2014

Path planning for autonomous vehicles

Mengzhe Zhang
Iowa State University

Path planning for autonomous vehicles

by

Mengzhe Zhang

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Iowa State University

Ames, Iowa

2014

DEDICATION

I would like to dedicate this thesis to my family and the people who had ever helped me with my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I want to express my sincere gratitude to those people who had helped me. First, I would like to thank my advisor, Dr. Sourabh Bhattacharya, who is insightful and can always find interesting problems. Without his patient supervision and guidance, I cannot address the problems and finish the thesis. The advices and guidance he offered are not simply the solutions, but the enlightenment, which is very helpful for me.

I am grateful to my committee members, Dr. Song Zhang, Dr. Yanbin Jia and Dr. Basak Tathagata for their assistance and suggestions which gave me a lot of help in my research. I also want to thank my lab mates, Rui Zou and Hamid Emadi, who work with me and provide me help along with the encouragements. In addition, my thanks are offered to the undergraduate students, Kin Gwn Lore, Ben Venturi, Tianshuang Gao and Yan Tian for helping me develop the hardware setup and conduct the experiments. To all my friends and my parents for helping me survive all the stress and not letting me give up.

ABSTRACT

In this study, we first address the problem of visibility-based target tracking for a team of mobile observers trying to track a team of mobile targets. Initially, we introduce the notion of pursuit fields for a single observer to track a single target around a corner based on the previous work. Pursuit fields are used to generate navigation strategies for a single observer. In order to account for the scenario with the presence of more than one observer or target, we propose a hierarchical approach. At first a ranking and aggregation technique is used for allocating each observer to a target. Subsequently, each observer computes its navigation strategy based on the results of the single observer-single target problem, thereby, decomposing a large multi-agent problem into numerous 2-agent problems. Based on the aforementioned analysis, we present a scalable algorithm that can accommodate an arbitrary number of observers and targets. The performance of this algorithm is evaluated based on simulation and implementation. To implement the strategy in reality, we further propose a setup of omni-directional camera, which can be used to get the visual information of the surroundings. With the help of this setup, we apply a position estimation technique for the pursuer to locate the evader. Experimental results show that the error has considerable effect when the measuring distance is very large. Due to this reason, the aforementioned tracking strategy is modified to keep the evader in an effective range for estimation. Finally, based on the error in position estimation, we present PID controllers for the pursuer to track the evader along a straight line. The responses of the proposed controllers are given by simulations.

Considering the situation that pursuer does not have an on board vision sensor, we propose a novel tracking strategy based on the information on social network. We first introduce the notion of common agents, who take pictures around and share them on social network website. In order to take advantage of these images, a network evolution algorithm and an image scanning algorithm are presented. Based on the information from these images, evader can be located accordingly. Implementation results are presented to validate the feasibility.

In the rest of the thesis, we address the scheduling and motion planning problem for an autonomous grain cart serving multiple combines. In the first part, we present the mathematical models of both combine harvester and grain cart. Based on the models, we propose a scheduling scheme which allows grain cart to unload all the combines without interruption in the harvesting activity. The proposed scheme is generalized to an arbitrary number of combines. In the second part, we present path planning analysis for the grain cart to switch between two combines. A numerical approach and a primitive-based approach are considered to obtain the time-optimal solution. The former approach needs a value function corresponding to the goal position to be computed beforehand. Based on the value function, a time-optimal path can be obtained accordingly. In the latter approach, path consists of singular primitives and regular primitives which ensure local time optimality. Finally, simulation results are presented to validate the feasibility of the proposed techniques.

CHAPTER 1. INTRODUCTION

1.1 Overview

In my thesis, we address a target tracking problem, as well as a logistics scheduling and motion planning problem.

At first, we present an extension on the previous target tracking work about a cell decomposition of the workspace based on strategies of the players. We consider the case with the presence of multiple pursuers and evaders in a general environment. To assign all pursuers for tracking tasks, we propose a ranking aggregation algorithm which unifies all available rankings and improve the pursuer assignment result. Moreover, we present a robot setup using omni-directional camera for tracking tasks. Based on the received images, evader's location is obtained by using position estimation technique. Uncertainty issue in locating the evader is discussed. Subsequently, we address a different pursuit-evasion game in which the information of evader comes from social website. We introduce the notion of common agents, who take pictures around and share them on social network website. In order to utilize such images, a network evolution algorithm and an image scanning algorithm are presented. Based on the information online, evader can be located accordingly. Implementation results are presented to validate the feasibility.

In the second problem, we address the logistics scheduling and motion planning of agricultural vehicles. A scheduling scheme for the grain cart to proceed unloading task in a general case is proposed. Regarding the motion planning of grain cart, we present a numerical approach and a primitive-based approach to obtain the time-optimal solution. Simulation results are provided to demonstrate the feasibility.

Next, we provide an outline of the thesis.

1.2 Thesis Outline

The thesis is outlined as follows. In Chapter 2, we present our work on visibility-based target tracking problem and social network target tracking problem. Simulations and implementations for both problems are posed. In Chapter 3, the work on logistics scheduling and motion planning for an autonomous agricultural vehicle is presented. In Chapter 4, a brief summary is presented, as well as the discussion of future work.

1.3 Publications

Published

- Mengzhe Zhang and Sourabh Bhattacharya (2014). Multi-agent Visibility Based Target Tracking Game. In *International Symposium on Distributed Autonomous Robotic Systems*, 440-451.
- Rui Zou, Mengzhe Zhang and Sourabh Bhattacharya (2014). Particle Swarm Optimization for Source Localization in Environment with Obstacles. In *IEEE Multi-conference on Systems and Control*, 1602-1607.

Accepted

- Mengzhe Zhang and Sourabh Bhattacharya (2014). Multi-agent Visibility Based Target Tracking Game. In *Distributed Autonomous Robotic Systems*.

Under review

- Mengzhe Zhang and Sourabh Bhattacharya (2015). Logistics Motion Planning for Agricultural Vehicles. In *ASCC 2015*.
- Mengzhe Zhang and Sourabh Bhattacharya (2015). Unloading Motion Planning for Agricultural Vehicles. In *IFAC MVS 2015*.
- Mengzhe Zhang and Sourabh Bhattacharya (2015). Logistics Scheduling and Motion Planning for Agricultural Vehicles. In *ICRA 2015*.

- Rui Zou, Mengzhe Zhang and Sourabh Bhattacharya (2015). Decentralized Multi-agent Visibility Based Target Tracking Game. In *ICRA 2015*.

Under preparation

- Mengzhe Zhang and Sourabh Bhattacharya (2015). Logistics scheduling and Motion Planning for Agricultural Vehicles. In *Journal of Dynamic Systems, Measurement, and Control*.

- Mengzhe Zhang, Rui Zou and Sourabh Bhattacharya (2015). Multi-agent Visibility Based Pursuit Evasion Game. In *Autonomous Robots*.

CHAPTER 2. TARGET TRACKING GAME

2.1 Introduction

Surveillance is an important problem in many application areas such as crime prevention, wildlife monitoring, traffic monitoring and industrial processes, etc. The development of surveillance technology facilitates government and law enforcement on maintaining social control, recognizing threats, and preventing criminal activities.

Among various branches of the study on surveillance problem, we focus on vision-based target tracking as our research. The problem is regarded as a pursuit-evasion game and subdivided into single observer-single target case and multi-observer multi-target case. In the following sections, we present an elaborated literature survey on both target tracking and pursuit-evasion game.

2.1.1 Target tracking

Target tracking is a special class of surveillance in which the intent is to keep track of the current state of a dynamic entity or an object of interest. Traditional target tracking methods include using sonar, radio or infrared sensors. Schulz et al. (2001) utilize two laser range sensors mounted on a mobile robot and get state information of the targets. Li et al. (2004) mount infrared sensors on the robot and apply a fuzzy target tracking control scheme for tracking task. Hollinger et al. (2012) use radio sensors to track a moving target radio node in the environment.

With the advent of computer vision, visibility-based target tracking gained a lot of interest in the research community. Papanikolopoulos et al. (1993) propose algorithms on real-time visual tracking of arbitrary 3D objects traveling at unknown velocities in a 2D space. PI (proportional-integral) controller, pole assignment controller and discrete steady-state Kalman filter are considered for the problem. LaValle et al. (1997a) present algorithms for maintaining visibility of a moving target. Depending on the information regarding the target's future actions, algorithms are proposed for a predictable as well as unpredictable target. Hu et al. (2004) give a comprehensive research survey on visual surveillance of object motion and behaviors. The review includes recent developments and general strategies of surveillance problem in different stages such as detection of motion, tracking, understanding and description of behaviors, etc. Kolling and Carpin (2007) present a distributed control algorithm which utilizes information from vision sensors, communication, and a mechanism to predict the minimum time before a robot loses a target. Tsalatsanis et al. (2007) use stereo vision and laser sensor in the proposed target tracking and collision avoidance method. Vision sensor is responsible for tracking predetermined or dynamically defined color so that target could be identified and the relative distance could be computed. In the work of Lee et al. (2012), vision-based object detection and tracking techniques for underwater robots have been studied in depth. Color restoration algorithm, detection and tracking method for underwater targets are discussed.

There has been a substantial amount of research to address the single observer-single target tracking problem. In the research of Gonzalez-Banos et al. (2002), a tracking algorithm is proposed that computes motion strategy for the observer based on the notion of escape risk and the computation of an escape-path tree. The escape-path tree is computed by taking local measurements, storing the most effective escape routes for a target to escape observer's field of view. Lee et al. (2002) propose an approach to keep moving target in the field of view without the assumption that a complete or partial model of the target's behavior is available. Bandyopadhyay et al. (2004, 2006, 2009) propose a tracking algorithm, and address stealth tracking problem based on visual sensors. A risk function is formulated to determine observer's direction of motion. A guaranteed tracking strategy is provided by Bhattacharya et al. (2007) for single observer-single target tracking problem in simple environments. The strategy is mainly based on the partition of the visibility polygon around a single corner. Al-Bluwi and Elnagar (2010a) tackle the tracking problem by considering three interacting components, namely tracking, collision avoidance and motion selection. The solution supports robot to tracking a moving target in a global dynamic environment. Anderson and Milutinovic

(2011) consider the problem of maintaining a nominal distance between a unmanned aerial vehicle and a ground-based moving target. A stochastic approach is introduced to address the problem.

In the recent years, there has been some efforts to address the problem when a team of observers is deployed for the tracking task. A distributed heuristic approach has been proposed by Parker (2002) based on local force vectors that causes robots to be attracted to nearby targets, and to be repelled from nearby robots. Jung and Sukhatme (2002, 2006, 2010) propose a region-based approach which controls the robot deployment by a coarse deployment controller and a target-following controller. An algorithm is presented that treats the densities of robots and targets as properties of the environment in which they are embedded. In the research of Frew and Elston (2008), a new task assignment algorithm that integrates area search and target tracking is presented. Frew (2009) further treats target tracking problem with ordered upwind methods, which can efficiently compute the target track boundary over a discretized nonuniform mesh of the environment. Hollinger et al. (2009) propose a scalable algorithm for efficiently solving the problem of planning paths for multiple robotic searchers trying to locate a non-adversarial target. Derenick et al. (2009) propose an optimization framework for dynamic target tracking by introducing the notion of weighted visibility graph. The framework guarantees that each target is tracked by at least one single team member. In the work of Lee et al. (2010), a distributed approach, which consists of two constituent algorithms: local interaction and target tracking, is presented to enable mobile robot swarms to track multiple targets moving unpredictably. A switching strategy is presented by Wu and Zhang (2013) for tracking a single target using mobile sensing agents that take bearings-only measurements. In the work of Ahmad et al. (2013), the problem of cooperative localization and target tracking is modeled within a team of agents as a least squares minimization problem, and show that it can be efficiently solved using sparse optimization methods. Xu et al. (2013) introduce learning models for target tracking, and propose a mechanism which reduces required communications among agents.

2.1.2 Pursuit-evasion game

In some cases, target tracking problem can be treated as a pursuit-evasion game. Pursuit-evasion game is a family of problems in which one team of members tries to capture another team in an environment. Observer and target in tracking problem could be interpreted as pursuer and evader, respectively. Pursuers attempt to catch evaders who, in turn, try to avoid being captured.

In recent years, search problem has received significant attention. The problem is originally introduced by Suzuki and Yamashita (1992) and could be described as follows. Given a 2D bounded Euclidean workspace, multiple pursuers equipped with vision sensors, an evader that has unknown position, pursuers would like to find a path which eventually leads them to the evader. During the process, pursuers are able to detect a subset of the environment based on their vision sensors. A large amount of approaches have been suggested to solve the problem. Under the work of LaValle et al. (1997b) and Guibas et al. (1997), control algorithms are proposed to guarantee that finally evader will lie in at least one visibility region of pursuers. Searching in unknown environment and with bounded speed are analyzed by Sachs et al. (2004) and Tovar and LaValle (2008), respectively. Gerkey et al. (2006) address search problem by introducing the notion of ϕ-searcher and present the analysis on searching multiple evaders in a given environment with limited field of view. Kolling and Carpin (2010) study the problem by modeling environment as a graph and let robot team execute sweep and block actions to detect intruders.

A similar problem in the research field is maintaining visibility, which could also be considered as a tracking problem. The problem is firstly suggested by LaValle et al. (1997b). Bhattacharya and Hutchinson (2008, 2011) study the case of which one pursuer and one evader present. The strategies are based on the result of partitioning environment into multiple regions. A sufficient condition of escape for the target is given. Murrieta-Cid et al. (2008) decompose the environment in another way. They define two graphs, namely mutual visibility graph and accessibility graph, for maintaining visibility and providing possible transitions of both pursuer and evader, respectively. Al-Bluwi and Elnagar (2010b) propose an approach to track single evader in a dynamic environment. The solution is carried out by using three separate components, namely, occlusion advisor, collision advisor and decision maker. Chung et al. (2011) present a detailed literature survey on search and pursuit-evasion in mobile robotics. They highlight the variances of pursuit-evasion games with different assumptions on searchers, targets and environment. Panagou and Kumar (2012) consider the problem as a leader-follower problem. They define a visibility set and translating the problem to controlling the robots so that the system trajectories start in the set and always remain in the set. This work is further extended to multiple agents by Panagou and Kumar (2014).

In the following sections, we present a visibility-based target tracking problem and a social network target tracking problem. For both problems, we provide tracking strategies which are validated by implementations. Next, we start elaborating the visibility-based problem.

2.2 Visibility-based Target Tracking Game

2.2.1 Problem description

The target tracking problem is formulated as follows. Consider a planar environment containing polygonal obstacles. The environment is assumed to contain two teams of mobile agents. One team of mobile agents, called the *observers (pursuers)*, are equipped with sensors for surveillance. In this work, we assume that the sensor mounted on top of each observer is an omni-directional camera having an infinite range. The objective of the team of observers is to track another team of mobile agents, called the *targets (evaders)*. By tracking, we mean that each member in the target team should be visible to at least one member in the observer team. Since the cameras are assumed to have an infinite range, the line-of-sight between the target and the observer is assumed to be only obstructed by the obstacles present in the environment. Let N_p and N_e denote the numbers of pursuers and evaders, respectively. Throughout this work, the term "observer" will be used interchangeably with the term "pursuer", and the term "target" will be used interchangeably with the term "evader".

Next, we describe the motion model for the mobile agents. In this work, we assume that all the agents have the following kinematic model

$$\dot{x} = u\cos\theta, \quad \dot{y} = u\sin\theta, \tag{2.1}$$

where (x, y) denotes the position of the agent in the plane, and u denotes the speed of agent. Let v_p and v_e denote the maximum speed of the pursuer and evader, respectively. In this work, we assume that all agents in each team possess the same maximum speed. Let a denote the ratio of their maximum speeds, i.e., $a = v_e/v_p$. Next, we describe the common information available to agents in each team. All agents in both teams are assumed to have a complete map of the environment, and are also assumed to have an exact knowledge of the positions of their team mates at all times, including themselves. Finally, we assume that each evader is visible to at least one pursuer initially. Given the initial positions of all the agents, we address the following question: What should be the strategy for the team of pursuers to track all the evaders for the maximum possible time?

In order to account for the worst-case scenarios, we assume that the evaders are adversarial in nature, and try to minimize the time required to escape from the region visible to the team of pursuers. For the simplest case, when all the agents have a complete information about the positions of all the players, the assumption of an antagonistic team of evaders leads to a two-person zero-sum differential game in between the two teams. Bhattacharya et al. (2007) have presented a thorough investigation of the problem for the special case when $N_p = N_e = 1$. A complete solution to the aforementioned case in general environments containing polygonal obstacles is still unknown (Bhattacharya and Hutchinson (2008)). However, the authors have provided a complete solution to the problem for specific environments. For example, in case of an environment containing a semi-infinite obstacle having one corner, a complete solution has been provided. In this work, we extend this work to propose algorithms for general environment containing polygonal obstacles that can handle arbitrary number of observers and targets. In the next section, we summarize the main results from Bhattacharya and Hutchinson (2011).

