther y_i is from x_j, the larger ρ is, so the smaller the probability of detection.

Next, the blue agents share detected red agents' states with each other in the

COMMUNICATE phase. Unlike red agents, blue agents are detected with probability 1 when within a radius of detection. Blue agent b passes all detected red agent states to all other blue agents within $d\_r_{x_j}$ and receives red agent states from other blue agents.

Lastly, during the INFER phase, each blue agent calculates the distance and azimuth angle to each red agent it is aware of from the previous step. These values are noisy measurements of $\phi(\mathbf{x}_j(t), \mathbf{y}_i(t)) := \tan^{-1}(\frac{y_i^N(t) - x_j^N(t)}{y_i^E(t) - x_j^E(t)})$ [1] and $\rho(\mathbf{x}_j(t), \mathbf{y}_i(t))$, namely $z_{j:i}(t) = [\phi(\mathbf{x}_j(t), \mathbf{y}_i(t)), \rho(\mathbf{x}_j(t), \mathbf{y}_i(t))] + v$ where $v \sim \mathcal{N}(0, R)$. The noisy measurements for the red agent that is closest to b are then passed to an extended Kalman filter, which updates the red agent state estimate under the assumption that the true state of red agent i follows $y_i(t + \delta t) = \Phi y_i(t) + w(t)$ where $w(t) \sim \mathcal{N}(0, Q)$. The following extended Kalman filter algorithm[2] is used to update the red agent state estimates as in *Cyclic Stochastic Optimization: Generalizations, Convergence, and Applications in Multi-Agent Systems* by Hernandez.

Predict: (execute at time $t - \delta t$)

1. $\hat{y}_i(t|t-\delta t) = \Phi \hat{y}_i(t-\delta t|t-\delta t)$ where $\Phi == \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

2. $P_{t|t-\delta t} = \Phi P_{t-\delta t|t-\delta t} \Phi^T + Q$ where Q is the covariance matrix of w taken as $\text{diag}(0.3, 0.3, 0.05, 0.05)$[1].

Correct: (execute at time t)

3. $\eta = z_{j:i}(t) - [\phi(x_j(t), \hat{y}_i(t|t-\delta t)), \rho(x_j(t), \hat{y}_i(t|t-\delta t))]^T$

4. $S = H P_{t|t-\delta t} H^T + R$ where R is the covariance of $z_{j:i}(t)$ and $H = \begin{bmatrix} \frac{x^N - y^N}{\rho^2(x,y)} & \frac{-(x^E - y^E)}{\rho^2(x,y)} & 0 & 0 \\ \frac{-(x^E - y^E)}{\rho(x,y)} & \frac{-(x^N - y^N)}{\rho(x,y)} & 0 & 0 \end{bmatrix}$

5. $K = P_{t|t-\delta t} H^T S^{-1}$

6. $\hat{y}_i(t|t) = \hat{y}_i(t|t-\delta t) + K\eta$

7. $P_{t|t} = (I - KH) P_{t|t-\delta t}$

It is important to note that H in the algorithm is defined to be the Jacobian of $\begin{bmatrix} \phi(x_j(t), \hat{y}_i(t|t-\delta t)) \\ \rho(x_j(t), \hat{y}_i(t|t-\delta t)) \end{bmatrix}$; however, this vector is not differentiable if and only if either of the following holds[2]:

1. $x_j^E(t) = \hat{y}_i^E(t|t-\delta t)$ and $x_j^N(t) > \hat{y}_i^N(t|t-\delta t)$

2. $\rho(x_j(t), \hat{y}_i(t|t-\delta t))$.

