Multi-robot Target Tracking

Arpit Aggarwal
University of Maryland, College Park
Maryland, USA
arpitagg@umd.edu

I. Introduction

The project, Multi-robot Target Tracking, focuses on the task of localizing the target(s) positions by use of mobile robots. The tasks in this project were two-fold, first determine an efficient way of localizing the target(s) position and secondly, determine an optimal robot trajectory for minimizing the uncertainty in the target(s) position. The initial information given to us was the initial robot(s) position, initial target(s) mean and initial target(s) variance. After creating the heatmap which modeled the uncertainty in the target(s) positions at a particular time t, the next step was to give this heatmap as an input to the RL/greedy algorithm that gave an optimal next step that can be taken by the robot(s) at time step t+1.

II. Related Work

The work followed for this project was the paper, Active Localization of Multiple Targets from noisy relative measurements[1]. In this paper, the uncertainty maps of the target(s) positions were modeled using Bayesian Histograms as the target(s) were assumed to be stationary. These heatmaps were given as an input to the Reinforcement Learning(RL) algorithm, that found an optimal trajectory the robot needed to follow. The purpose of this project was to extend the idea of this paper for the case where target(s) could be moving.

III. Method for creating heatmap

The first task was creating a heatmap that models the uncertainty in the target(s) position. Two approaches, namely, Extended Kalman Filters(EKF) and Bayesian Histograms were used and the most optimal one was used ahead as an input for the second task. The two approaches are briefly described below.

A. Bayesian Histograms

In this approach, we divide the environment into grid cells of fixed width(w) and height(h). All grid cells are initialized with the probability of 1, meaning that each grid cell has equal probability of containing the target. Since the measurement model assumes zero-mean normal distribution for the sensor noise, the probability of a grid cell v being the true target position q^* given a relative measurement $\widehat{z}$ as:

$$P(v = q^*|p, \widehat{z}) = f(z_v|\widehat{z}, \sigma_s^2) = \frac{1}{\sqrt{2\pi\sigma_s^2}} exp(\frac{-1}{2\sigma_s^2}(z_v - \widehat{z})^2$$

where, z_v is the distance between the point v and the robot position p. As the robot plans its trajectory, the probability of the grid cell belonging to the target is updated as:

$$P(v = q^*|P_{1:T}, \widehat{z}_{1:T}) = \prod f(z(p_k, v)|\widehat{z_k}, \sigma_s^2)$$

B. Extended Kalman Filter

As the target(s) motion in our case was non-linear, Extended Kalman Filters(EKF) were used for generating heatmaps that modeled the uncertainty in target(s) position at a particular time t. The equations for the prediction and update steps in the Extended Kalman Filter(EKF) are given in Figures 1 and 2.

Prediction:

$$\hat{o}^-(k) = \hat{o}(k-1),$$

$$\hat{\Sigma}^-(k) = \hat{\Sigma}(k-1) + R(k).$$

Fig. 1.

Update:

$$K(k) = \hat{\Sigma}^-(k)H^T(k)(H(k)\hat{\Sigma}^-(k)H^T(k) + Q(k))^{-1},$$

$$\hat{o}(k) = \hat{o}^-(k) + K(k)(z(k) - h(\hat{o}^-(k)))$$

$$\hat{\Sigma}(k) = (I - K(k)H(k))\hat{\Sigma}(k)^-$$

where $R(k)$ and $Q(k)$ are the covariance matrices of the noise from target's motion model and robot's measurement, respectively. $h(\hat{o}^-(k)) := \|p(k) - \hat{o}^-(k)\|_2$. $z(k)$ denotes the noisy distance measurement from the robot. $H(k)$ is the Jacobean of $h(\hat{o}^-(k))$.

Fig. 2.

C. Comparison of EKF and Bayesian Histogram approaches

For determining the most optimal approach for modeling the uncertainty in the target(s) position, the analysis of both approaches was done in an environment of size 20x20 unit. The range sensor was used for tracking the target and it

was assumed that target was always within the range of the sensor. The two approaches were compared for target(s) moving slow case(omega=100 rad/s) and target(s) moving fast case(omega=33 rad/s). For first case, the robot was kept stationary, while for the other case, the robot followed the greedy algorithm. The greedy algorithm took into account the radius around the robot of 2 units and discretized the circular region into 12 actions. The optimal action was decided using the determinant of the Fisher Information Matrix(FIM).