2.2.2 Cell decomposition around a corner

Bhattacharya and Hutchinson (2011) pose the target-tracking problem as a pursuit-evasion game of kind (Isaacs (1965)) in which the observer is modeled as a pursuer and the evader is modeled as a target. The pursuer wants to maintain the line-of-sight to the target, and the evader wants to break the line-of-sight in finite amount of time. The authors present a cell decomposition of the workspace around a corner based on the strategy used by the winner to ensure a successful outcome. Given the initial position of the pursuer, Figure 2.1 depicts the geometry of the individual cells or partitions, and Table 2.1 provides the strategies of the winner. Figure 2.2 depicts the same partitions computed by fixing the position of the evader. The pursuer wins the game in all partitions except for Region 1. However, the strategy used by the pursuer in Region 1 maximizes the time for which it can keep the evader in sight irrespective of the evader's strategy.

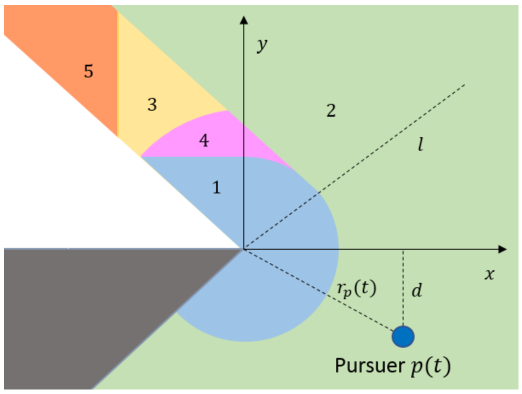


Figure 2.1: Pursuer-based partition

Table 2.1: Policies for the agents

Evader Policies	Evader Region	Control Law
A	1 and $\phi_e \in [\alpha - \pi, \frac{\pi}{2}]$	$\dot{r}_e(t) = -\overline{v}_e$
	1 and $\phi_e \in [\frac{\pi}{2}, \pi + \phi_p]$	$\dot{y}_e(t) = -\overline{v}_e$
Pursuer Policies	**Evader Region**	**Control Law**
B	2, 4	$\dot{y}_p(t) = \overline{v}_p$
C	3	$\mathbf{u}_t(t) = \frac{r_p(t)}{r_e(t)} \lvert\mathbf{v}_t(t)\rvert$
		$\mathbf{u}_r(t) = -\frac{r_p(t)}{r_e(t)} \lvert\mathbf{v}_r(t)\rvert$
D	5	$\mathbf{u}_t(t) = \overline{v}_p$

Based on the current position of the evader around the corner, the pursuer has an optimal direction of motion that maximizes the time for which the evader is visible depending on the

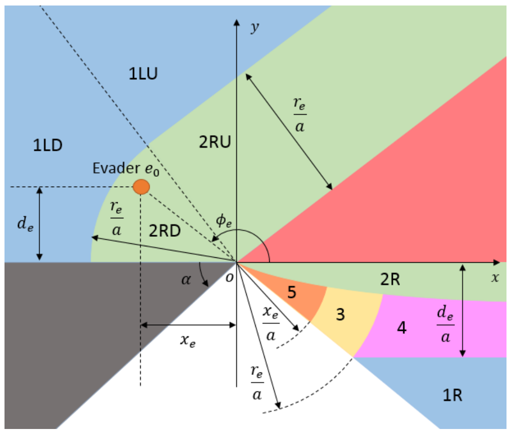


Figure 2.2: The partitions computed by fixing evader position

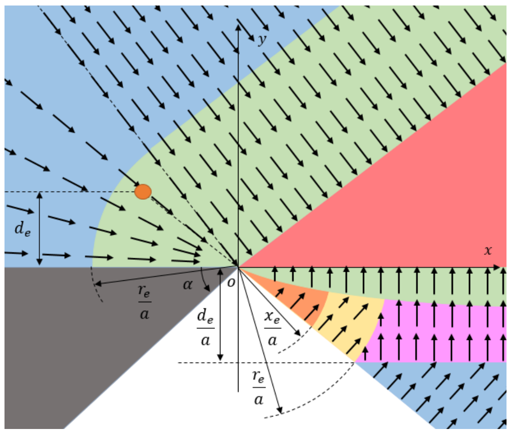


Figure 2.3: Pursuit fields with given evader position

Table 2.2: Vector fields for optimal navigation around a corner

Region	Vector Fields
1LU, 2RU	$\cos\alpha\frac{\partial}{\partial x} - \sin\alpha\frac{\partial}{\partial y}$
1LD, 2RD	$\cos\theta\frac{\partial}{\partial x} - \sin\theta\frac{\partial}{\partial y}$
1R	$-\sin\theta\frac{\partial}{\partial x} + \cos\theta\frac{\partial}{\partial y}$
2R	$\frac{\partial}{\partial y}$
3	$-\sin\theta\frac{\partial}{\partial x} + \cos\theta\frac{\partial}{\partial y}$
4	$\frac{\partial}{\partial y}$
5	$\frac{1}{\sqrt{x^2+y^2}}(-\sin\theta\frac{\partial}{\partial x} + \cos\theta\frac{\partial}{\partial y})$

partition in which it is placed. Since the optimal direction of motion at a point can be denoted by a vector, this generates a vector field in the environment that defines the optimal direction of motion for the pursuer at any instant in time. Since the positions of the pursuer and the evader change with time, the vector fields are time varying in nature. We use the term *Pursuit Fields* to represent these vector fields. The pursuer can navigate this time varying vector field to optimally track the evader. Figure 2.3 shows the vector fields for a given position of the pursuer and the evader. Table 2.2 provides the vector fields generated by the evader in the partitions. In the table, $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ represent the basis vectors of the tangent space at a given point.

2.2.3 Extension to general environments

2.2.3.1 Case 1: $N_p = N_e = 1$

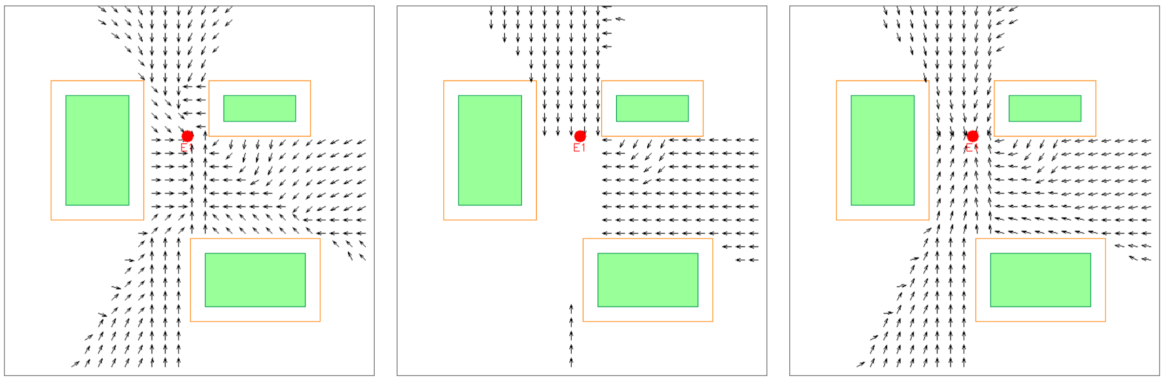
In this subsection, we use the results for a single corner in order to provide navigation strategies for the pursuer in general polygonal environments. From the previous section, one can construct pursuit fields based on the position of the evader around a corner. One can use the pursuit fields to generate pursuit strategies for environments containing multiple obstacles. A plausible way to do so is described next. The presence of an evader in the environment generates a set of pursuit fields, each corresponding to a corner visible to the pursuer around which the evader can escape. In order to generate the guidance law for the pursuer, one can use different metrics to obtain a resultant vector field from the set of pursuit fields. In this work, we use a weighted summation of the individual pursuit fields in order to compute the resultant vector field for pursuer navigation, i.e.,

$$v(\mathbf{x}) = \sum_{|C_i|} w_i v_i, \tag{2.2}$$

where C_i is the ith corner in the environment visible to the pursuer. The vector $\mathbf{w} = [w_1, \ldots, w_k]$ is called the *Risk Vector*. The ith element of the risk vector models the relative risk of the evader breaking the line-of-sight with the pursuer around the ith corner. Define d_i to be the distance between evader and the ith corner, we consider the following metrics for measuring risk:

1. Uniform risk: $w_i = \frac{1}{k}$
2. Majority risk: $w_i = 1$, where $i = \arg\max \frac{1}{d_i}$
3. Proportional risk: $w_i = \dfrac{d_i^{-1}}{\sum_{|C_j|} d_j^{-1}}$

The uniform risk vector provides equal weight to all the corners visible to the pursuer around which the evader can escape. The majority risk vector only takes into account the pursuit field generated by the corner that is nearest to the evader. The proportional risk vector assigns a weight to each corner that depends on the proximity of the evader from the corner. Figure 2.5 shows the vector fields that are generated for an environment using the three different techniques. The strategy of the pursuer at any given instant is to navigate along the vector field based on a chosen risk vector. Figure 2.6 shows the trajectories of the pursuer generated by the three risk vectors for a fixed evader trajectory. Red dots stand for the trajectory of evader. Green, black and blue dots show the trajectories of pursuers using uniform, majority and proportional risk, respectively. Three pursuers are placed at the same starting position.



(a) Uniform risk (b) Majority risk (c) Proportional risk

Figure 2.5: Vector fields

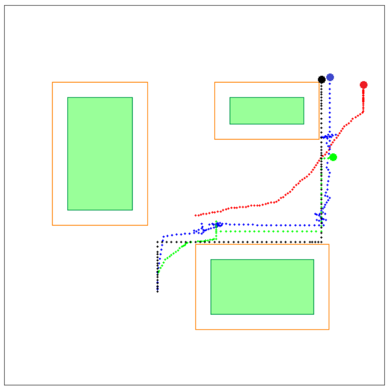


Figure 2.6: Trajectories of pursuers using uniform, majority and proportional risk

2.2.3.2 Case 2: $N_p > 1$, $N_e > 1$

In this subsection, we present an extension of the previous technique for the case $N_p > 1$, $N_e > 1$. Initially, we discuss the case when all the pursuers use the same risk function to compute the risk associated with a team of targets, following which, we discuss the case when each pursuer may have its own risk function.

First, let us analyze the case when all pursuers have the same risk function. For a given pursuer and evader, one can compute the risk of evasion directly. For two teams of pursuers and evaders, one can compute the risks associated with each pair of pursuer and evader, and use the Hungarian algorithm (Kuhn (1955)) in order to compute the minimum matching. This works perfectly if $N_p = N_e$. If $N_p > N_e$, we have some pursuers that remain unassigned for the tracking task. In this case, there can be several ways to solve the reassignment problem. In our implementation, for each pursuer that remains unassigned, we let it stay stationary until it is included in the output of Hungarian algorithm. On the other hand, if $N_p < N_e$, there are some evaders that might not be assigned to any pursuer. As long as they are visible to at least one pursuer, they are being tracked by definition.

Next we discuss the case when all pursuers do not use the same risk function. In this case, it is not possible to use the above technique since the values of risk generated by all pursuers may not be comparable. In order to resolve this issue, we propose a ranking and aggregation algorithm. Each pursuer ranks the evaders that are in its field of view. Based on the risk function, pursuer i will have a sequence of risks associated with each evader in its field of view. By sorting this sequence of risks, the pursuer obtains a ranking for the evaders. Therefore, each pursuer will have a ranking of the evaders in its visibility polygon. In order to aggregate the ranking obtained from different pursuers, we use the Borda Count (Emerson (2013)) corresponding to each evader. In each ranking, points are allocated to an evader based on its position in the ranking. Finally, the cumulative score for an evader is obtained by adding the points obtained from all the rankings. All the evaders present in the environment can be ranked based on this score. This provides an aggregation scheme to generate a unique ranking of the evaders from the individual preference of the pursuers.

After the aggregation stage, pursuers are allocated to the evaders based on the final scores obtained from the Borda Counts. First, we define $S \in \mathbb{R}^{N_p} \times \mathbb{R}^{N_e}$ as follows:

$$S_{ij} = \begin{cases} \text{Borda Score of Evader } j & \text{If evader } j \text{ is visible to pursuer } i \\ \infty & \text{If evader } j \text{ is not visible to pursuer } i \end{cases} \tag{2.3}$$

We apply Hungarian algorithm to S, and obtain an assignment minimizing the total score. Based on the result, we allocate pursuers to the targets. If $N_p \leq N_e$, we have a matching which leaves no pursuer unassigned. However, if $N_p > N_e$, we assign to each unassigned pursuer an evader that is in its field of view, and has the maximum non-infinity Borda score. This reflects the strategy that these remaining pursuers try to track the most risky evaders. The complete algorithm is presented in Algorithm 1.

2.2.4 Position estimation algorithm

In the aforementioned strategy, it requires the camera to have infinite range of sight and accurate positions of evaders which is not reasonable in real life due to the limitations on camera. For this reason, we present an omni-directional camera setup which can be used in real experiments on visibility-based tracking. By using this setup, we will further apply a position estimation technique for locating the evader.

2.2.4.1 Omni-camera setup

In order to have the omni-directional view, we consider the following setup, which consists of a pursuer (viewed from the side) and an evader (view from the front).

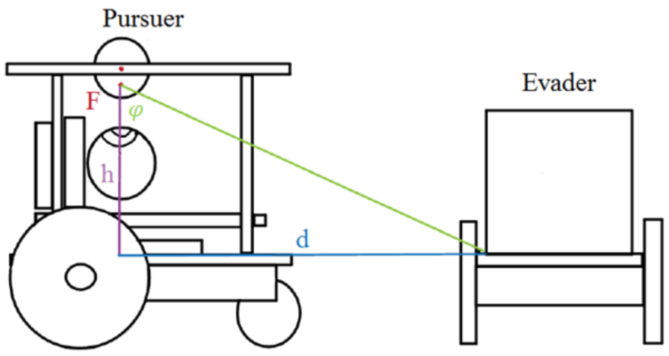


Figure 2.7: Omni-camera setup

The focal length is obtained by dividing the diameter of the reflective sphere by four (i.e. $F = D/4$). This value can be used to calculate the height h of the focal length from the bottom stage. The angle φ between the vertical axis and the reflected tracing ray to the bottom edge of the evaders colored shell, could be used to compute the quantity d, which is the actual distance from the pursuers camera to the evaders sides. With the camera mounted, we can obtain the output images as shown in Figure 2.8 and 2.9:

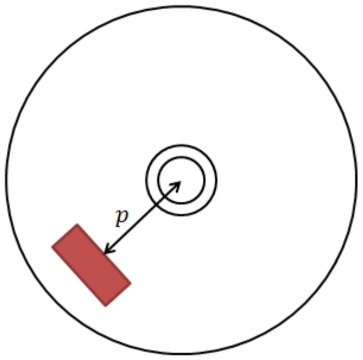


Figure 2.8: Omni-camera image sketch

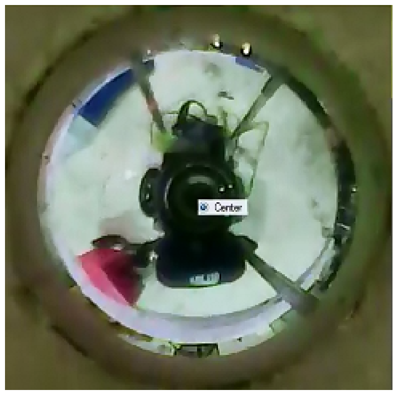


Figure 2.9: Omni-camera image

Based on the geometric relationships shown in Figure 2.7, we know that

$$d = h\tan\varphi \approx h\tan\frac{p}{k} \tag{2.4}$$

where k is a constant value that depends on the factors such as mirror shape and distance of the camera from the reflective surface (Scaramuzza (2008)). The value of k needs to be experimentally calibrated before we can use this equation to estimate the distance. In the next subsection, we describe the procedures for calibrating k.

2.2.4.2 Calibration procedures for k

The procedures of calibration could be described as follows:

1. Place the evader 15 inches away from the center of the camera.
2. Acquire the image from the camera.
3. Measure the pixel distance between the center of the camera to the sides of the evader from the acquired image. This will be the value p.
4. Repeat procedures 1-3 with distances of 20, 25, 30, 35, 40, 45, 50, 55 and 60 inches.
5. Compute the k value corresponding to each distance and use the mean value as the final k.

Based on the experimental result, we set $k = 107$. Once this has been done, we can measure the pixel distance to the object in the image plane and have the equation output the estimated actual distance. Considering the noise in the obtained images, we apply dilation and erosion to the original image and finally use a color threshold to find the target.