Furthermore, this algorithm relies on $\hat{y}_i(t-\delta t|t-\delta t)$ and $P_{t-\delta t|t-\delta t}$ from the previous time step; however it is difficult for a blue agent to associate measurements of states at different time steps to a particular red agent because multiple red agents could be in close proximity to each other. Association of states to red agents is done by projecting $P_{t-\delta t|t-\delta t}$ forward to $P_{t|t-\delta t}$ and then calculating $\eta = z_{j:i}(t) - [\phi(x_j(t), \hat{y}_i(t|t-\delta t)), \rho(x_j(t), \hat{y}_i(t|t-\delta t))]^T$ as is done in the algorithm. If $\eta < \eta_T$, for some threshold η_T, then the red agent corresponding to the current relative state measurement is assumed to be the same as the red agent whose state is estimated as $\hat{y}_i(t-\delta t|t-\delta t)$ at time $t-\delta$. Alternatively, if $\eta \geq \eta_T$, then the red agents are assumed to be different, so $\hat{y}_i(t-\delta t|t-\delta t)$ in the extended Kalman filter algorithm is taken to be the blue agent's state plus the relative state offset plus the product of the blue agent's current velocity and δt. This correction is done because the blue agent does not have an estimate of the previous state position of the red agent it is now tracking, so it assumes the red agent moved to its current position at the blue agent's speed but in the opposite direction.

5 Results

A baseline must be established in order to analyze this algorithm. Without the extended Kalman filtering and gradient updates and over 2000 runs of a five blue agent versus five red agent simulation, on average 4.0111 red agents were intercepted. The constraints imposed in this baseline imply that the blue agents are on static east-oriented trajectories towards the blue agents. In comparison, when utilizing the stochastic optimizations, experiments over four tuning parameters were done. The parameters of interest were the gain sequence, detection radius, relative distance threshold (η_T threshold), and the covariance of the relative distance measurements. The use of stochastic optimization did appear to moderately improve the number of intercepts that occurred as seen in the tables below. Table 1 shows that the average number of red agents intercepted is higher for an η threshold value closer to 0.6 than 0.8, but that the effect of the detection radius is more ambiguous.

Setting $\eta_T = 0.6$ and $d_r = 2.0$, simulations were run to compare gain sequence values and relative distance covariances. Table 2 shows that for larger gain sequences, the blue agents were able to intercept the red agents slightly more easily. The table also indicates that the algorithm appears to be fairly robust against increasing noise levels in the relative distance measurements.

There are many other tunable parameters in the algorithm that could be adjusted

	$d_r = 2.0$	$d_r = 5.0$	$d_r = 8.0$
$\eta_T = 0.6$	4.0452	4.0427	4.0419
$\eta_T = 1.0$	4.0279	4.0325	4.0295
$\eta_T = 1.4$	4.0208	4.0252	4.023

Table 1: η_T versus Detection Radius

	$\rho_\sigma = 0.1$	$\rho_\sigma = 0.2$	$\rho_\sigma = 0.3$
gain = 0.6	4.0334	4.0312	4.0347
gain = 0.7	4.0371	4.035	4.0387
gain = 0.8	4.0464	4.0475	4.0446

Table 2: σ_ρ versus Gain

for experimentation such as the noise level of the relative azimuth measurement, the time step size, the Kalman filter process noise, and the number of agents in the simulation. Future work could analyze how these parameters affect the number of intercepts. More involved future work would entail removing many of the assumptions used in this algorithm. For example, modeling fratricide of blue agents, reactive red agents, and a more elaborate coordination scheme between agents would provide a more realistic approximation of the problem.

6 Conclusion

This project describes and analyzes a low-fidelity model for the problem of distributed multi-agent intercept. All red agents travel on a static trajectory toward a fixed point p while blue team agents attempt to intercept them by cyclically updating their states and communicating the states of detected red agents with each other. The goal is for the blue agents to intercept as many red agents as possible before the red agents reach point p. This is achieved by each blue agent estimating the state of its nearest red agent with an extended Kalman filter and minimizing its own loss function accordingly.

Experimentation shows that using these stochastic optimization techniques improves the number of red agents intercepted compared to the baseline where blue

agents are on static east-oriented trajectories. Although simulations do show some improvement, a better understanding of the problem would be attained from widening the scenario space from a 10x10 plane into a larger, three-dimensional space and modeling more complex, reactive behavior.

References

[1] C. Botts, J. Spall, and A. Newman, "Multi-agent surveillance and tracking using cyclic-stochastic gradient," in *Proc. of the American Control Conference*, Boston, MA, June 2016, pp. 270-275.

[2] K. Hernandez, "Cyclic stochastic optimization: generalizations, convergence, and applications in multi-agent systems," Johns Hopkins University, 2017, ch. 8.