Preemptive Multi-Robot Motion Planning for Target Tracking

PRAMOD KULKARNI, Carnegie Mellon University

ABSTRACT – This report presents the problem statement, approach, results of my 15-780 Graduate Artificial Intelligence (Spring 2013) course project. This project explores an approach to preemptive multi-robot motion planning for active target tracking. Contrary to previous approaches, the technique presented here is to track the reachability set for a target, for a given look-ahead period, instead of tracking a target's state. Preemptive decisions are made using the following procedure: (1) apply a motion model to the target, (2) find the target's reachability set for a set look-ahead period, and (3) plan robot motions to maintain sensor coverage of the target's reachability set. Coverage gained by performing an action is proposed as a heuristic for robot motion planning. Anticipated results are shown based on preliminary findings, and future work is outlined.

1. INTRODUCTION

Motion planning for autonomous active target tracking is a well established and studied problem in the Artificial Intelligence and Robotics communities. Aspects of the problem span the full spectrum of topics from perception and state estimation to planning/decision systems and control. Tracking using a single sensor [1] and in more recent years, multiple sensors [2], [3] have been explored. Also, a variety of work has been done on using different observations for target state localization [4], [5], [6]. Although a wide range of techniques have been tried and tested over the years, many of the approaches we find in the literature are reactionary. Reactionary approaches have the drawback of potentially falling into a game of "catch up". For an algorithm that tracks the state of a given target, this issue stems from the delays accrued in state estimation.

This project explores a simple approach to avoiding this game of "catch up". The goal is to generate preemptive motion plans for a team of robots that cooperatively track a target in an environment with obstacles. The concept at the heart of the approach is to plan for tracking the reachability set of a target (for a defined $\boldsymbol{\Delta t}$ look-ahead) instead of planning to track the state of the target. Of course, the first question that immediately arises: How do we calculate this reachability set? For this we will need a model of the target's behavior. Making an assumption about the model and applying it to predict the behavior of a target is not all that unrealistic. For example, many ground vehicles can be modeled as differential drive vehicles.

The work presented in this report simplifies the real world scenario for the purposes of a 15-780 Graduate Artificial Intelligence (Spring 2013) course project.

Section 2 further details the problem statement. Section 3 organizes the overall approach being used. Sections 4 and 5 define state and motion models for the target and robots. Section 6 describes the sensor model being used. Section 7 describes the simulation setup being used. Section 8 and 9 show anticipated results. And finally, Section 10 and 11 discuss some directions in which future work can be taken and conclusions.

2. PROBLEM STATEMENT

2.1 Time and Space

For the purposes of the simplified problem at hand, time is discretized into time steps (t0, t1, t2, etc …). The environment is also discretized, into an X-by-Y grid world with obstacles. Grid cells can be occupied by a maximum of one entity (robot, target, or obstacle) at any given time. An example environment is shown in Figure 1.

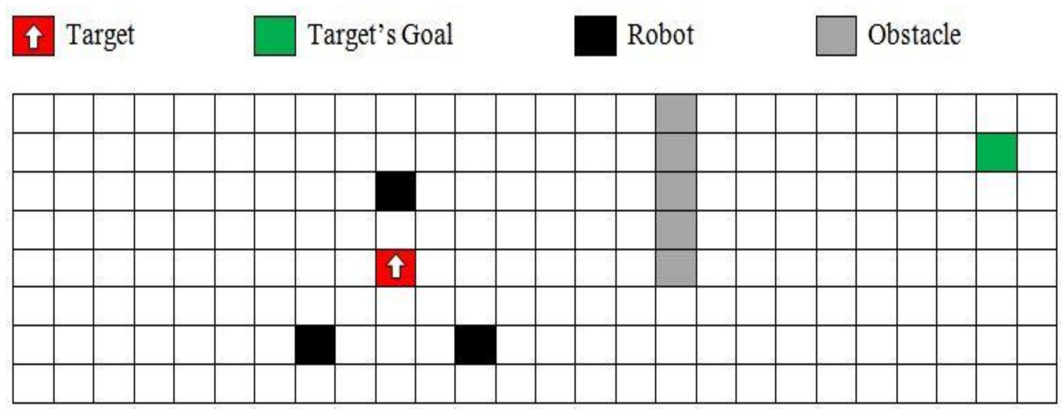

Fig. 1. Example Environment

2.2 Target

There is one target in the environment which occupies one grid cell of the environment. This target traverses a path through the environment trying to reach a goal. The target's goal is given by an (x, y) position. In order to achieve the objective, described in Section 2.4, we will need to have a model (Section 4) describing the motion of the target. The purpose of having a model is to enable the computation of the reachability set of the target for a given $\boldsymbol{\Delta t}$ look-ahead.