Figure 2.10 illustrates the performance of distance estimation technique when evader locates in different direction to the camera. We compute the standard deviation of estimated distance within all directions, as shown in Figure 2.12. Figure 2.11 depicts the absolute error between the average of estimated distance and real distance.

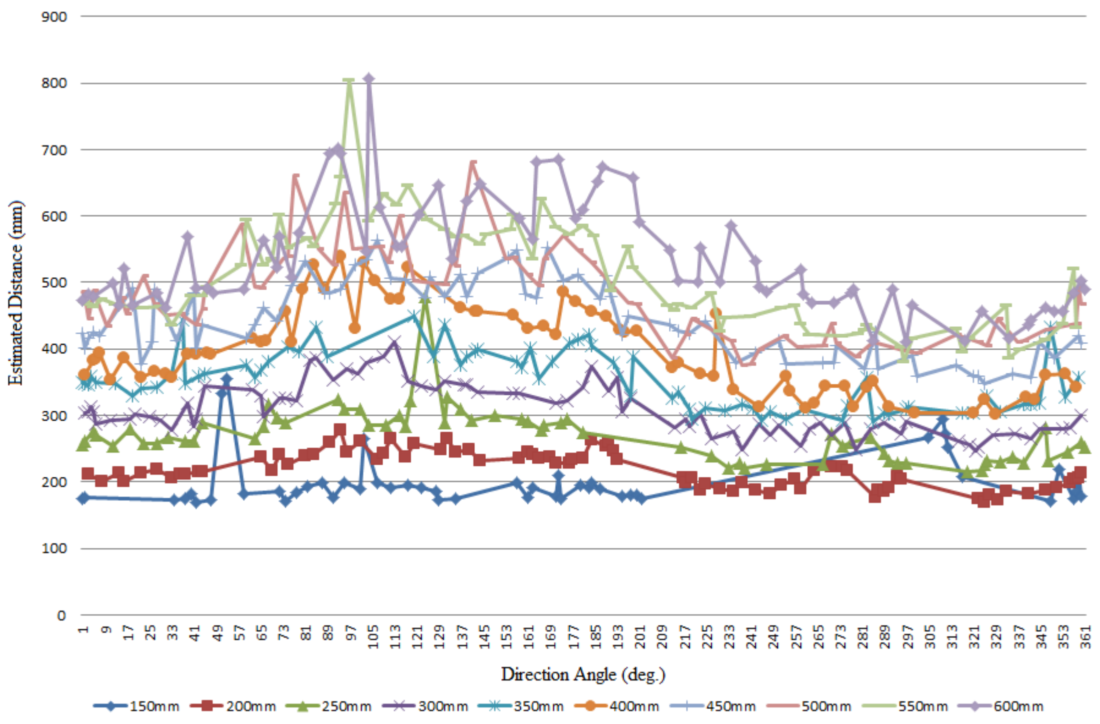
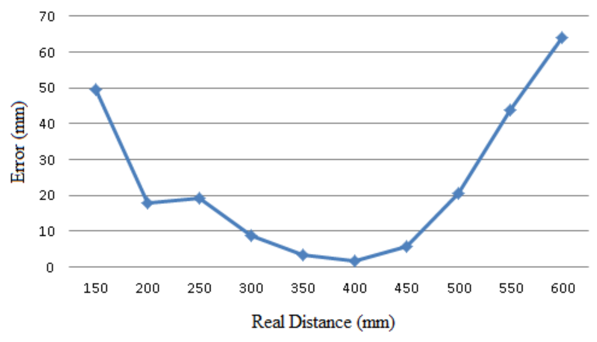


Figure 2.10: Estimated distance with respect to direction angle

2.2.4.3 Evader localization

Since the orientation of the pursuer in the actual world, denoted by θ_p, is a prior knowledge to us, we can calculate the bearing to the evader from the front of the mobile robot, denoted



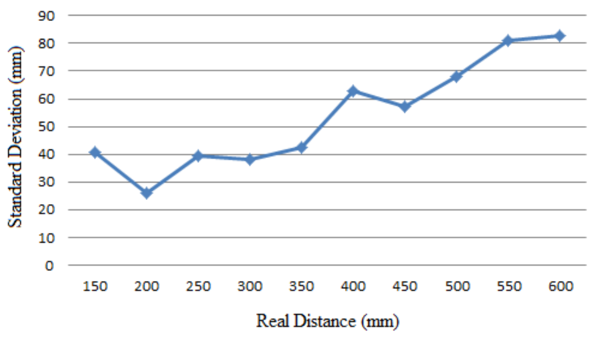


Figure 2.11: Absolute error of estimated distance with respect to real distance

Figure 2.12: Standard deviation of esitmated distnace with respect to real distance

by α, based on the position of the detected evader in the image plane. Thus,

$$\theta_e = \theta_p + \alpha \tag{2.5}$$

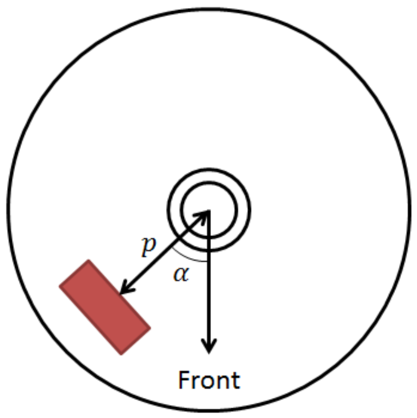


Figure 2.13: Omni-camera image sketch

The position of the evader could be estimated as follows

$$x_e = x_p + d\cos\theta_e, \quad y_e = y_p + d\sin\theta_e \tag{2.6}$$

2.2.5 Modified pursuer strategy

From experimental results, we have learned that notable errors occur when the measuring distance between camera and target is too large. To resolve the issue, we add a vector towards evader, denoted as v', to the original vector field, denoted as v_o. In particular, the motion of pursuer could be described as

$$v = wv_o + (1-w)v' \tag{2.7}$$

where $0 \leq w \leq 1$ is a weight parameter.

As shown in Figure 2.14, the modified strategy aims to track evader and meanwhile keep evader in an effective range of sight for position estimation.

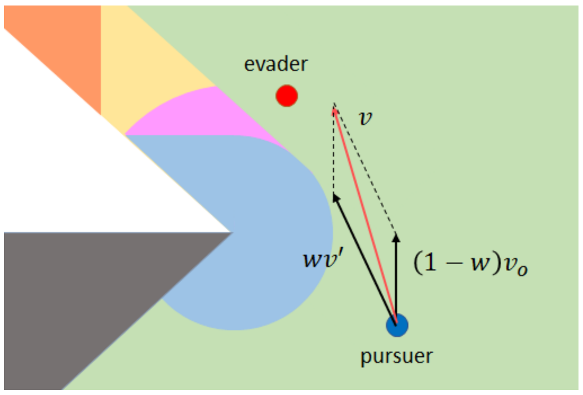


Figure 2.14: Vector combination

2.2.6 PID controller design

At last, we consider the problem that pursuer tracking evader along a straight line. In this simplified scenario, we would like to design a PID controller for the pursuer to maintain a desired distance r with the evader. Denote the displacements of pursuer and evader as x_p and x_e, respectively, Figure 2.15 shows the aforementioned pursuit-evasion system.

According to the data illustrated in Figure 2.11, we fit the error as a parabola as follows

$$e(x_e - x_p) = a(x_e - x_p)^2 + b(x_e - x_p) + c \tag{2.8}$$

where a, b and c are constants.

Let $\tilde{x_e}$ to be the estimated x_e and y to be $(x_e - x_p)$, we design P, PI, PD and PID controllers for the pursuer to perform tracking task. In the following designs, K_P, K_I and K_D are used to denote the proportional, integral and derivative constants, respectively.

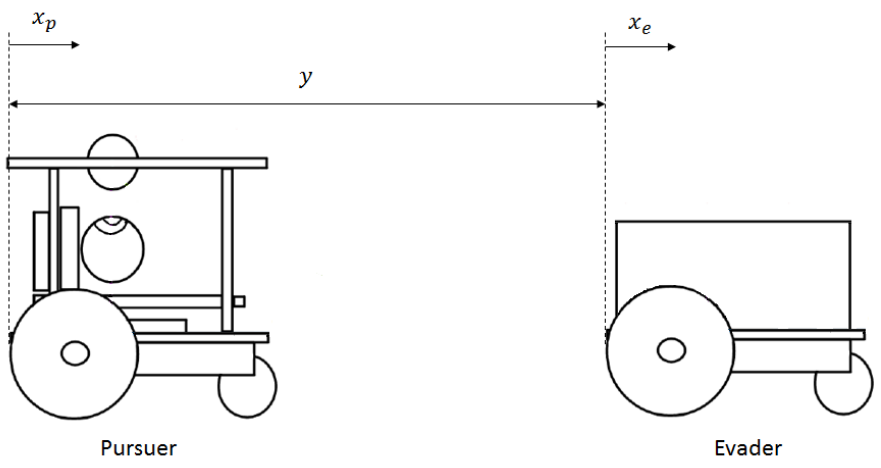


Figure 2.15: System of pursuer and evader

2.2.6.1 P controller

$$
\begin{aligned}
\dot{x_p} =& \dot{x_e} + K_P[(\tilde{x_e} - x_p) - r] \\
\dot{x_p} =& \dot{x_e} + K_P[(x_e - x_p) - (x_e - \hat{x_e}) - r] \\
\dot{x_p} =& \dot{x_e} + K_P[(x_e - x_p) - e(x_e - x_p) - r] \\
-(x_e \dot{-} x_p) =& K_P[(x_e - x_p) - e(x_e - x_p) - r] \\
\dot{y} =& K_P[ay^2 + (b-1)y + c + r] \qquad (2.9)
\end{aligned}
$$

2.2.6.2 PI controller

$$
\begin{aligned}
\dot{x_p} =& \dot{x_e} + K_P[(\tilde{x_e} - x_p) - r] + K_I \int_0^t [(\tilde{x_e} - x_p) - r]dt \\
\ddot{y} =& K_P[ay^2 + (b-1)y + c + r] + K_I \int_0^t [ay^2 + (b-1)y + c + r]dt \\
\ddot{y} =& K_P[2ay\dot{y} + (b-1)\dot{y}] + K_I[ay^2 + (b-1)y + c + r] \qquad (2.10)
\end{aligned}
$$

2.2.6.3 PD controller

$$\begin{aligned}
\dot{x_p} =& \dot{x_e} + K_P[(\tilde{x_e} - x_p) - r] + K_D \frac{d[(\tilde{x_e} - x_p) - r]}{dt} \\
\dot{y} =& K_P[ay^2 + (b-1)y + c + r] + K_D \frac{d[ay^2 + (b-1)y + c + r]}{dt} \\
\dot{y} =& K_P[ay^2 + (b-1)y + c + r] + K_D[2ay\dot{y} + (b-1)\dot{y}] \\
\dot{y} =& \frac{K_P[ay^2 + (b-1)y + c + r]}{1 - 2aK_D y - K_D(b-1)} \qquad (2.11)
\end{aligned}$$

2.2.6.4 PID controller

$$\begin{aligned}
\dot{x_p} =& \dot{x_e} + K_P[(\tilde{x_e} - x_p) - r] + K_I \int_0^t [(\tilde{x_e} - x_p) - r]dt + K_D \frac{d[(\tilde{x_e} - x_p) - r]}{dt} \\
\dot{y} =& K_P[ay^2 + (b-1)y + c + r] + K_I \int_0^t [ay^2 + (b-1)y + c + r]dt + K_D \frac{d[ay^2 + (b-1)y + c + r]}{dt} \\
\ddot{y} =& K_P[2ay\dot{y} + (b-1)\dot{y}] + K_I[ay^2 + (b-1)y + c + r] + K_D[2a\dot{y}^2 + 2ay\ddot{y} + (b-1)\ddot{y}] \\
\ddot{y} =& \frac{K_P[2ay\dot{y} + (b-1)\dot{y}] + K_I[ay^2 + (b-1)y + c + r] + 2aK_D\dot{y}^2}{1 - 2aK_D y - (b-1)} \qquad (2.12)
\end{aligned}$$

2.2.7 Simulation

In this subsection, we present the simulation results using the original strategy. Simulations are conducted within a 4000×4000 environment with rectangular obstacles. t_1, t_2 and t_3 denote the time for which the pursuers can track the evaders using the three risk vectors above. In our simulations, we choose $a = 0.3$ and $v_p = 300$. The following simulations are done for random initial positions of the agents with the constraint that every evader is visible to at least one pursuer.

In the first simulation, we consider using the same risk function for computing the risk of each pair of pursuer and evader. The risk function is described as follows

$$r_{ij} = \sum_{|C_k|} d_{jk}^{-1}. \qquad (2.13)$$

where C_k is the kth corner which is visible to both pursuer i and evader j, d_{jk} denotes the distance between evader j and corner k. Figure 2.16, 2.17 and 2.18 show the average values of t_1, t_2 and t_3 for 1000 simulations as we change the number of pursuers and evaders in the environment. These histograms indicate that the proposed strategies offer pursuers good tracking performances. When $N_p \geq N_e$, all the values of t_1 are greater than 50 time units. The minimum average value of t_1 is 74.62 units when $N_p = 1$ and $N_e = 4$, which shows that the strategies can still work when more evaders present. From the simulation results, we can see that with a given number of pursuer, the tracking time decreases when the number of evader increases. Similarly, fixing the number of evader, tracking time increases when the number of pursuer increases. Furthermore, as the number of agents increases, the variances between t_1, t_2 and t_3 increase, which means using different risk vectors has a growing influence on the tracking time.

In the second simulation, we consider the case of three pursuers using three different risk functions. Figure 2.19 shows the average tracking time of using ranking and aggregation algorithm for 1000 simulations. As a reference, we arrange another two teams of pursuers in the game. Each team has three pursuers, using the same risk functions with the first team. Pursuers in one team are assigned randomly to their visible targets, pursuers in the other team are assigned to the most risky evader according to their own ranking results. When $N_e = 1$, all the teams of pursuers reach the maximum time we count. But for multiple evaders, especially when $N_p \geq N_e$, ranking and aggregation algorithm shows a better performance.

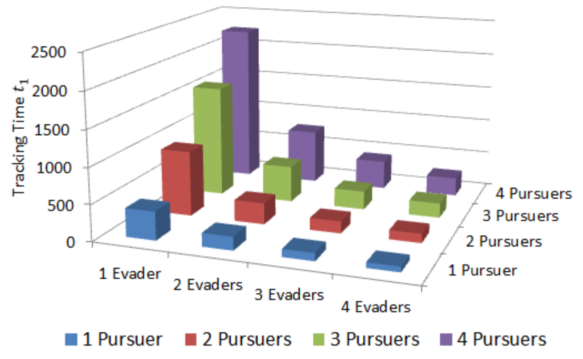


Figure 2.16: Tracking time t_1 of pursuers using uniform risk vector.

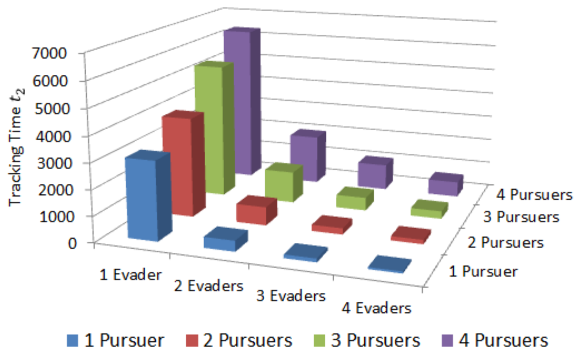


Figure 2.17: Tracking time t_2 of pursuers using majority risk vector.

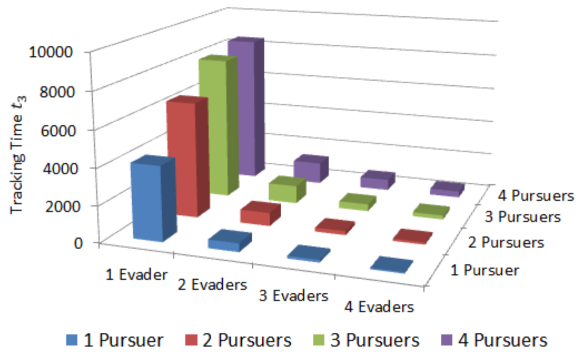


Figure 2.18: Tracking time t_3 of pursuers using proportional risk vector.