$$L = \frac{1}{2\sigma^2} * v_i^2 * v_j^2 * sin^2\alpha$$

where,

v_i = the distance between robot position at time step t and estimated target mean
v_j = the distance between the future robot position and estimated target mean
alpha = angle between v_i and v_j

Case 1(Stationary robot)

Bayesian Histograms for Target moving slow
Bayesian Histograms for Target moving fast
EKF for Target moving slow
EKF for Target moving fast

Case 2(Robot following greedy algorithm)

Bayesian Histograms for Target moving slow
Bayesian Histograms for Target moving fast
EKF for Target moving slow
EKF for Target moving fast

IV. Method for Generating Optimal Action

Heatmaps generated in the first task were used as an input to the Actor-Critic model for generating the action that the robot(s) need to take at next time step. The different Actor-Critic models are briefly described below.

A. Description of the Actor-Critic Model

The actor-critic model used for this project was Twin Delayed Deep Deterministic Policy Gradients(TD3)[2]. TD3 is the successor of Deep Deterministic Policy Gradient(DDPG). Like many RL algorithms, training DDPG can be unstable and heavily reliant on finding the correct hyperparameters. This is caused by the algorithm continuously over estimating the Q values of the critic network. TD3 addresses this issue by focusing on reducing the overestimation bias seen in previous algorithms[3].

B. Actor-Critic Model 1

In this model, the agent state that was used as an input to the Actor-Critic model was the flattened heatmap along with the robot(s) position. The block diagram is shown in Figure 3. The reward training curve is shown in Figure 4. The action space was, angle from -pi to pi and step size from 0 to 1.

Case 1(Robots=1 and targets=1)

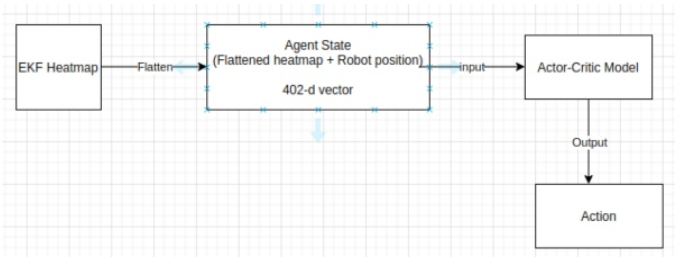

Fig. 3. Actor-Critic Model 1

The reward training curve for robots equal to 1 and targets equal to 1 is shown in Figure 4. The video of the agent performance during the evaluation stage can be seen here.

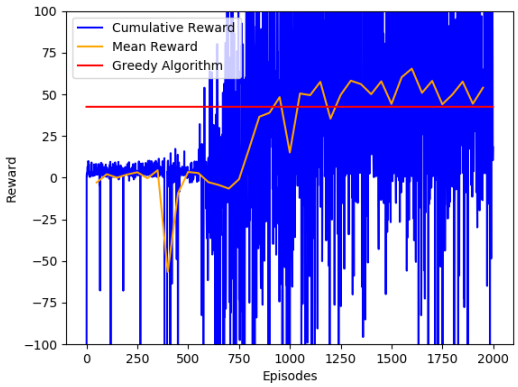

Fig. 4. Reward Training Curve for robots=1 and targets=1 case

Case 2(Robots=1 and targets=2)
The reward training curve for robots equal to 1 and targets equal to 2 is shown in Figure 5. The video of the agent performance during the evaluation stage can be seen here.

Case 3(Robots=1 and targets=4)
The reward training curve for robots equal to 1 and targets equal to 4 is shown in Figure 6. We observe two possible scenarios in this case. The best case occurs when the target(s) are near to each other and in that case the policy is being learnt by the agent. However, when the target(s) are a distance apart from each other, in that case, the robot simply follows a straight line motion as its policy. The video of the agent performance during the evaluation stage(best case) can be seen here. The video of the agent performance during the

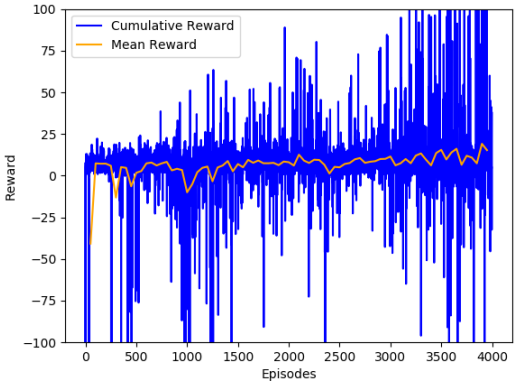

Fig. 5. Reward Training Curve for robots=1 and targets=2 case

evaluation stage(worst case) can be seen here.