2.3 Robots

There are $\boldsymbol{M}$ robots in the environment which need to cooperatively track the target's movement in the environment. Each robot occupies one grid cell of the environment. Every robot has a sensor that can detect the target's state. However, these sensors have a limited range of operation; hence, each robot has a limited field of view. The limited field of view will motivate the motion of robots. Robots themselves will have a simple motion model (Section 5).

2.4 Objective

The goal of this project is as follows: Given the state of the target and a model of the target's behavior, generate feasible and safe motion plans, for the $\boldsymbol{M}$ robots, which guarantee that at least $\boldsymbol{n}$ robots maintain view of the target's reachability set (for a given $\boldsymbol{\Delta t}$ look-ahead). A feasible motion plan is one that can be executed by a robot, given the constraints of its motion model. Safe motion plans keep the robots inside the grid world and do avoid collisions among robots, collisions with the target, and collisions with obstacles.

3. APPROACH

For this project, a simple centralized approach based on search will be used to generate plans. First we define state and motion models for the target and the robots. Using the target's model, a planner is employed to generate the target's motion.

Next, as a prerequisite to robot motion planning, the target's motion model is used to compute the target's reachability set at a given time step for a given Δt look-ahead. Finally, a centralized search based planner is used to generate robot motion plans. The goal of the planner is to ensure that the fields of view of at least n robots cover the reachability set of the target (for the given Δt look-ahead). The coverage gained by a robot performing an action is used as a heuristic.

4. TARGET STATE AND MOTION MODEL

4.1 Target State

The state of the target is given by: (1) the grid cell that it occupies, and (2) its orientation. A grid cell is specified by its (x, y) location, and the orientation can be: 0 (zero), N (north), S (south), E (east), or W (west). The zero orientation is used to represent the target's state when it is not moving (i.e. when it has zero "velocity").

4.2 Target Action Schema

In one time step the target can move from its current grid cell to a neighbouring grid cell. The action schema for the target is shown in Figure 2 using a STRIPS [7], [8] like formulation. In the problem formulation presented here, there is no uncertainty associated with the effect of actions performed by the target.

Action: TargetNorth Pre: targetAt(x, y) ^ (targetFacing(N) v targetFacing(0)) ^ unoccupied(x, y+1) Effect: targetAt(x, y+1) ^ ~targetAt(x, y) If targetFacing(0) targetFacing(N) ^ ~targetFacing(0)	Action: TargetSouth Pre: targetAt(x, y) ^ (targetFacing(S) v targetFacing(0)) ^ unoccupied(x, y-1) Effect: targetAt(x, y-1) ^ ~targetAt(x, y) If targetFacing(0) targetFacing(S) ^ ~targetFacing(0)
Action: TargetEast Pre: targetAt(x, y) ^ (targetFacing(E) v targetFacing(0)) ^ unoccupied(x+1, y) Effect: targetAt(x+1, y) ^ ~targetAt(x, y) If targetFacing(0) targetFacing(E) ^ ~targetFacing(0)	Action: TargetWest Pre: targetAt(x, y) ^ (targetFacing(W) v targetFacing(0)) ^ unoccupied(x-1, y) Effect: targetAt(x-1, y) ^ ~targetAt(x, y) If targetFacing(0) targetFacing(W) ^ ~targetFacing(0)
Action: TargetNorth-East Pre: targetAt(x, y) ^ (targetFacing(N) v targetFacing(E)) ^ unoccupied(x+1, y+1) Effect: targetAt(x+1, y+1) ^ ~targetAt(x, y) If targetFacing(N) targetFacing(E) ^ ~targetFacing(N) Else targetFacing(N) ^ ~targetFacing(E)	Action: TargetNorth-West Pre: targetAt(x, y) ^ (targetFacing(N) v targetFacing(W)) ^ unoccupied(x-1, y+1) Effect: targetAt(x-1, y+1) ^ ~targetAt(x, y) If targetFacing(N) targetFacing(W) ^ ~targetFacing(N) Else targetFacing(N) ^ ~targetFacing(W)
Action: TargetSouth-East Pre: targetAt(x, y) ^ (targetFacing(S) v targetFacing(E)) ^ unoccupied(x+1, y-1) Effect: targetAt(x+1, y-1) ^ ~targetAt(x, y) If targetFacing(S) targetFacing(E) ^ ~targetFacing(S) Else targetFacing(S) ^ ~targetFacing(E)	Action: TargetSouth-West Pre: targetAt(x, y) ^ (targetFacing(S) v targetFacing(W)) ^ unoccupied(x-1, y-1) Effect: targetAt(x-1, y-1) ^ ~targetAt(x, y) If targetFacing(S) targetFacing(W) ^ ~targetFacing(S) Else targetFacing(S) ^ ~targetFacing(W)
Action: TargetStop	