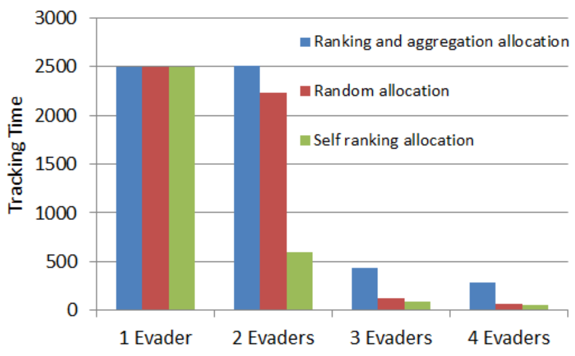


Figure 2.19: Tracking time with different allocations

In the last simulation, we plot the responses of aforementioned P, PI, PD and PID controllers. We take desired distance $r = 200$. Based on the data of error in Figure 2.11, we apply constants $a = 0.001062$, $b = -0.7598$ and $c = 137$. For the initial conditions, we use $y(0) = 100$ and $\dot{y}(0) = 0$. Figure 2.20 illustrates the performances of these controllers with PID constants $K_P = 10$, $K_I = 5$ and $K_D = 1$.

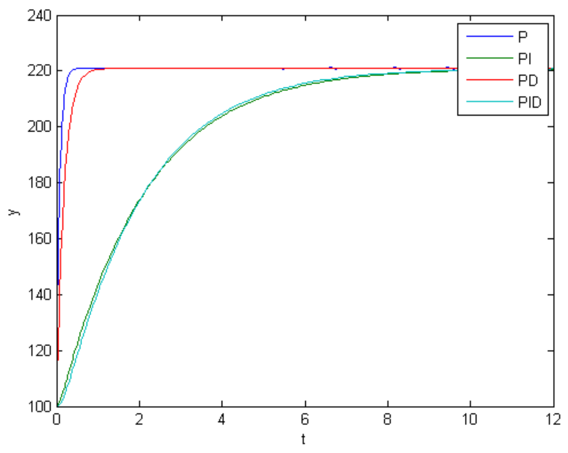


Figure 2.20: Responses of PID controllers

2.2.8 Implementation

Next, we present a description of the experimental testbed that is used to implement the strategies above. As shown in Figure 2.23, the environment is a 4000 mm $\times$ 4000 mm plane.

All obstacles are rectangular with four corners. Vicon tracking system is used for collecting real-time positions of all agents. Both pursuers and evaders are differential drive ground robots. Figure 2.21 and 2.22 show the normal ground robot and omni-cam robot setup, respectively. Each robot is equipped with an Arduino Mega 2560 board as the controller, and a XBee module as the communication unit. In each iteration, a work station collects all the data and computes the control of each pursuer based on the proposed strategies. The commands are sent through XBee module accordingly.

Figure 2.24 shows the user interface written for the experiments. The bottom left monitor shows visibility relationships among pursuers and evaders. A line-of-sight and a chase between a pursuer and an evader are denoted by a black edge and a purple edge, respectively. The mini-map on the right shows the motions of all agents. Green rectangles are obstacles, with orange boxes around them being the actual boundaries in case of collision with robots. Offset between orange boxes and obstacles is 160 mm. Next, we will present the experimental results.

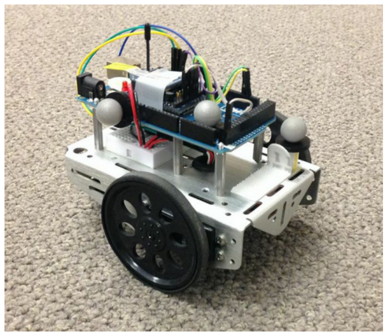


Figure 2.21: Ground robot

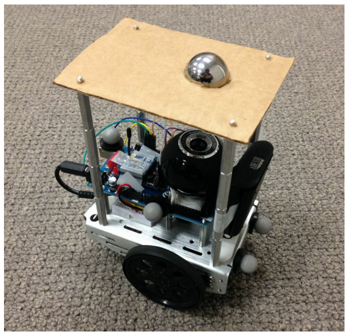


Figure 2.22: Omni-cam setup on real robot

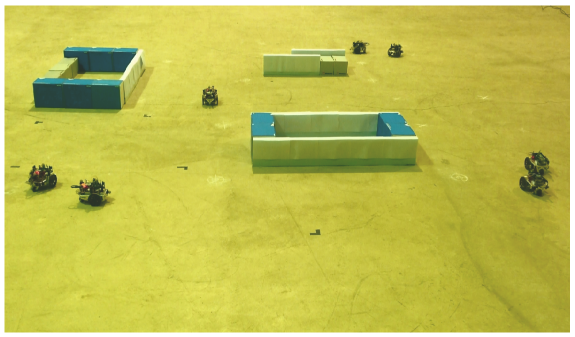


Figure 2.23: Implementation setup

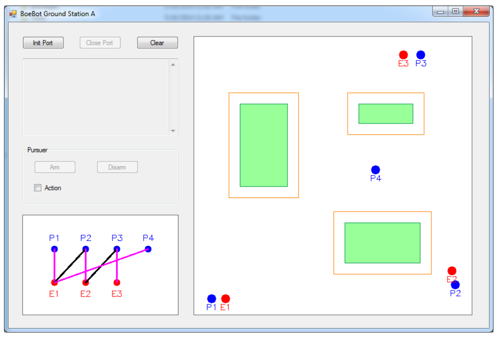


Figure 2.24: User interface

2.2.8.1 Original tracking strategy using the same risk function

In this part, we present the implementation result of the strategy proposed in Section 2.2.3. We consider three scenarios using Equation (2.13) as risk function for all pursuers.

- $N_p < N_e$: In the scenario of $N_p = 3$ and $N_e = 4$, each pursuer is allocated to one evader based on the result of Hungarian algorithm. All the pursuers can successfully follow their targets according to proposed moving strategies. Once the matching changes, pursuer switches its target accordingly.
- $N_p = N_e$: In the scenario of $N_p = N_e = 3$, pursuers have different targets, and all of them are able to follow their assigned targets.
- $N_p > N_e$: In the scenario of $N_p = 4$ and $N_e = 3$, three evaders are tracked by three pursuers initially, and one pursuer remains unassigned. When the matching changes, the stationary pursuer takes over and starts moving. In this case, all the evaders are tracked and algorithm can automatically choose the best combination of pursuers for tracking assignments.

2.2.8.2 Original tracking strategy using different risk functions

In this case, we implement ranking and aggregation algorithm by letting $N_p = 4$ and $N_e = 3$. Figure 2.25 indicates the implementation result which shows that the algorithm can appropriately allocate all pursuers, including one extra pursuer. In addition, all the pursuers are able to follow their targets.

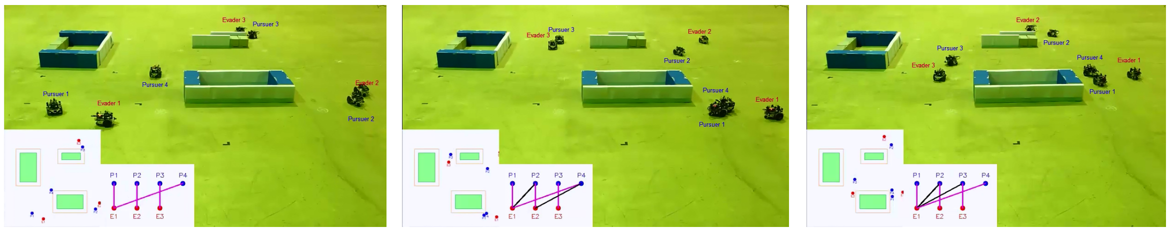


Figure 2.25: Implementation of original tracking strategy with 4 pursuers and 3 evaders using different risk functions

2.2.8.3 Modified tracking strategy

To demonstrate the feasibility of evader localization method and modified pursuer strategy, we implement them with a single pursuer and a single evader. In the experiment, pursuer uses the setup presented in Subsection 2.2.4.1 (Figure 2.22). we set the weight parameter $w = 0.5$. Figure 2.26 shows the implementation result, which shows that pursuer can locate evader and follow it effectively. To facilitate observation, we attach the image from omni-camera and a mini-map on the bottom of each figure.

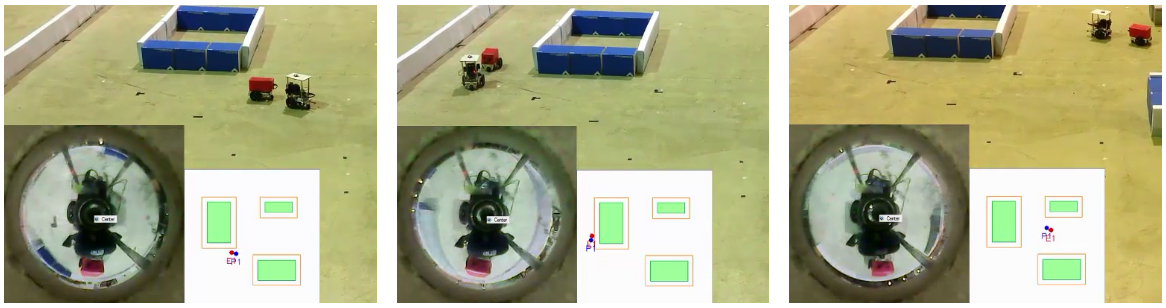


Figure 2.26: Implementation of modified tracking strategy with single pursuer and single evader

The case of 2 pursuers chasing 2 evaders is implemented as shown in Figure 2.27. Red and green shells are used to distinguish the two evaders. During the experiment, both pursuers are able to trail their targets without interruption.

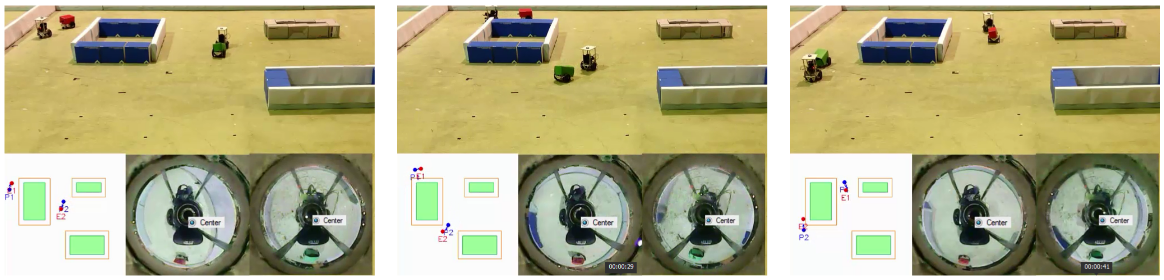


Figure 2.27: Implementation of modified tracking strategy with 2 pursuers and 2 evaders

2.2.9 Conclusion

In this section, the problem of visibility-based target tracking for a team of mobile observers trying to track a team of mobile targets was addressed. Initially, we introduced the notion of *pursuit fields* for a single observer to track a single target around a corner based on the results obtained by Bhattacharya and Hutchinson (2011). We used the pursuit fields to generate navigation strategies for a single observer. In order to tackle the scenario when more than one observer or target was present, we proposed a hierarchical approach. At first a ranking and aggregation technique was used for allocating each observer to a target. Subsequently, each observer computed its navigation strategy based on the results of the single observer-single target problem, thereby, decomposing a large multi-agent problem into numerous 2-agent problems. Based on the aforementioned analysis, we presented a scalable algorithm that can accommodate an arbitrary number of observers and targets. The performance of this algorithm was evaluated based on simulation and implementation.

We further considered the real situation of which each pursuer was equipped with a finite range omni-camera. We applied a position estimation technique based on the images from omni-camera. Due to the error caused by large relative distance, we proposed a modified navigation strategy which lets pursuer be able to keep evader in an effective observing range. Additionally, we designed PID controllers for the pursuer to track evader based on the experimental data on the error of position estimation. The proposed algorithm and strategy were validated by implementation with real robots.

2.3 Social Network Target Tracking Game

In the previous section, the positions of evaders are obtained by either global tracking system or omni-cameras. But in reality, we do not always have such devices or technologies around. In other words, if a pursuer does not have vision sensors on-board, it cannot locate evader's position. Therefore, previous strategies are not feasible at this point. A similar situation occurs when pursuer cannot see evader initially. The tracking strategies cannot be applied since pursuer does not have any information regarding evader. To address this problem, we consider taking advantage of the Internet and letting pursuer to collect available online information. One way to do so is to search on the social network website. Next, we will formulate the problem and present the approach we use for this problem.

2.3.1 Problem description

Consider a planar environment with multiple obstacles and landmarks. Obstacles are assumed to be polygonal and can occlude visibility. Landmarks are unique and fixed in the environment. The environment contains one pursuer, one evader and multiple agents, namely *common agents*, who are equipped with vision sensors. These sensors are assumed to have a limited visual angle. Common agents can only use their vision sensors to take pictures around. Both pursuer and common agents have access to the Internet. Once the pictures are taken, common agents will post the pictures on the social website without any time delay. This models the situation that people post their pictures to social website in real life. The objective of pursuer is to find evader according to the online information and track its path.

In this problem, we assume all the mobile agents have the same model as described in previous section. We consider the entire map of the environment, including the information of landmarks, as prior knowledge to all agents. We also assume that pursuer and evader know their own positions at all times. The initial positions of all agents are given randomly so that evader may not be seen by any agent at the beginning. So we would like to find an Internet-based approach which can lead pursuer to track evader's path. In the following sections, we present the approach for searching evader.

2.3.2 Searching approach based on social network

From previous section, we know that the pictures taken from common agents are kept under their online social network accounts. With this in mind, we would like to let pursuer get access to these pictures. So in the first step, pursuer needs to send friend requests on social website in order to get access. Let F denote the set of friend agents. At the beginning of network evolution, pursuer would like to send requests to new common agents. Depending on the responses, the requested agents can be placed into three categories. The first one represents those who accept. In this case, the agents can be added to F directly. In the second category, we have those agents who reject the requests. To tackle this case, pursuer will try to resend friend requests. The last category contains those agents who don't respond. For each of them, we set up an counter. If the counter exceeds the maximum waiting time T_{wait}, we look for new common agent instead of keep waiting the respond. Note that the counter also ticks when an agent rejects friend request. This ensures that the pursuer will pass the uncooperative agent when it keeps rejecting. Considering the energy capacity of pursuer, we constrain the number of requests sent in one iteration by N_{req}, and the total number of friend agents by N_f. In other words, pursuer will not send more than N_{req} requests at one iteration and will not have more that N_f friends. Furthermore, friend agents may unfriend with pursuer at any time instant. At this point we still proceed with the network evolution algorithm. The algorithm regards these leaving agents as new common agents, and may try to connect them again. The complete network evolution algorithm can be found in Algorithm 2.

In the next stage, pursuer proceeds by accessing social website accounts of its friends. For each account, pursuer scans the most recent N_{img} pictures for detecting evader and landmarks. The complete scanning algorithm is presented in Algorithm 3. In one iteration, if evader and any landmark are found in the same picture, evader can be located based on the known information. The localization method will be elaborated in the next section.

2.3.3 Evader localization

In this section, we present the evader localization method based on image information obtained from common agents. In order to figure out evader's position, we have to find the location of the common agent who took the picture (i.e. the position of the camera). With this in mind, we analyze the received image, and first compute the distance d_1 between one vertical edge of the landmark and camera, based on the following equation.

$$d_1 = f * (d_{landmark}/(n_{edge}/k_{ppm})) \tag{2.14}$$

where f is the focal length of camera, $d_{landmark}$ is the real length of landmark's edge, n_{edge} is the number of pixels of the edge on image plane, and k_{ppm} is the pixel density.

By using two vertical edges of the landmark, we are able to derive camera's relative position to the landmark by geometry, as shown in Figure 2.28. To get the orientation, we measure the number of pixels that one edge deviates from image's center on the image plane, denoted as

n_{dev}. Then orientation offset θ between one edge to camera's front can be derived as follows

$$\theta = atan(d_{dev}/f) \tag{2.15}$$

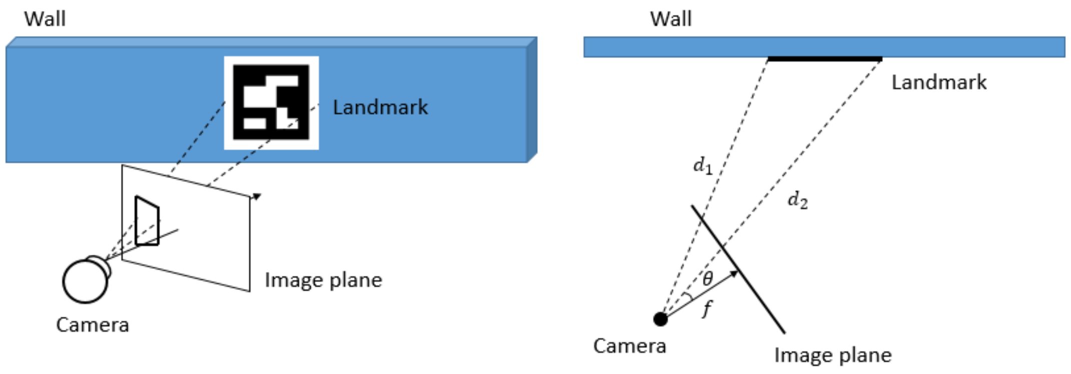


Figure 2.28: Common agent/camera localization

With real distances d_1, d_2 and angle θ, we are able to obtain the absolute position and orientation of the camera. Similarly, by measuring the size of evader in picture, we can further get the relations between the positions of the evader and the camera. Since the camera has been located, evader's location can be derived straightforwardly.