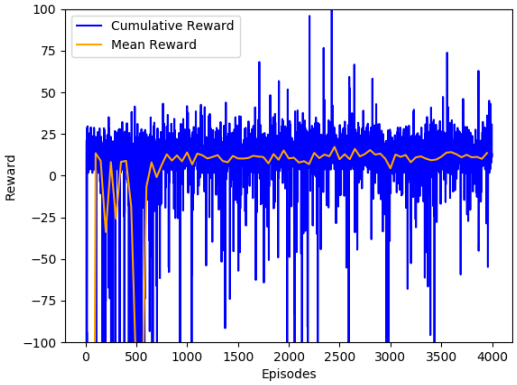

Fig. 6. Reward Training Curve for robots=1 and targets=4 case

Case 4(Robots=2 and targets=2)
The reward training curve for robots equal to 2 and targets equal to 2 is shown in Figure 7. The video of the agent performance during the evaluation stage can be seen here. We observe that each robot learns to move towards its respective target and follows that target for the rest of the time steps.

Case 5(Robots=2 and targets=4)
The reward training curve for robots equal to 2 and targets equal to 4 is shown in Figure 8. The video of the agent performance during the evaluation stage can be seen here. We observe that in this case the robots hardly move. This is due to the fact that the policy being learnt is not optimal and this makes us experiment with other actor-critic configurations as illustrated below.

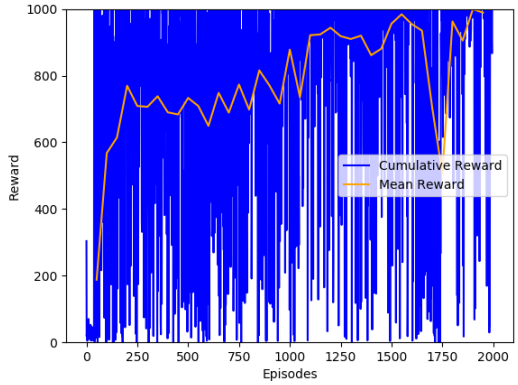

Fig. 7. Reward Training Curve for robots=2 and targets=2 case

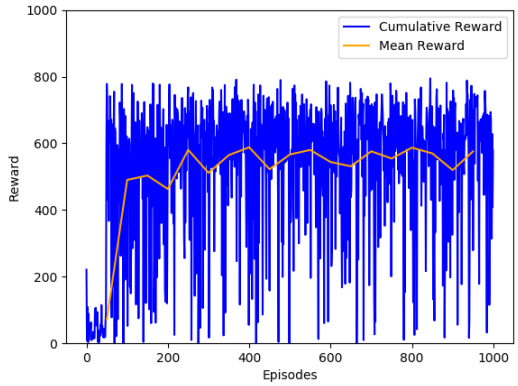

Fig. 8. Reward Training Curve for robots=2 and targets=4 case

C. Actor-Critic Model 2

In this model, the agent state that was used as an input to the Actor-Critic model was the feature vector after passing the heatmap through ResNet-34 with fixed weights and the robot(s) position were concatenated to the feature vector obtained from the CNN. The block diagram is shown in Figure 9. The weights of the CNN were kept fixed. The reward is the same used as before. The action space was, angle from -pi to pi and step size from 0 to 1.

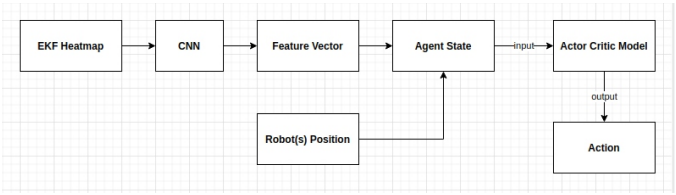

Fig. 9. Actor-Critic Model 2

Case 1(Robots=2 and targets=4)
The video of the agent performance during the evaluation stage can be seen here. We saw improvement in the agent

performance for robots=2 and targets=4 case in this scenario as compared to the previous version of the Actor-Critic model which can be seen in Figure 10. However, there is no clear distinction on whether this version of Actor-Critic model performed better than the previous version and thus we needed to explore more configurations of the Actor-Critic models.