Pre: targetFacing(dir) Effect: targetFacing(0) ^ ~targetFacing(dir)	

Fig. 2. Target Action Schema

4.3 Target Reachability Set

The target's reachability set for a given $\boldsymbol{\Delta t}$ look-ahead time is found by applying the definitions of the target's state the target's action schema. Figure 3 illustrates some reachability sets. If the target is standing still, in one time step it can move North, South, East, West, or stay where it is. However, if the target has a "velocity", as indicated by the orientation of the target, it can move "forward", "forward-left", or "forward-right" in one time step. This behaviour is enforced by action preconditions in the action schema.

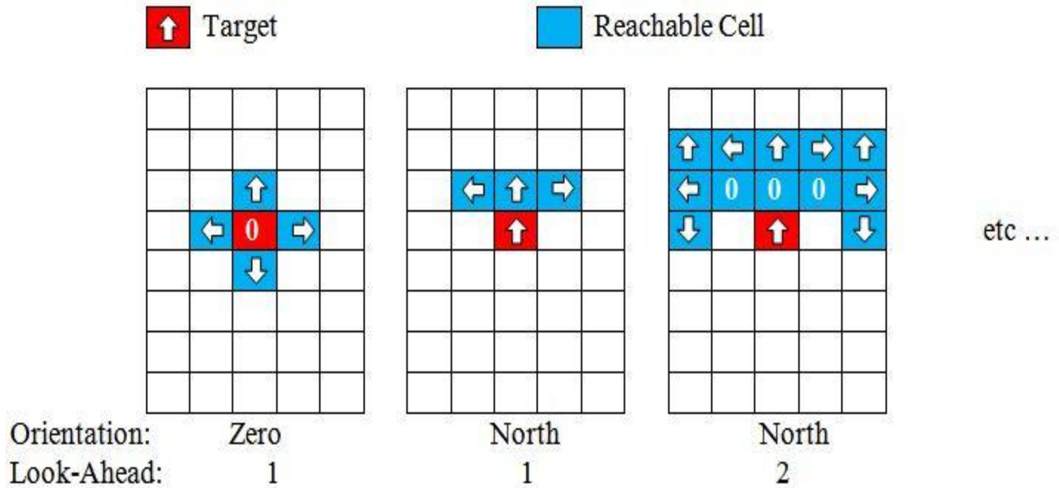

Fig. 3. Target Reachability Sets

5. ROBOT STATE AND MOTION MODEL

5.1 Robot State

The state of an individual robot is given by the grid cell that it occupies. A grid cell is specified by its (x, y) location. For simplicity, orientation is not used to characterize the state of a robot. However, including orientation in the future can help one draw more realistic conclusions from the results of this project.

5.2 Robot Action Schema

In one time step an individual robot can move from its current grid cell to any of its eight neighbouring grid cells. The action schema for the robot is shown in Figure 4 using a STRIPS like formulation. In the problem formulation presented here, there is no uncertainty associated with the effect of actions performed by the robot.