2.3.4 Implementation

In this section, we present the implementation results. The environment is an $8ft \times 8ft$ planar maze containing obstacles. Thirteen landmarks are places with known positions and all of them are unique Chinese characters.

Due to the limitations on space and hardware, we consider a simplified case with one pursuer, one evader, and three common agents. All of them are differential drive robots. Compared with the robots used in Section 2.2, we additionally mount four infrared sensors on pursuer and each common agent for collision avoidance. To model the activity of taking pictures, each common agent is equipped with a wireless camera so that it can capture images of the scenes on the front. Evader is attached with several identical markers, as shown in Figure 2.32, for detection. All the robots are placed with random initial positions.

To facilitate our experiment, a user interface program, as shown in Figure 2.30, is designed. Mini-map on the right shows real-time position of the pursuer, which is collected from Vicon tracking system. Blue lines are the boundaries of obstacles. Considering the obstacle avoidance problem, the boundaries of obstacles (green lines) are offset in path planning computation. Red disk and red lines indicates the location and trajectory of the pursuer, respectively.

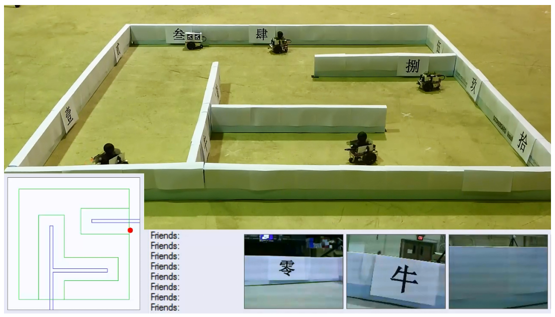


Figure 2.29: Experiment setup

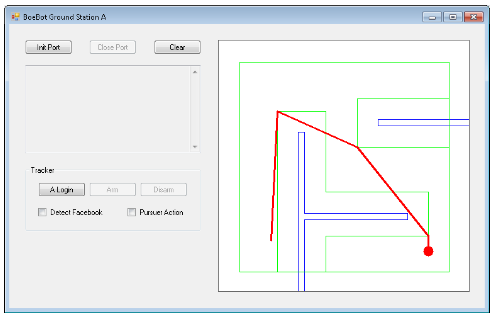


Figure 2.30: User interface

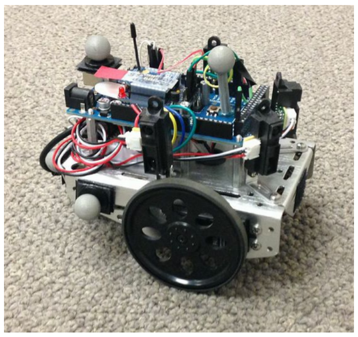


Figure 2.31: Pursuer with infrared sensors

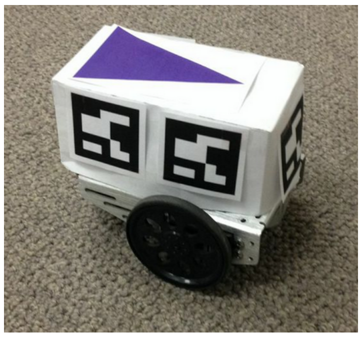


Figure 2.32: Evader with a unique marker

2.3.4.1 Social activities

We implement our communication method among robots using Facebook website, which is one of the biggest social networking service provider. We create a personal Facebook account for each common agent, as well as the pursuer. With the help of Facebook Software Development Kit (SDK), we are able to let robots interact with Facebook website automatically. For the common agents, once any of them captures an image, it will automatically share the image online under its Facebook account.

For pursuer, social activities are also achieved by Facebook SDK. Due to the restrictions of current SDK, friend requests cannot be sent by program. To model the same scenario, we let pursuer connect to all common agents at the beginning, and simulate the network connectivity variation by controlling which common agents ought to be checked. In other words, if a common agent is not connected, pursuer will not check its images until they are reconnected. The second stage of the searching approach is achieved by using the techniques of SURF detection, which will be presented in the next subsection.

2.3.4.2 Speeded-up robust features detection

Speeded-Up Robust Features, abbreviated as SURF, is a robust feature detector developed by Bay et al. (2008). It utilizes the sum of Haar wavelet response around the points for features and can be computed very efficiently. In our case, we use this technique for searching evader and landmarks. Initially, we store the feature descriptors of evader's marker and all landmarks as references. During the experiment, SURF detection is applied to compare the features of online images and reference images. Once the similarities are found, the program will save the results and inform the pursuer.

2.3.4.3 Path planning

In our experiment, common agents and pursuer are autonomous robots. This requires them to have a path planning algorithm helping them in navigation. On one hand, the algorithm should control robots proceed with their tasks. On the other hand, it should integrate collision avoidance in case of hitting obstacles or other robots.

With this in mind, we develop the path planning strategy in the following way. For common agents, since they are randomly deployed in the environment and do not have specific destinations, we let them move forward in every time instant till obstacle is met. For each of them, we estimate the relative location of the obstacle nearby according to the measurements of infrared sensors, and let common agent turn before collision. Since common agents are used to model real people, this moving strategy satisfies the real situation that people walk around.

For pursuer, it needs to navigate to specific position when evader is located. So we apply

Dijkstra's algorithm for path planning. Dijkstra's algorithm is a graph search algorithm for finding the shortest path on a graph with non-negative edge path costs. This algorithm is widely used in network routing and data structure, etc. To implement the algorithm, we define a visibility graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of all vertices of obstacles after offset, and $\mathcal{E}$ is the set of corresponding edges based on visibility. The cost of each edge is the real distance between corresponding nodes. Once evader is located, we update G by adding two vertices: current position of the pursuer and estimated position of the evader. After constructing G, we are able to compute the shortest path between pursuer and evader. But during the navigation, pursuer still has chance to collide with obstacles. For this reason, four infrared sensors are installed on-board. When pursuer proceeds from one node to another, it will deviate from original path based on the measurements of sensors in order to avoid collision. This also helps when pursuer meets other robots during navigation. The complete process can be found at Algorithm 4. Therefore, by integrating Dijkstra's algorithm and collision avoidance, pursuer is able to reach evader's estimated location safely.

2.3.4.4 Experimental result

In real experiment, the parameters we use in the algorithms are listed in Table 2.3. To model social network changing, we set a friending probability and an unfriending probability for each common agent. The values we use are listed in Table 2.4.

Table 2.3: Experiment parameters

N_{req}	3
T_{wait}	10
N_{img}	1

Table 2.4: Probability parameters

	Friending	Unfriending
Common Agent 1	0.4	0.05
Common Agent 2	0.5	0.02
Common Agent 3	0.3	0.04

Figure 2.33 depicts the experimental scene and Figure 2.34 illustrates the corresponding connectivity changing. We can see that network evolution algorithm is effective in the network development. All the three agents are connected in most of the time, and at least one of them is connected in any time instant. Furthermore, with the help of proposed searching approach, pursuer is able to track evader based on the online information. Path planning algorithms for common agents and pursuer are feasible in real environment. Note that there exists obvious time delay between the discovery of evader and the corresponding response of pursuer. This can be caused by some reasons such as Internet transmission delay and wireless camera transmission delay, etc.

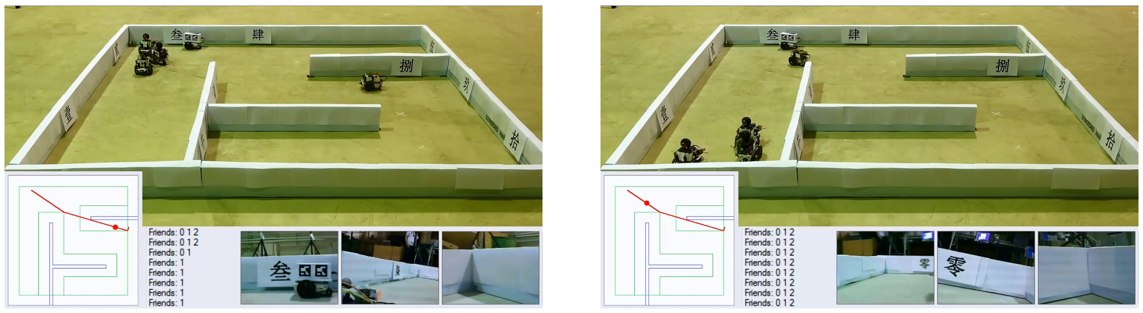


Figure 2.33: Implementation

2.3.5 Conclusion

In this chapter, we described a pursuit-evasion game based on social network. The game is different from those in previous chapters because the pursuer does not have any vision sensor on-board. Instead, it has Internet access for collecting information. We first introduced the notion of common agent. Rather than getting position information from global tracking system, we established a searching approach with the help of social network website. The approach included network evolution algorithm and image scanning algorithm, which were very useful to locate the evader. SURF detection was used to find the evader and landmarks. Since the entire map was regarded as prior knowledge to pursuer, we used a path planning strategy integrating Dijkstra's algorithm and collision avoidance based on the measurements of infrared sensors. Implementation results demonstrated the feasibility of this tracking approach.

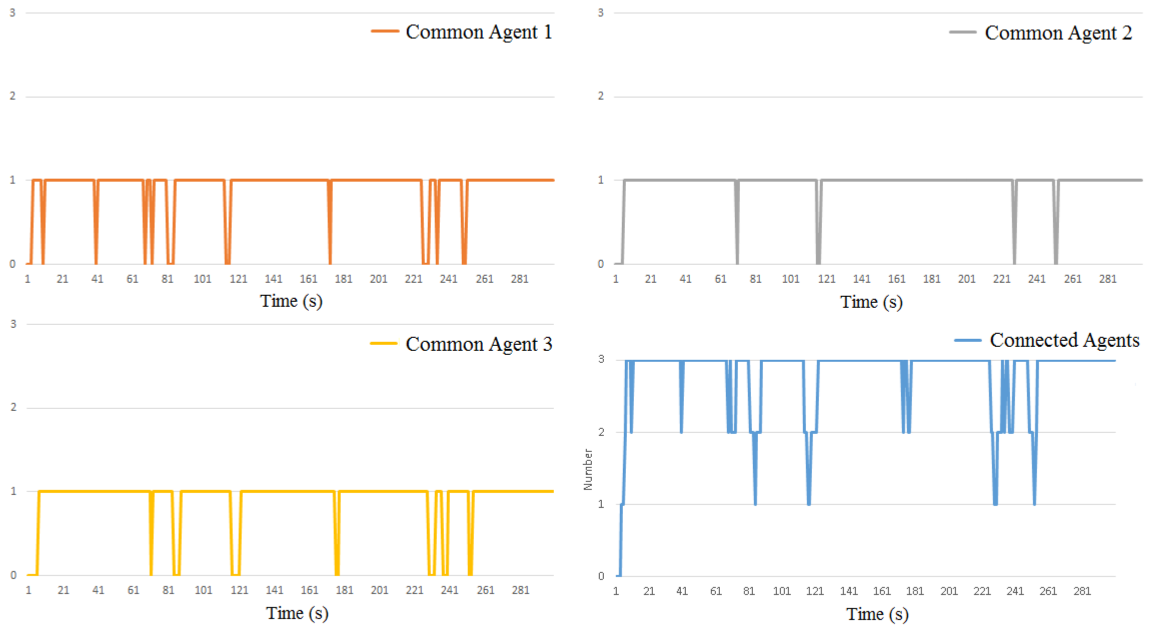


Figure 2.34: Connectivity variation between pursuer and common agents

CHAPTER 3. SCHEDULING AND MOTION PLANNING FOR AGRICULTURAL VEHICLES

3.1 Introduction

Logistics problem can be described as the management of resources in order to meet specific requirements, or customers. The resources to be dealt with include physical items such as materials and tools, as well as abstract items such as information and energy. Recently, logistics problem is embodied in many aspects, such as business, economics and agriculture. We address the problem of logistics in motion planning for agricultural vehicles is addressed.

Currently, crop-harvesting is usually performed by agricultural machines called combines (combine harvesters). Due to the limited capacities of the combines, a grain cart is involved for transporting the grains from combines to the depot. Ali and Van Oudheusden (2009) address the motion planing of one combine by using integer linear programming formulation. Scheuren et al. (2013) present a path planning approach for the unloading vehicle.

If there are sufficient number of grain carts, each combine can go alongside with a grain cart for unloading. However, if the number of combines exceeds the number of grain carts, we have to consider the problem of scheduling the grain carts to unload all the combines. Moreover, most farmers prefer to unload the combine when it is harvesting in order to improve the efficiency. This indeed requires better coordination between the combines and grain cart so that the unloading process can be performed without any interruption. Based on the fact that the grain cart is primarily responsible for unloading grains, we explore the problem of motion planning and scheduling for an autonomous grain cart that serves multiple combines in a field.

There have been some efforts in the past to address the problem of harvesting in large-scale farming scenarios. Fokkens and Puylaert (1981) present a linear programming model for harvesting operations. The model gives the management results of harvesting operations at the large scale grain farm. Foulds and Wilson (2005) and Basnet et al. (2006) analyze the scheduling problem of farm-to-farm harvesting operations for hay, and rape seed. These works mainly focus on scheduling harvesting operations from farm to farm. In contradistinction, our research focuses on the motion planing for both harvesting and unloading vehicles.

Recently, path planning of agricultural machines has received some attention in the research community. Makino et al. (1999) develop a motion planning system, which integrates global and local motion planning components, for agricultural vehicles. Oksanen and Visala (2009) propose algorithms and methods to solve coverage path planning problem. The algorithms not only aim to find an efficient route, but also ensure the coverage of the whole field. Hameed et al. (2013) propose a coverage planning approach with the consideration of the presence of obstacles. However, these works do not consider the presence of multiple harvesting machines, which is a distinguishing feature of our work.

There has been some previous research to plan optimal trajectories for tractor-trailer model which are prevalent in farming applications. Divelbiss and Wen (1994) present an algorithm to find a feasible path which satisfies the given non-holonomic constraints. Divelbiss and Wen (1997) propose a trajectory tracking strategy which controls a tractor-trailer system moving along a path generated off-line. By introducing the notion of equivalent size, Liu et al. (2008) propose an approach for path planning based on genetic algorithm. Yu and Hung (2012) consider the tractor-trailer model as a Dubins vehicle, which can only move with constant speed and turn with upper bounded curvature. The proposed algorithm is used to find the shortest path in Dubins Traveling Salesman Problem with Neighborhoods Isaacs et al. (2011). Based on the tractor-trailer model, Chyba and Sekhavat (1999) introduce the notion of regular primitive and singular primitive which ensure local time optimality. Chitsaz (2013) present an extension on the study on primitives and give a detailed characterization of the proposed curves.

3.2 Mathematical Modeling

3.2.1 Combine harvester

Combine harvester is the machine for harvesting crops, for example, wheat, oats, rye, barley corn, soybeans and flax. Figure 3.1 shows a combine at work. In this active mode, the header cuts the crop and feeds it into threshing cylinder. Grain and chaff are separated from the straw when crop goes through the concaved grates. The grain, after being sieved, will be stored in the on-board tank temporarily, and the waste straw is ejected. We use C to denote the maximum capacity of on-board tank.

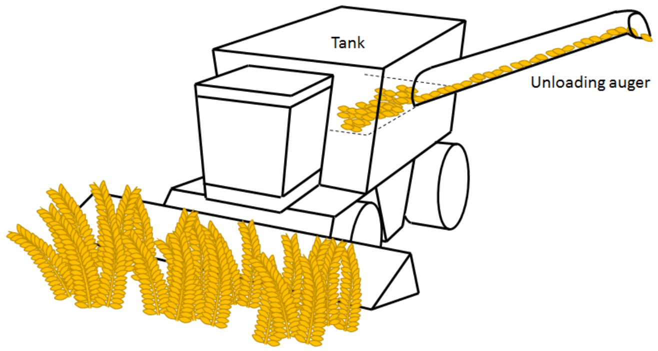


Figure 3.1: Combine harvester

Threshing grain loss is an important issue for combine harvester. For any combine, the quantity of threshing grain loss greatly depends on the forward speed of the harvester. Flufy and Stone (1983) show that automatic control has a better performance than manual control on the threshing grain loss. The forward speed is controlled to give a level of crop feed according to the required threshing grain loss. In this study, we simplify the model and assume that all the combines posses identical constant speed, and the field has a constant density of crop. With these assumptions, the filling rate of the tank, denoted as r_f, can be regarded as a constant.