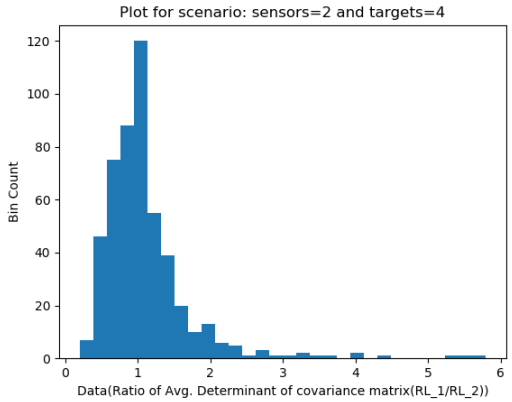

Fig. 10. Comparison of RL algorithms for robots=2 and targets=4 case

D. Actor-Critic Model 3

In this model, the agent state that was used as an input to the Actor-Critic model was the feature vector after passing the heatmap through ResNet-18 with fixed weights and the robot(s) position were concatenated to the feature vector obtained from the CNN. The block diagram is shown in Figure 11. The weights of the CNN were kept fixed. The reward is the same used as before. The action space was, angle from -pi to pi and step size from 0 to 1.

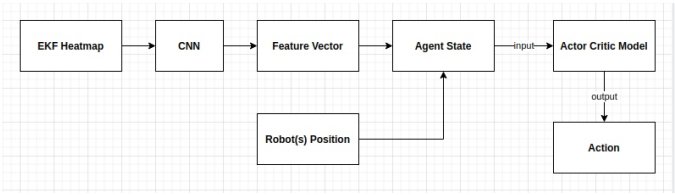

Fig. 11. Actor-Critic Model 3

Case 1(Robots=2 and targets=4)
The video of the agent performance during the evaluation stage can be seen here. We saw improvement in the agent performance for robots=2 and targets=4 case in this scenario as compared to the previous version of the Actor-Critic model where no CNN was used. However, there is no clear distinction on whether this version of Actor-Critic model performed better than the previous version and thus we needed to explore more versions of the Actor-Critic models.

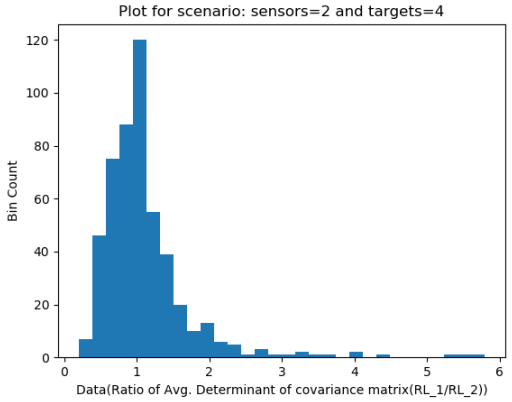

Fig. 12. Comparison of RL algorithms for robots=2 and targets=4 case

Case 2(Robots=2 and targets=2)
The video of the agent performance during the evaluation stage can be seen here. We saw the same performance for robots=2 and targets=2 case in this scenario as compared to the previous version of the Actor-Critic model where no CNN was used, as can be seen in Figure 13.

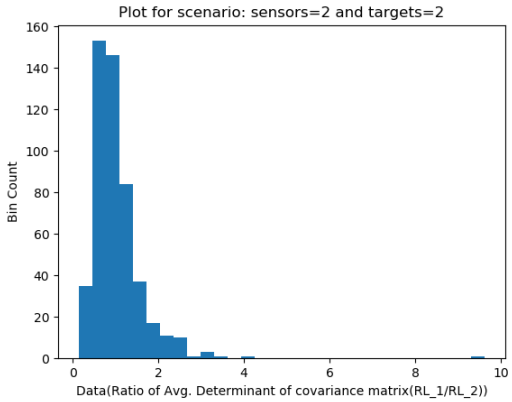

Fig. 13. Comparison of RL algorithms for robots=2 and targets=2 case

E. Actor-Critic Model 4

In this model, the agent state that was used as an input to the Actor-Critic model was the feature vector after passing the heatmap through a CNN and the robot(s) position were concatenated to the feature vector obtained from the CNN. The block configuration is shown in Figure 11. The weights of the CNN were not fixed and were updated during each training iteration. The reward is the same used as before. The action space was, angle from -pi to pi and step size from 0 to 1.

Case 1(Robots=2 and targets=4)
The video of the agent performance during the evaluation stage

can be seen here. For the case where the CNN parameters were learnable, the agent wasnt able to learn an optimal policy and the robots remained at the same location as can be seen in the video. The reward training curve indicates that no learning was taking place. Different hyperparameters setting were tried, however, similar performance was observed.