Action: RobotNorth Pre: robotAt(x, y) ^ unoccupied(x, y+1) Effect: robotAt(x, y+1) ^ ~robotAt(x, y)	Action: RobotSouth Pre: robotAt(x, y) ^ unoccupied(x, y-1) Effect: robotAt(x, y-1) ^ ~robotAt(x, y)
Action: RobotEast Pre: robotAt(x, y) ^ unoccupied(x+1, y) Effect: robotAt(x+1, y) ^ ~robotAt(x, y)	Action: RobotWest Pre: robotAt(x, y) ^ unoccupied(x-1, y) Effect: robotAt(x-1, y) ^ ~robotAt(x, y)
Action: RobotNorth-East	Action: RobotNorth-West

Pre: robotAt(x, y) ^ unoccupied(x+1, y+1) Effect: robotAt(x+1, y+1) ^ ~robotAt(x, y)	Pre: robotAt(x, y) ^ unoccupied(x-1, y+1) Effect: robotAt(x-1, y+1) ^ ~robotAt(x, y)
Action: RobotSouth-East Pre: robotAt(x, y) ^ unoccupied(x+1, y-1) Effect: robotAt(x+1, y-1) ^ ~robotAt(x, y)	Action: RobotSouth-West Pre: robotAt(x, y) ^ unoccupied(x-1, y-1) Effect: robotAt(x-1, y-1) ^ ~robotAt(x, y)
Action: RobotStop Pre: robotAt(x, y) Effect: robotAt(x, y)	

Fig. 4. Robot Action Schema

6. SENSOR MODEL

Each robot has a sensor that can detect the target's state (position and orientation). Further, each sensor has a maximum range (***r***) defined by the number of rings of grid cells around the robot that it can "see". This is illustrated in Figure 5. In the problem formulation presented here, there is no uncertainty associated with measurements.

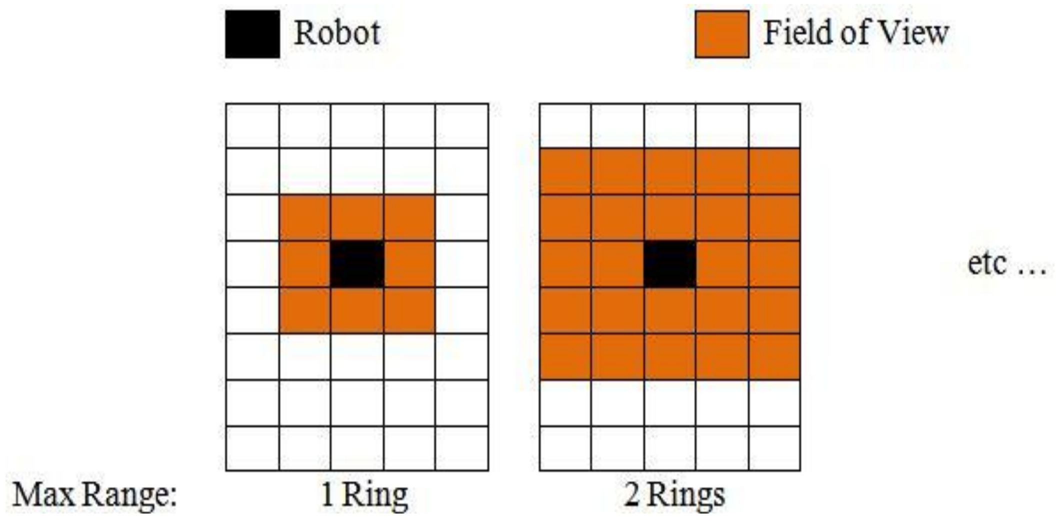

Fig. 5. Sensor Model

7. SIMULATION

A simulation platform, with visualization, was created for this project using Java. For simplicity, the target and robots' states are treated as properties of the environment. This allows for a straight forward implementation. Also, information about whether a grid cell of the environment is occupied or unoccupied is maintained by the environment itself. The target and robot action schema make extensive use of this information; however, none of the actions effect the occupied/unoccupied information. Instead, the environment updates the occupied/unoccupied information by keeping track of all actions. The purpose of this arrangement is that we do not want the planners to move the target or the robots just to make space for each other.

The environment needs to be static in order to generate a plan, so the simulation is executed in time steps. After each time step, the environment is static. At that point if the preconditions of the next planned action are met, that next action is performed. However, if the preconditions of the plans next action are not met, we replan based on the current state of the environment.