Since the tank does not hold a large capacity, modern combine usually has an unloading auger for removing the grains from the tank to other vehicles. For most of the combines, the auger is mounted on the left side, as shown in Figure 3.1. At this point, a vehicle has to be on the left side of the combine to empty the tank. Here we denote the unloading rate of the tank using auger as r_u. So when a combine proceeds with the harvesting and the unloading operations simultaneously, the unloading rate is $r_u - r_f$ $(r_u > r_f)$.

In general, for a rectangular field, multiple combines travel to harvest from the perimeter of the field to its center (Zhang (2014)). In other words, when each combine finishes harvesting the current row, it will move to the row along the perimeter of unharvested field. Figure 3.2 shows an example for 3 combines. This guarantees that each combine has an open space on its left side, so that it can be unloaded anytime by a grain cart.

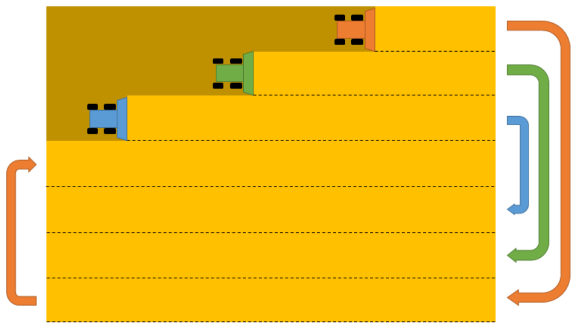


Figure 3.2: Motion planning of 3 combines

3.2.2 Grain cart

A grain cart, also known as chaser bin, is a trailer towed by a tractor. In the thesis, we use term grain cart to represent the system including both the tractor and the trailer. Figure 3.3 (a) shows the appearance of a grain cart. Because of the larger capacity, one can use it to collect grains from multiple combines so that the combines could keep working without interruption.

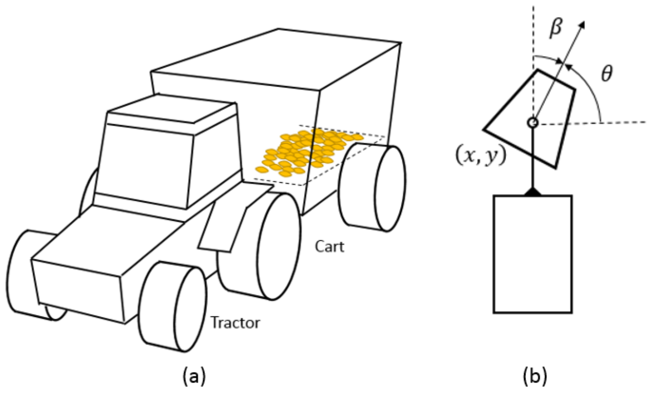


Figure 3.3: Grain cart and its mathematical model

Figure 3.3 (b) shows the model a grain cart. We model the grain cart as a trailer attached to a car-like robot. The robot is hitched by the trailer at the center. The equations of motion for the grain cart are as follows:

$$\dot{q} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\beta} \end{pmatrix} = \begin{pmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ -v\sin\beta + \omega \end{pmatrix}$$

where $q = (x, y, \theta, \beta) \in \mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^1$ is the configuration, $(v, \omega) \in U = [-1, 1]^2$ is the control (Chyba and Sekhavat (1999) and Chitsaz (2013)). In the configuration q, (x, y) is the coordinate of robot's center, θ is the robot's orientation, β is the angle between trailer orientation, and the robot orientation. v and ω denote the speed and angular velocity of the robot, respectively.

3.3 Problem Description

Consider the scenario when N combines harvest the field in the manner described in the previous section, and there is one grain cart that serves all of them. We assume all the combines posses the same capacity C, filling rate r_f and unloading rate r_u. All of them harvest on different rows of the field, and keep a constant distance between the ones on adjacent rows. Besides, for the first $(N-1)$ combines, it takes ΔT for grain cart to move from one combine to the next. Therefore, the travel time between the Nth combine and the first one will be $(N-1)\Delta T$. This models a real scenario since the distance could be a measure of the time for grain cart with a constant speed. In this study, we address the following question: How to schedule the grain cart for sequentially unloading the combines so that all the combines could keep harvesting, and the tanks will not overflow?

3.4 Scheduling Strategy

In this section, we propose a sequential strategy for the grain cart to visit the combines, and unload them. In other words, in the case of N combines, the grain cart will try to serve the first one till the Nth one in order, and then come back to the first one. In order to motivate the proposed strategy for N combines, we first study the case when $N = 2, 3$, and generalize the result to N combines with one grain cart.

3.4.1 2 combines with 1 grain cart

In order to solve the problem for $N = 2$, we first introduce a notion of *waiting time* t. The waiting time starts when grain cart empties the serving combine, and ends when it leaves the same combine. Figure 3.4 illustrates the waiting time at this point. During the waiting time, the grain cart continues to empty the tank of combine which is being served. Based on the scheduling, it can be seen form the figure that the waiting time t is associated with both combines.

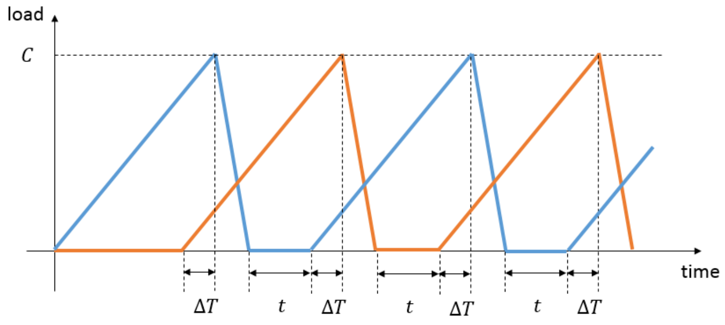


Figure 3.4: Scheduling for 2 combines

With this in mind, we can check if with waiting time $t \geq 0$, the grain cart is able to serve both combines and the scheduling pattern can be repeated. So $t \geq 0$ can be considered as a sufficient condition for the repeatability of this process. From Figure 3.4, we can infer the following:

$$\begin{aligned} t = \frac{C}{r_f} - 2\Delta T - \frac{C}{r_u - r_f} \geq 0 \\ \Rightarrow \Delta T \leq \frac{r_u C - 2 r_f C}{2 r_f (r_u - r_f)} \end{aligned} \tag{3.1}$$

So, we obtain the constraint on ΔT from Equation (3.1) which indicates the solution to the problem. Note that Figure 3.4 can also be interpreted as the scenario that the combines start one after the other. This scenario satisfies real situation that farmers may ignite the combines one after the other and proceed with the harvesting work.

In the previous analysis, we assume that the distance d between two adjacent combines keeps constant, so that ΔT does not change in each iteration. However, if the combines need to work without interruption, d will become smaller based on the models described in Section 3.2. This can be inferred from the fact that two combines have different travel distances between rows. Let us denote the initial horizontal distance between two combines as d_{init} and use the subscription i to denote the iteration ($i \geq 1$), then we have

$$d_i = d_{init} - 2l(i-1)$$

where l is the width of field row. If the distance has a restriction d_{min}. Then we would have

$$d_{init} \geq d_{min} + 2l(I-1)$$

where I is the total number of iterations.

3.4.2 3 combines with 1 grain cart

In the case of 3 combines, we need to consider the situation when two combines require unloading. It is intuitive that the grain cart is supposed to leave the current combine earlier so that there is enough time to unload the other two combines. Figure 3.5 illustrates the load variation among the three combines as a function of the time. Red, green and blue lines represent the loads of three combines. In the beginning, let us assume that the grain cart is engaged with the red combine. Let C' denote the load of the green combine when the grain cart starts to unload its tank. The blue combine will be served when its tank is full.

To facilitate the computation, we extend the notion of waiting time. As shown in Figure 3.5, we use t' to denote the time period for which the grain cart is following the green combine after it finishes unloading its tank for the first time. t' is different from the normal waiting time t_1 or t_2 because it depends on C' instead of C.

Based on the results of the previous subsection, a sufficient condition to ensure that the grain cart is able to serve all three combines is $t' \geq 0$, $t_1 \geq 0$ and $t_1 \geq 0$. From the figure, ΔT is given by the following expression:

$$\frac{C'}{r_f} - \Delta T = t_1 + 2\Delta T + \frac{C}{r_u - r_f} \tag{3.2}$$

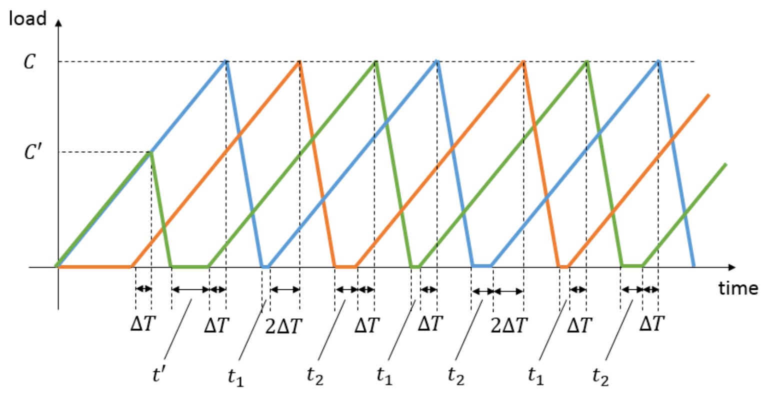


Figure 3.5: Scheduling for 3 combines

Moreover, $t_1 \geq 0$ yields the following:

$$t_1 = \frac{C'}{r_f} - \frac{C}{r_u - r_f} - 3\Delta T \geq 0$$

$$\Delta T \leq \frac{(r_u - r_f)C' - r_f C}{3r_f(r_u - r_f)} \tag{3.3}$$

From the above computations, t_2 is given by the following:

$$\Delta T + \frac{C'}{r_u - r_f} + t' = t_2 + \Delta T + \frac{C}{r_u - r_f}$$

$$t' = t_2 + \frac{C - C'}{r_u - r_f} \tag{3.4}$$

The condition $t_2 \geq 0$ implies that $t' \geq \frac{C-C'}{r_u-r_f}$. From Figure 3.5, we can infer the following:

$$t' = \frac{C}{r_f} - \Delta T = \frac{C'}{r_u - r_f} - \frac{C'}{r_f} \geq \frac{C - C'}{r_u - r_f}$$

$$\Delta T \leq \frac{(r_u - r_f)(C - C') - r_f C}{r_f(r_u - r_f)} \tag{3.5}$$

If we consider the case when $t_1 = t_2 = 0$, we can solve C' and ΔT by substituting inequalities (3.3) and (3.5) by equalities. This leads to the following:

$$\Delta T = \frac{(r_u - r_f)C' - r_f C}{3r_f(r_u - r_f)} = \frac{(r_u - r_f)(C - C') - r_f C}{r_f(r_u - r_f)}$$

$$C' = \frac{(3r_u - 5r_f)C}{4(r_u - r_f)} \tag{3.6}$$

$$\Delta T = \frac{(r_u - 3r_f)C}{4r_f(r_u - r_f)} \tag{3.7}$$

3.4.3 N combines with 1 grain cart

Based on the discussion in the previous subsections, we can generalize the scheduling scheme to N combines with 1 grain cart. Figure 3.6 shows the load variation for this scenario. Since there are N combines, we need to consider C'_1, $C'_2 \cdots C'_{N-2}$, t'_1, $t'_2 \cdots t'_{N-2}$ and t_1, $t_2 \cdots t_{N-1}$. Similar to the analysis of 3 combines, t'_1, $t'_2 \cdots t'_{N-2} \geq 0$ and t_1, $t_2 \cdots t_{N-1} \geq 0$ in order to ensure enough serving time for each combine.

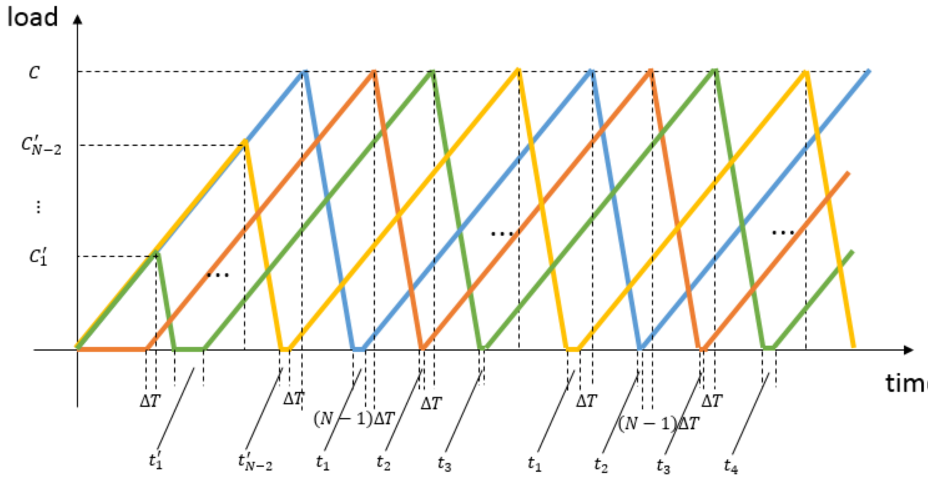


Figure 3.6: Scheduling for N combines

First, we analyze the relationship between t'_n and t_{n+1} when $1 \leq n \leq N-2$. Based on the parallelogram law, the following equation holds:

$$\frac{C'_n}{r_u - r_f} + t'_n + \Delta T = t_{n+1} + \Delta T + \frac{C}{r_u - r_f}$$
$$\Rightarrow t'_n = t_{n+1} + \frac{C - C'_n}{r_u - r_f} \tag{3.8}$$

Then, from the position of each C', it can be seen that

$$t'_n = \begin{cases} \frac{C'_{n+1}}{r_f} - \Delta T - \frac{C'_n}{r_u - r_f} - \frac{C'_n}{r_f} & 1 \leq n \leq N-3 \\ \frac{C}{r_f} - \Delta T - \frac{C'_n}{r_u - r_f} - \frac{C'_n}{r_f} & n = N-2 \end{cases}$$

Since $t_{n+1} \geq 0$ yields $t'_n \geq \frac{C - C'_n}{r_u - r_f}$ $(1 \leq n \leq N-2)$, we can find the corresponding constraints of ΔT.

$$\Delta T \leq \begin{cases} \frac{(C'_{n+1} - C'_n)(r_u - r_f) - r_f C}{r_f(r_u - r_f)} & 1 \leq n \leq N-3 \\ \frac{(C - C'_n)(r_u - r_f) - r_f C}{r_f(r_u - r_f)} & n = N-2 \end{cases}$$

Moreover, according to parallelogram law, the following equation can be obtained, as well as the constraint on ΔT.

$$(t_1 + \cdots + t_{N-1}) = (2N-2)\Delta T + \frac{(N-1)C}{r_u - r_f} - \frac{C}{r_f} \geq 0$$

$$\Delta T \leq \frac{(r_u - r_f)C - (N-1)r_f C}{r_f(r_u - r_f)(2N-2)} \tag{3.9}$$

If we assume $t_n = 0$ $(1 \leq n \leq N-1)$, we obtain the following expression for ΔT based on Equation (3.9).

$$\Delta T = \frac{(r_u - r_f)C - (N-1)r_f C}{r_f(r_u - r_f)(2N-2)} \tag{3.10}$$

Since we also have

$$\Delta T = \begin{cases} \frac{(C'_{n+1} - C'_n)(r_u - r_f) - r_f C}{r_f(r_u - r_f)} & 1 \leq n \leq N-3 \\ \frac{(C - C'_n)(r_u - r_f) - r_f C}{r_f(r_u - r_f)} & n = N-2 \end{cases}$$

we can recursively solve for all values of C'. The solution also satisfies the results obtained in the previous subsections for 2 and 3 combines.

3.5 Motion Planning for Agricultural Vehicles

In this chapter, we consider the time-optimal path planning problem for the grain cart to move between combines on adjacent rows. In other words, we would like to find a path from the initial configuration q_i to the goal configuration q_g with the minimum travel time. A numerical approach and a primitive-based approach are studied to solve the problem. In the following section, we present the numerical approach to compute the time-optimal path between two specific configurations.

3.5.1 Numerical approach

In this section, we present a numerical approach used to obtain the time-optimal solution for the navigation between two given configurations. We first define a value function for the entire configuration space. Based on the definition of value function, we establish Hamilton-Jacobi equation using tractor-trailer model. The trajectory is computed finally using the obtained value function.