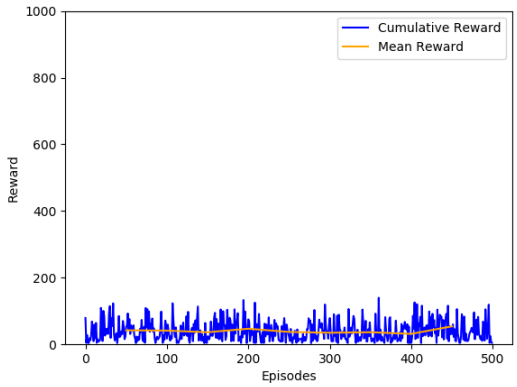

Fig. 14. Reward Training Curve for robots=2 and targets=4 case

F. Actor-Critic Model 5

Current work involves the creation of heatmap for each sensor in the environment. That is, previously a single heatmap was generated for the entire environment and all the sensors, however, now by creating a heatmap for each sensor separately, the effect of a wrong sensor reading was eliminated on the heatmaps generated for the other sensors. The types of models that are being used are shown in Figures 15 and 16. The action space was, angle from -pi to pi and step size from 0 to 1.

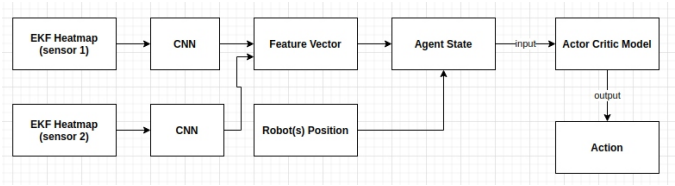

Fig. 15. Actor-Critic Model 5

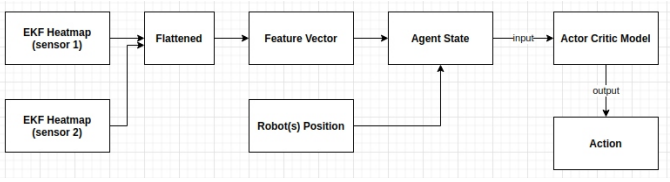

Fig. 16. Actor-Critic Model 5

Case 1(Robots=2 and targets=4)

The reward training curve shown in Figure 17 indicates that no learning was taking place. Different hyperparameter setting were tried, however, similar performance was observed.

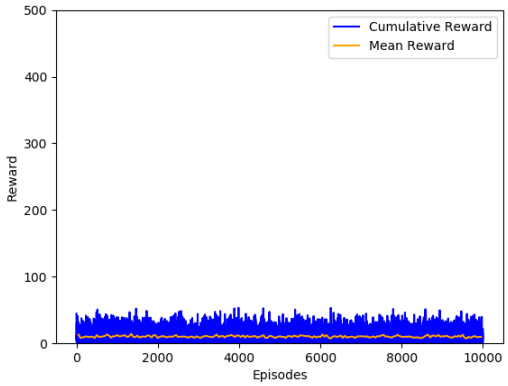

Fig. 17. Reward Training Curve for robots=2 and targets=4 case

Case 2(Robots=2 and targets=2)

The reward training curve shown in Figure 18 indicates that no learning was taking place. Different hyperparameter setting were tried, however, similar performance was observed.

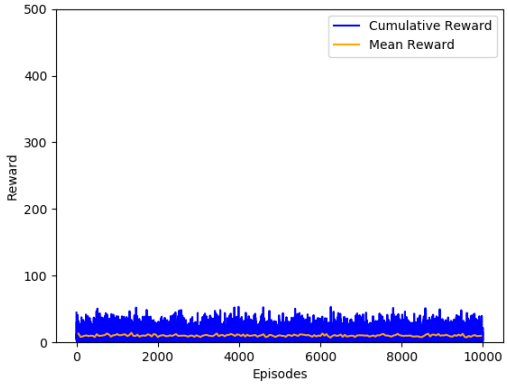

Fig. 18. Reward Training Curve for robots=2 and targets=2 case

V. Future Work

1. Extend work on heatmap generation for each sensor and compare the results where only one heatmap is generated for the multi-robot case.

2. Instead of concatenating the robot position with the final feature vector obtained from CNN, try using the image with 2 channels where the first channel encodes the heatmap information and second channel encodes the robot position given as the input to the CNN

REFERENCES

[1] https://github.com/ksengin/active-target-localization
[2] https://arxiv.org/pdf/1802.09477.pdf
[3] https://towardsdatascience.com/td3-learning-to-run-with-ai-40dfc512f93
[4] https://github.com/arp95/robot_target_tracking