8. TARGET MOTION PLANNING

Figure 6 shows an example target path motivated by preliminary results of the target planning. The planner is given the goal of reaching the (x, y) position of the goal regardless of the targets orientation (Figure 6). Hence, there are four goal states (each with the same grid cell position). Distance to the target can be used as a heuristic for optimal path planning.

```
Goal:     targetAt(goalX, goalY)
```

Fig. 6. Target Planner Goal

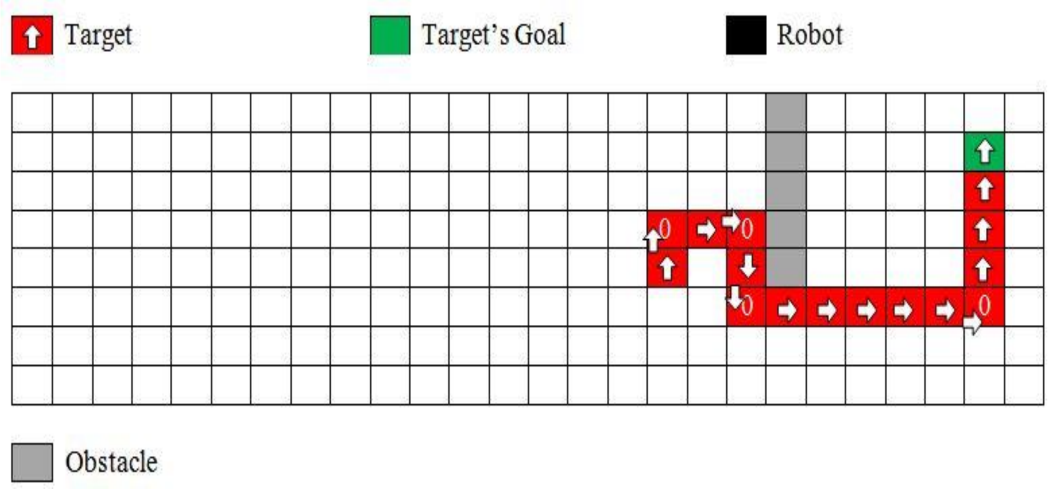

Fig. 7. Target Motion Plan

9. ROBOT MOTION PLANNING

Let us consider an example instantiation of the problem at hand. Let $\boldsymbol{M}$ = 3 robots, $\boldsymbol{\Delta t}$ = 1 time step, $\boldsymbol{r}$ = 2 rings, $\boldsymbol{n}$ = 2 robots. In this example, there are three robots tracking the target, each with a sensor with maximum range of two rings around the robot. A $\boldsymbol{\Delta t}$ look-ahead of 1 time step is used for calculating reachability sets. And, the objective is to have the sensor ranges of at least two robots cover the reachability set of the target as the target moves in the environment. The coverage gained by performing a particular action in a particular state, given the state of the target, can be used as a heuristic. In order to obtain results in the short period of time available, a simple brute force search of the 8 grid cells around each robot, at each time step, was used to find motion plans step-by-step. Using this, the anticipated behavior of the robots, generated by the motion planning described above, is shown in Figure 7. Areas where the trajectories of the blue arrows coincide show areas where interesting behavior will arise. For example: collision avoidance, and breaking and re-forming robot formation.