3.5.1.1 Hamilton-Jacobi equation

Denote the set of admissible path from the configuration q_i as $\mathcal{A}_{x_i,y_i,\theta_i,\beta_i}$. Given a goal configuration q_g, we define the corresponding value function $u : q \to \mathbb{R}^+ \cup \{0\}$ (Takei et al. (2010)):

$$u(q(T)) = \inf\{T : q(t) \in \mathcal{A}_{x_i,y_i,\theta_i,\beta_i}, q(T) = q_g\} \tag{3.11}$$

The value function can be regarded as the optimal cost-to-go for the tractor-trailer model with given constraints, an initial configuration and a final configuration. By applying dynamic programming principle for the Equation (3.11), we have

$$u(q(t)) = \inf\{u(q(t + \Delta t)) + \Delta t : q(t) \in \mathcal{A}_{x_i,y_i,\theta_i,\beta_i}\} \tag{3.12}$$

Dividing the terms by Δt and taking $\Delta t \to 0$, we are able to derive

$$-1 = \inf\{\nabla u \cdot \dot{q} : |v| = 1, |\omega| \leq 1\} \tag{3.13}$$

With the equations of motion of the grain cart, *Hamilton-Jacobi-Bellman equation* can be obtained as follows

$$-1 = \cos(\theta)\frac{\partial u}{\partial x} + \sin(\theta)\frac{\partial u}{\partial y} - \sin(\beta)\frac{\partial u}{\partial \beta} + \inf_{|\omega|\leq 1}\{\dot{\theta}(\frac{\partial u}{\partial \theta} + \frac{\partial u}{\partial \beta})\} \tag{3.14}$$

The last term in Equation (3.14) can be eliminated by applying bang-bang principle $w = \pm 1$. Since q_i is the goal configuration which has no cost-to-go, we have $u(q_g) = 0$. For the points located in the obstacle or outside the space, we define the cost-to-go to be infinity. In the next section, we present the update scheme of the defined value function.

3.5.1.2 Update scheme

In order to find the time-optimal path satisfying (3.14), we apply fast sweeping method and propose an update scheme for the value function $u(q)$ for the entire space. The basic idea is to discretize the space, as well as the control. For each configuration, the dynamics is moved by a small time step Δt, and the value of this configuration is the smallest value among all the points reachable adding Δt. We compute all the points in the space until value function converges.

For each configuration reachable, if it lies on the grids, we can know the value directly, if it is not, we take the average of its neighbors.

With this idea, we set up a four dimensional uniform Cartesian grid with discritization refinements $(h_x, h_y, h_\theta, h_\beta)$. Let $u_{a,b,c,d} = u(q_{a,b,c,d}) = u(ah_x, bh_y, ch_\theta, dh_\beta)$ be the approximation of the solution on the grid nodes. We discretize ω in the range of $[-1, 1]$ and further define $u^*_{a,b,c,d}$ as follows

$$u^*_{a,b,c,d} = \min_{\omega_i \in [-1,1]} \{u(q_{a,b,c,d} + \dot{q}\Delta t)\} + \Delta t \tag{3.15}$$

where $\dot{q} = (\cos(ch_\theta), \sin(ch_\theta), \omega_i, -\sin(dh_\beta) + \omega_i))^T$, ω_i is the ith element in the discretization and Δt is the length of time step. The value of $u(q_{a,b,c,d} + \dot{q}\Delta t)$ is approximated by taking the average value of the adjacent nodes in the presented grid.

Finally, the update scheme can be described as follows

$$u^{n+1}_{a,b,c,d} = \min\{u^n_{a,b,c,d}, u^{*n}_{a,b,c,d}\} \tag{3.16}$$

where the superscripts denote the iteration. The termination condition of the computation could be described as for any $\epsilon > 0$, the following inequality holds.

$$(\|u^{n+1}_{a,b,c,d} - u^n_{a,b,c,d}\|_2)^2 < \epsilon \tag{3.17}$$

3.5.1.3 Computing trajectory

By using the obtained value function $u(q)$, we are able to derive the time-optimal path from any initial configuration q_i to the goal configuration q_g. According to Equation (3.14), the control law can be summarized as follows

$$\dot{x} = \cos\theta \tag{3.18}$$

$$\dot{y} = \sin\theta \tag{3.19}$$

$$\dot{\theta} = -sgn(\frac{\partial u}{\partial \theta} + \frac{\partial u}{\partial \beta}) \tag{3.20}$$

$$\dot{\beta} = -\sin\beta + \dot{\theta}. \tag{3.21}$$

Note that the partial derivative in Equation (3.20) is obtained by applying centered difference approximation. The values of u which are not on the nodes are computed using a nearest-neighbor interpolation.

The numerical approach computes the time-optimal trajectory efficiently if the corresponding value function is provided. The main time consumption is in computing the value function of the final configuration. But in real implementation, one can compute the value function beforehand. Therefore, the time cost of computing the value function will not influence the real operation on path planning.

3.5.2 Primitive-based approach

3.5.2.1 Related work

Based on the model presented in Section 3.2, the time-optimal trajectory satisfies Pontryagin Maximum Principle. Chitsaz (2013) define adjoint variables $\lambda = (\lambda_x, \lambda_y, \lambda_\theta, \lambda_\beta)$ and the Hamiltonian as follows

$$\begin{aligned} H(q, \lambda, u) &= \lambda_x \dot{x} + \lambda_y \dot{y} + \lambda_\theta \dot{\theta} + \lambda_\beta \dot{\beta} \\ &= v(\lambda_x \cos\theta + \lambda_y \sin\theta - \lambda_\beta \sin\beta) + \omega(\lambda_\theta + \lambda_\beta) \end{aligned} \tag{3.22}$$

Additionally, the switching functions are defined as

$$\phi_v = \lambda_x \cos\theta + \lambda_y \sin\theta - \lambda_\beta \sin\beta \tag{3.23}$$

$$\phi_\omega = \lambda_\theta + \lambda_\beta \tag{3.24}$$

Depending on ϕ_v and ϕ_ω, the optimal trajectory, which is called *extremal*, consists of two categories, namely, *regular* and *singular*. On one hand, an extremal is called regular if the times at which $\phi_v = 0$ or $\phi_\omega = 0$ have zero measure. On the other hand, an extremal is called *abnormal* if it has both $\phi_v \equiv 0$ and $\phi_\omega \equiv 0$. A singular is an extremal which contains a positive measure along which $\phi_v \equiv 0$ or $\phi_\omega \equiv 0$. The subtrajectories of a regular and a singular extremal are called *regular primitive* and *singular primitive*, respectively.

In the regular primitive, $\phi_v \neq 0$ and $\phi_\omega \neq 0$. Based on the state equations, a regular primitive satisfies

$$\theta(t) = \omega t + \theta(t_0) \tag{3.25}$$

$$x(t) = x(t_0) + (v/\omega)(\sin(\theta) - \sin(\theta(t_0))) \tag{3.26}$$

$$y(t) = y(t_0) - (v/\omega)(\cos(\theta) - \cos(\theta(t_0))) \tag{3.27}$$

$$\beta(t) = 2(v/\omega)\arctan(\frac{t - 2v + K_1}{t + K_1}) \tag{3.28}$$

$$K_1 = \frac{2}{v - \omega\tan(\beta(t_0)/2)}$$

where t_0 is the starting time instant and t is the passing time.

In singular primitive, we have either $\phi_v \equiv 0$ or $\phi_\omega \equiv 0$. Here since the grain cart has a constant forward speed, we only consider the case of $\phi_\omega \equiv 0$. It has been proved by Chitsaz (2013) that if a ϕ_ω-singular primitive contains a straight line segment, either the entire primitive is a straight line segment, or

$$\alpha(t) = \pm 2\beta(t) \tag{3.29}$$

$$\omega(t) = d = \pm 2\sin(\beta(t)) \tag{3.30}$$

$$-\frac{\pi}{6} \leq \beta(t) \leq \frac{\pi}{6} \ \ or \ \ \frac{5\pi}{6} \leq \beta(t) \leq \frac{7\pi}{6} \tag{3.31}$$

in which α denotes the angle between the robot orientation and the line, d denotes the distance between robot's center and the line. The path for the latter case is called a *merging curve.*

3.5.2.2 Path planning of grain cart

In our case, when the grain cart finishes unloading one combine, it should follow a path to the second combine, and move parallel to it. This implies that the final path is supposed to end with a singular primitive, denoted as S_g. Considering the fact that a merging curve has the constraints (3.30) and (3.31), we propose a combination of the path containing 2 regular primitives and 2 ϕ_ω-singular primitives, as shown in Figure 3.7. The grain cart initially proceeds with a regular primitive R_1. S_1 is a singular primitive connecting to R_1. R_2 and S_g are the following regular primitive and goal singular primitive, respectively.

In order to minimize the travel time for the grain cart, we consider using straight lines instead of merging curves for ϕ_ω-singular primitives. Furthermore, since q_i and q_g has the same

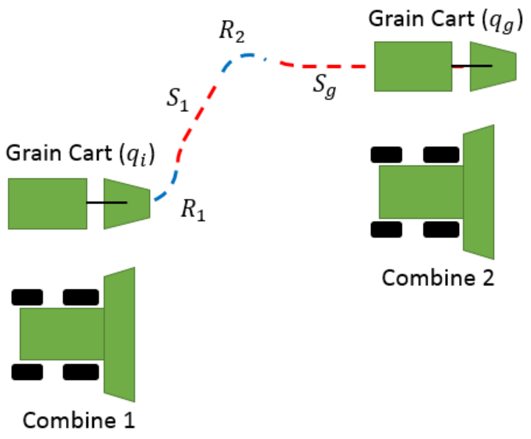


Figure 3.7: The path consists of 2 singular primitives (red lines) and 2 regular primitives (blue lines)

θ, it is obvious that R_1 and R_2 should have the same length, and the same central angle γ as well. Based on this fact, we can find that with a given γ, the slope of S_1 can be computed. Since q_i and q_g are known, the entire path can be derived. To minimize the travel time, we change the central angle corresponding to the regular primitives, and take the minimum time trajectory in all feasible trajectories as the final path. The complete algorithm for computing the trajectory can be found in Algorithm 5.

3.6 Simulation

In this section, we present the simulation results. The first simulation is carried out with a 400 units $\times$ 275 units field. The land is divided into 11 rows. Two combines, as shown in Figure 3.8 are harvesting on the path presented in Subsection 3.2.1. The simulation shows that during the harvesting operation, the distance between combines keep decreasing. Also, all combines keep an open space at their left side so that the unloading activities can be proceeded safely.

Next, we present the simulation results for the proposed motion planning approaches. Figure 3.9 illustrates the paths obtained using the numerical approach and primitive-based approach. Note that due to the fact that the tractor-trailer model has four state variables, it is hard to visualize the variations of all the variables. For this reason, in the simulation we

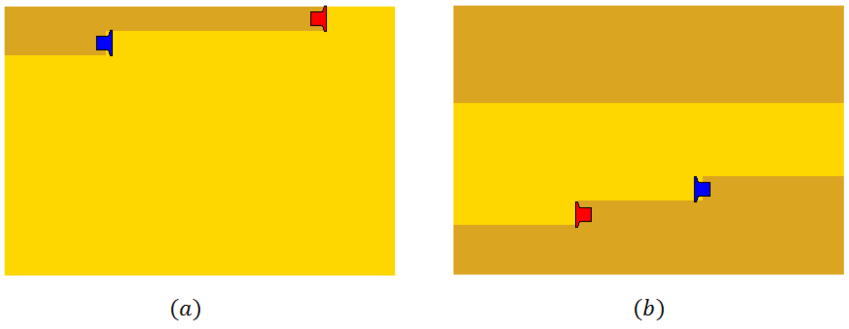


Figure 3.8: Paths of two combines

only show the path of (x, y), which represents the physical position of grain cart. In the simulation, the initial and goal configuration are set to be $q_i = (1, 1, 0, 0)^T$ and $q_g = (4, 4, 0, 0)^T$, respectively. For numerical approach, Table 3.1 lists all the refinement parameters in the computation of value function. Angular velocity ω is discretized into 50 uniform nodes. In the final trajectory, bang-bang control is used by letting $\omega = -sgn(\frac{\partial u}{\partial \theta} + \frac{\partial u}{\partial \beta})$. In both approaches, the path computation terminates when the state of the grain cart reaches a range of the final configuration.

Table 3.1: Refinement parameters

h_x	0.2
h_y	0.2
h_θ	0.251 (radium)
h_β	0.251 (radium)
Δt	0.05
ϵ	0.005

Simulation results show that both paths finally reach the goal configuration which validates the proposed approaches. Note that in the numerical approach, the path is affected by the error due to the large refinement parameters. With smaller refinement parameters, the path could be more accurate, but leads to higher time consumption on the computation of value function.

In the next simulation, we compare the travel time of using two proposed approaches. We

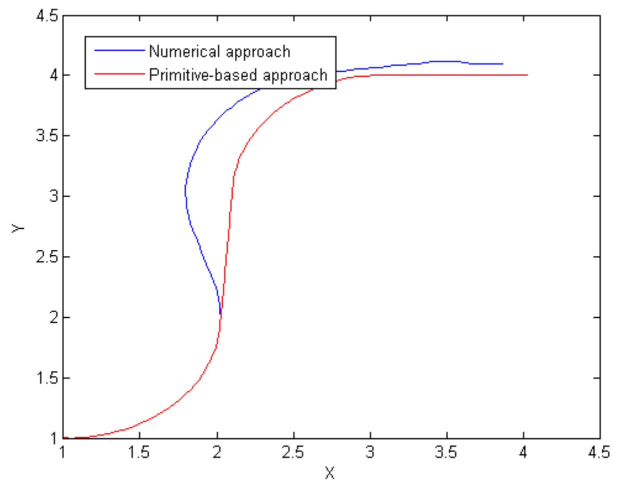


Figure 3.9: Path of (x, y) with given initial and final configurations

keep $y_i = y_g - 4$ and plot the travel time corresponding to the two approaches with respect to the ratio which is $\frac{x_g - x_i}{y_g - y_i}$ (Figure 3.10).

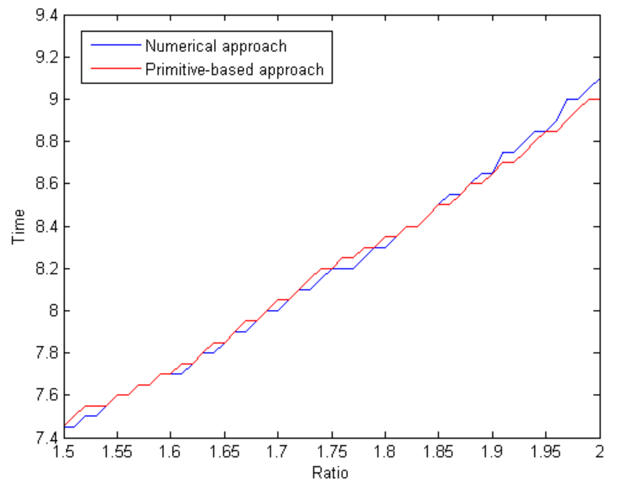


Figure 3.10: Travel time comparison of using two approaches

The results show that with a given ratio, the travel time of both approaches are very close. Since the numerical approach provides us the time-optimal solution, we can know that the primitive-based approach also has a good performance. Note that in some cases, the performance of primitive-based approach is better due to the error in the numerical approach.

3.7 Conclusion

In this chapter, we considered the scheduling and motion planning problem for N combines with one grain cart. We first gave the mathematical models for both combine and grain cart. Based on the presented models, we analyzed the scheduling problem with different number of combines and finally provided the solution for a general case. In the second part, we presented two approaches to find the time-optimal solution to the path planning problem. Firstly, a numerical solution was carried out by applying bang-bang control based on a generated value function. In the primitive-based approach, regular and singular primitives were introduced to construct the final path. Both primitives guaranteed the local optimality. Simulation results for both approaches were provided to demonstrate the feasibility.

CHAPTER 4. RESEARCH SUMMARY

In this chapter, we present a brief summary for all the problems we have addressed. Some directions of future work are also included.

4.1 Visibility-based Target Tracking Game

In this study, the problem of visibility-based target tracking for a team of mobile observers trying to track a team of mobile targets was addressed. Based on the results of previous work, the notion of pursuit fields around a single corner was introduced. We used the pursuit fields to generate navigation strategies for a single observer to track a single target in general environments. In order to tackle the case when more than one observer or target was present in the environment, we proposed a two level hierarchical approach. At the upper level, the team of observers used a ranking and aggregation technique for allocating each target to an observer. At the lower level, each observer computed its navigation strategy based on the results of the single observer-single target problem, thereby, decomposing a large multi-agent problem into several 2-agent problems. Finally, we presented a scalable algorithm that can accommodate an arbitrary number of observers and targets. An empirical evidence of the efficacy of our algorithm was presented based on simulation and implementation results.