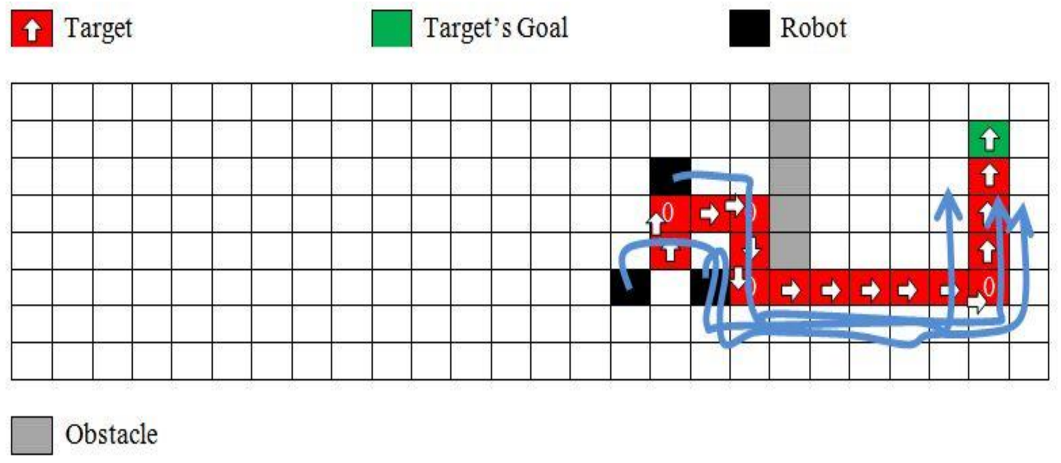

Fig. 8. Robot Motion Planning

10. FUTURE WORK

There are many directions for future work on this project. Several layers of complexity need to be added to answer the real world research questions which motivated the problem.

The robot motion planning presented in this report evaluates motion plans only based on coverage of the target's reachability set for a given Δt look-ahead. A useful step forward from this stage will be to incorporate a constrained optimization to guide the robot motion planning. The objective function for this constrained optimization can be used to control factors such as the distance between robots. With a constrained optimization approach, safety, feasibility, and reachability set coverage can be factored in using constraints.

Also, to gain a real world understanding, the assumption of certainty must be dropped. Uncertainty can be modelled in three aspects of this problem: (1) target action schema, (2) robot action schema, and (3) sensor model. One approach to handling uncertainty is to use the objective function of the aforementioned constrained optimization to minimize uncertainties. And, filtering must be used for accurate target state estimation in the face of noisy measurements.

In moving forward to pre-emptive motion plan with real systems, Kinematic/Dynamic models of robots and targets will have to be used in place of the discretized approximations used in this project. Specifically, the first step is to use real differential drive models for describing the behaviour of the target and the robots. This will allow for further development of a solution for implementation and experimentation using unmanned ground vehicles. Of course, even further in the future would be incorporating motion models for aerial vehicles for active target tracking using a team of unmanned aerial vehicles.

Finally, from an algorithmic stand point, an important aspect to explore for multi agent system is decentralization. In the case of this project, decentralizing the individual robot motion planning will bring up significant and important challenge to be tackled. First is the challenge of cooperative target state estimation, and second, coordinated planning. The core of the decentralization problem is that the robot motion plans are interdependent; hence, individual robots will have to form a team, communicating their beliefs and intentions to one another.

11. CONCLUSION

The lessons leaned over the course of this project will serve as a starting point for the planning aspects of my research project. Even though the results generated thus far don't seem to provide direct information regarding real robot behavior, the hope is that the fundamental principles will provide useful grounding points from which to continue building on top of the work accomplished in this project.

REFERENCES

[1] Kanayama, Yutaka, et al. "A stable tracking control method for an autonomous mobile robot." Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on. IEEE, 1990.

[2] Morbidi, Fabio, and Gian Luca Mariottini. "On active target tracking and cooperative localization for multiple aerial vehicles." Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011.

[3] Zhou, Ke, and Stergios I. Roumeliotis. "Multirobot active target tracking with combinations of relative observations." Robotics, IEEE Transactions on 27.4 (2011): 678-695.

[4] P. Yang, R. A. Freeman, and K. M. Lynch, "Distributed cooperative active sensing using consensus filters," in Proceedings of the IEEE International Conference on Robotics and Automation, Rome, Italy, Apr. 10-14 2007, pp. 405–410.

[5] J. P. Le Cadre, "Optimization of the observer motion for bearings-only target motion analysis," in Proceedings of the 36th IEEE Conference on Decision and Control, San Diego, CA, Dec. 10-12 1997, pp. 3126–3131.

[6] A. W. Stroupe and T. Balch, "Value-based action selection for observation with robot teams using probabilistic techniques," Robotics and Autonomous Systems, vol. 50, no. 2-3, pp. 85–97, Feb. 2005.

[7] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach, Chapters 3-4. Prentice Hall, 2nd edition, 2003.

[8] Fikes, Richard E., and Nils J. Nilsson. "STRIPS: A new approach to the application of theorem proving to problem solving." Artificial intelligence 2.3 (1972): 189-208.