Furthermore, a setup of omni-directional camera was designed for the pursuers to get visual information of the surroundings. Based on the aforementioned setup, we applied a position estimation technique for locating the evader. The technique contained an internal parameter which needed to be calibrated based on experimental results. According to the performance of the technique, we found that the error has remarkable effect when the measuring distance was very large. For this reason, we further presented a modified tracking strategy which kept the evader in an effective observing range for the position estimation algorithm. At last, we presented PID controllers for the pursuer to track the evader in a simplified case using the data of the error in position estimation technique. Performances of the controllers were given by simulations.

4.2 Social Network Target Tracking Game

In order to address the situation that pursuer does not have an on-board vision sensor, we considered performing a novel tracking strategy based on the available information on social network. The strategy relied on using the images posted by common agents. To achieve this, we further proposed a network evolution algorithm and an image scanning algorithm. In the proposed approach, SURF feature tracking was used to recognize evader and landmarks. Dijkstra's algorithm was applied for pursuer path planning. With the help of the proposed strategy, pursuer does not require to have vision sensor on-board, and the evader does not require to be in pursuer's field of view initially.

4.3 Logistics Planning for Agricultural Vehicles

In this study, we addressed the scheduling and motion planning problem for an autonomous grain cart serving multiple combines. By giving the mathematical models of both combine harvester and grain cart, we proposed a scheduling scheme which allows grain cart to unload all the combines without interruption in the harvesting activity. The cases with different number of combines were discussed. The scheme was finally generalized to an arbitrary number of combines. Furthermore, we presented path planning analysis for the grain cart based on tractor-trailer model. A numerical approach and a primitive-based approach were considered to obtain the time-optimal solution for path planning. In the former approach, a value function corresponding to the goal position was computed beforehand. Based on the value function, a time-optimal path can be obtained accordingly. In the latter approach, path was composed of singular primitives and regular primitives which guarantee local time-optimality. Finally, simulation results were provided to validate the feasibility of the proposed techniques. In the next section, we present our future work corresponding to these projects.

4.4 Future Work

Visibility-based target tracking can be extended in several directions. One direction is to consider the situation that pursuers and evaders do not have a complete communication graph. In this decentralized case, each pursuer does not have the full information of evaders so that evader assignment will be an interesting problem. Another direction is to consider different risk vectors for the proposed strategy. Comparisons of these risk vectors may improve the technique. Furthermore, other methods can be used for ranking and aggregation instead of Borda Count. This may help pursuers make a better decision on the evader assignment.

For the social network tracking problem, future work involves considering the effect of mutual friends when sending friend requests. People apt to accept requests from those who have more mutual friends with them. Based on this fact, finding the critical persons to send friend requests may help improve the network evolution algorithm. Moreover, based on real data of people's activities on social network, a better model may help improve the proposed algorithms.

In the logistics problem of agricultural vehicles, one of the future directions is to account for the backing up problem of the tractor-trailer model. Besides, studying the scenario when more than one grain cart is deployed for multiple combines will be another direction. In this case, cooperations among grain carts can be considered to improve the harvesting efficiency.

APPENDIX A. TARGET TRACKING ALGORITHMS

Algorithm 1 Ranking and aggregation algorithm

```
1:  call BordaCountMethod($r_{ij}$) and return $score \in \mathbb{R}^{N_e}$ and $rank \in \mathbb{R}^{N_e}$
2:  declare $S \in \mathbb{R}^{N_p \times N_e}$
3:  for $i = 1 \to N_p$ do
4:    for $j = 1 \to N_e$ do
5:      if evader $j$ is visible to pursuer $i$ then
6:        $S_{ij} = score_j$
7:      else
8:        $S_{ij} = \infty$
9:      end if
10:   end for
11: end for
12: Apply Hungarian algorithm to $S$ and assign pursuers accordingly
13: if $N_p > N_e$ then
14:   for $i = 1 \to$ number of extra pursuers do
15:     for $j = 1 \to N_e$ do
16:       if evader $rank_j$ is visible to pursuer $i$ then
17:         Assign pursuer $i$ to evader $rank_j$
18:       end if
19:     end for
20:   end for
21: end if
22: function BordaCountMethod($r_{ij}$)
23: declare $score \in \mathbb{R}^{N_e}$ and $rank \in \mathbb{R}^{N_e}$
24: for $i = 1 \to N_p$ do
25:   Rank the $i$th row of $r_{ij}$ from high to low
26:   for $j = 1 \to N_e$ do
27:     $score_k = score_k + (N_e - j)$ where $k$ is the original column index of $r_{ij}$
28:   end for
29: end for
30: Rank $score$ from high to low and save each element's original index to $rank$
31: return $score$ and $rank$
```

Algorithm 2 Network evolution algorithm

```
1: declare common agent set $A = \phi$
2: declare friend agent set $F = \phi$
3: declare counter set $C = \phi$
4: for $i = 1 \rightarrow N_{req} - length(A)$ do
5:    if $length(F) + length(A) \leq N_f$ then
6:       Send friend request to new common agent $a$
7:       Add $a$ to $A$
8:       Add counter $c_a = 0$ to $C$
9:    end if
10: end for
11: for $i = 1 \rightarrow length(A)$ do
12:    Check whether common agent $A_i$ accepts the request
13:    if Common agent $A_i$ accepts then
14:       Add $A_i$ to $F$
15:       Delete $A_i$ from $A$
16:       Delete $c_{A_i}$ from $C$
17:    else if Common agent $A_i$ rejects then
18:       $c_{A_i} = c_{A_i} + 1$
19:       Resend friend request to $A_i$
20:    else
21:       $c_{A_i} = c_{A_i} + 1$
22:       if Counter $c_{A_i} > T_{wait}$ then
23:          Delete $A_i$ from $A$
24:          Delete $c_{A_i}$ from $C$
25:       end if
26:    end if
27: end for
```

Algorithm 3 Image scanning algorithm

```
1: declare $F$ to be the set of friend agent
2: for $i = 1 \rightarrow length(F)$ do
3:    for $j = 1 \rightarrow N_{img}$ do
4:       Check the $j$th recent image of common agent $F_i$
5:    end for
6: end for
```

Algorithm 4 Pursuer path planning algorithm

```
1: declare visibility graph $G(\mathcal{V}, \mathcal{E})$
2: if Evader is located then
3:     Add current position of pursuer to $\mathcal{V}$
4:     Add estimated position of evader to $\mathcal{V}$
5:     Update set $\mathcal{E}$ and construct cost matrix
6:     Apply Dijkstra's algorithm to $G$
7:     Navigate pursuer
8: end if
```

APPENDIX B. PATH PLANNING ALGORITHMS

Algorithm 5 Trajectory computation using primitive-based approach

```
1: declare $\gamma$ to be the central angle of regular primitives
2: declare $T_f = \infty$ to be the travel time of final path
3: declare $P_f$ to be the final path
4: for $\gamma = 0 \to \pi$ do
5:    Compute the path $P_\gamma$ starting from $q_i$
6:    if $P_\gamma$ reaches $q_g$ then
7:       if Travel time of $P_\gamma < T_f$ then
8:          $T_f$ = Travel time of $P_\gamma$
9:          $P_f = P_\gamma$
10:      end if
11:   end if
12: end for
```

BIBLIOGRAPHY

Ahmad, A., Tipaldi, G., Lima, P., and Burgard, W. (2013). Cooperative robot localization and target tracking based on least squares minimization. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5696–5701.

Al-Bluwi, I. and Elnagar, A. (2010a). Maintaining visibility of a moving target: Maximizing escape time vs. exposure time. In *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pages 982–987.

Al-Bluwi, I. and Elnagar, A. (2010b). Pursuit evasion in dynamic environments with visibility constraints. In Liu, H., Ding, H., Xiong, Z., and Zhu, X., editors, *Intelligent Robotics and Applications*, volume 6425 of *Lecture Notes in Computer Science*, pages 116–129. Springer Berlin Heidelberg.

Ali, O. and Van Oudheusden, D. (2009). Logistics planning for agricultural vehicles. In *Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference on*, pages 311–314.

Anderson, R. and Milutinovic, D. (2011). A stochastic approach to dubins feedback control for target tracking. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3917–3922.

Bandyopadhyay, T., Hsu, D., and Ang, MarceloH., J. (2009). Motion strategies for people tracking in cluttered and dynamic environments. In Khatib, O., Kumar, V., and Pappas, G., editors, *Experimental Robotics*, volume 54 of *Springer Tracts in Advanced Robotics*, pages 463–472. Springer Berlin Heidelberg.

Bandyopadhyay, T., Li, Y., Ang, M.H., J., and Hsu, D. (2006). A greedy strategy for tracking a locally predictable target among obstacles. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2342–2347.

Bandyopadhyay, T., Li, Y., Ang, M. H., Jr., and Hsu, D. (2004). Stealth tracking of an unpredictable target among obstacles.

Basnet, C. B., Foulds, L. R., and Wilson, J. M. (2006). Scheduling contractors' farm-to-farm crop harvesting operations. *International Transactions in Operational Research*, 13(1):1–15.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359.

Bhattacharya, S., Candido, S., and Hutchinson, S. (2007). Motion strategies for surveillance. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA.

Bhattacharya, S. and Hutchinson, S. (2008). Approximation schemes for two-player pursuit evasion games with visibility constraints. In *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland.

Bhattacharya, S. and Hutchinson, S. (2011). A cell decomposition approach to visibility-based pursuit evasion among obstacles. *The International Journal of Robotics Research*, 30(14):1709–1727.

Chitsaz, H. (2013). On time-optimal trajectories for a car-like robot with one trailer. *CoRR*, pages –1–1.

Chung, T., Hollinger, G., and Isler, V. (2011). Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31(4):299–316.

Chyba, M. and Sekhavat, S. (1999). Time optimal paths for a mobile robot with one trailer. In *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 3, pages 1669–1674 vol.3.

Derenick, J., Spletzer, J., and Hsieh, A. (2009). An optimal approach to collaborative target tracking with performance guarantees. *Journal of Intelligent and Robotic Systems*, 56(1-2):47–67.

Divelbiss, A. and Wen, J. (1994). Nonholonomic path planning with inequality constraints. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 52–57 vol.1.

Divelbiss, A. and Wen, J. (1997). Trajectory tracking control of a car-trailer system. *Control Systems Technology, IEEE Transactions on*, 5(3):269–278.

Emerson, P. (2013). The original borda count and partial voting. *Social Choice and Welfare*, 40(2):353–358.

Flufy, M. L. and Stone, G. (1983). Speed control of a combine harvester to maintain a specific level of measured threshing grain loss. *Journal of Agricultural Engineering Research*, 28(6):537 – 543.

Fokkens, B. and Puylaert, M. (1981). A linear programming model for daily harvesting operations at the large-scale grain farm of the ijsselmeerpolders development authority. *The Journal of the Operational Research Society*, 32(7):pp. 535–547.

Foulds, L. and Wilson, J. (2005). Scheduling operations for the harvesting of renewable resources. *Journal of Food Engineering*, 70(3):281 – 292. Operational Research and Food Logistics.

Frew, E. W. (2009). Combining area patrol, perimeter surveillance, and target tracking using ordered upwind methods. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pages 3123–3128, Kobe, Japan.

Frew, E. W. and Elston, J. (2008). Target assignment for integrated search and tracking by active robot networks. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 2354–9, Pasadena, CA.

Gerkey, B. P., Thrun, S., and Gordon, G. (2006). Visibility-based pursuit-evasion with limited field of view. *Int. J. Rob. Res.*, 25(4):299–315.

Gonzalez-Banos, H., Lee, C.-Y., and Latombe, J.-C. (2002). Real-time combinatorial tracking of a target moving unpredictably among obstacles. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, pages 1683–1690 vol.2.

Guibas, L., Latombe, J.-C., Lavalle, S., Lin, D., and Motwani, R. (1997). Visibility-based pursuit-evasion in a polygonal environment. In Dehne, F., Rau-Chaplin, A., Sack, J.-R., and Tamassia, R., editors, *Algorithms and Data Structures*, volume 1272 of *Lecture Notes in Computer Science*, pages 17–30. Springer Berlin Heidelberg.

Hameed, I., Bochtis, D., and Srensen, C. (2013). An optimized field coverage planning approach for navigation of agricultural robots in fields involving obstacle areas. *International Journal of Advanced Robotic Systems*, 10(231):1–9.

Hollinger, G., Singh, S., Djugash, J., and Kehagias, A. (2009). Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, 28(2):201–219.

Hollinger, G. A., Djugash, J., and Singh, S. (2012). Target tracking without line of sight using range from radio. *Autonomous Robots*, 32(1):1–14.

Hu, W., Tan, T., Wang, L., and Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352.

Isaacs, J., Klein, D., and Hespanha, J. (2011). Algorithms for the traveling salesman problem with neighborhoods involving a dubins vehicle. In *American Control Conference (ACC), 2011*, pages 1704–1709.

Isaacs, R. (1965). *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization.* Dover Publications.

Jung, B. and Sukhatme, G. (2002). Tracking targets using multiple robots: The effect of environment occlusion. *Autonomous Robots*, 13(3):191–205.

Jung, B. and Sukhatme, G. (2006). Cooperative multi-robot target tracking. In Gini, M. and Voyles, R., editors, *Distributed Autonomous Robotic Systems 7*, pages 81–90. Springer Japan.

Jung, B. and Sukhatme, G. (2010). Real-time motion tracking from a mobile robot. *International Journal of Social Robotics*, 2(1):63–78.

Kolling, A. and Carpin, S. (2007). Cooperative observation of multiple moving targets: an algorithm and its formalization. *The International Journal of Robotics Research*, 26(9):935–953.

Kolling, A. and Carpin, S. (2010). Pursuit-evasion on trees by robot teams. *Robotics, IEEE Transactions on*, 26(1):32–47.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97.

LaValle, S., Gonzalez-Banos, H., Becker, C., and Latombe, J.-C. (1997a). Motion strategies for maintaining visibility of a moving target. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 1, pages 731–736 vol.1.

LaValle, S., Lin, D., Guibas, L., Latombe, J.-C., and Motwani, R. (1997b). Finding an unpredictable target in a workspace with obstacles. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 1, pages 737–742 vol.1.

Lee, C.-Y., Gonzalez-Banos, H., and Latombe, J.-C. (2002). Real-time tracking of an unpredictable target amidst unknown obstacles. In *Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on*, volume 2, pages 596–601 vol.2.

Lee, D., Kim, G., Kim, D., Myung, H., and Choi, H.-T. (2012). Vision-based object detection and tracking for autonomous navigation of underwater robots. *Ocean Engineering*, 48(0):59 – 68.

Lee, G., Chong, N., and Christensen, H. (2010). Tracking multiple moving targets with swarms of mobile robots. *Intelligent Service Robotics*, 3(2):61–72.

Li, T.-H. S., Chang, S.-J., and Tong, W. (2004). Fuzzy target tracking control of autonomous mobile robots by using infrared sensors. *Fuzzy Systems, IEEE Transactions on*, 12(4):491–501.

Liu, Z., Lu, Q., Yang, P., and Chen, L. (2008). Path planning for tractor-trailer mobile robot system based on equivalent size. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 5744–5749.

Makino, T., Yokoi, H., and Kakazu, Y. (1999). Development of a motion planning system for an agricultural mobile robot. In *SICE Annual, 1999. 38th Annual Conference Proceedings of the*, pages 959–962.

Murrieta-Cid, R., Monroy, R., Hutchinson, S., and Laumond, J.-P. (2008). A complexity result for the pursuit-evasion game of maintaining visibility of a moving evader. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2657–2664.

Oksanen, T. and Visala, A. (2009). Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8):651–668.

Panagou, D. and Kumar, V. (2012). Maintaining visibility for leader-follower formations in obstacle environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1811–1816.

Panagou, D. and Kumar, V. (2014). Cooperative visibility maintenance for leader-follower formations in obstacle environments. *Robotics, IEEE Transactions on*, 30(4):831–844.

Papanikolopoulos, N., Khosla, P., and Kanade, T. (1993). Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision. *Robotics and Automation, IEEE Transactions on*, 9(1):14–35.

Parker, L. (2002). Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12(3):231–255.

Sachs, S., Rajko, S., and Lavalle, S. M. (2004). Visibility-based pursuit-evasion in an unknown planar environment. *International Journal of Robotics Research*, 23:3–26.

Scaramuzza, D. (2008). *Omnidirectional Vision: from Callibration to Robot Motion Estimation*. PhD thesis, Universit di Perugia.

Scheuren, S., Hertzberg, J., Stiene, S., and Hartanto, R. (2013). Infield path planning for autonomous unloading vehicles. In *GIL Jahrestagung'13*, pages 299–302.

Schulz, D., Burgard, W., Fox, D., and Cremers, A. (2001). Tracking multiple moving objects with a mobile robot. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–371–I–377 vol.1.

Suzuki, I. and Yamashita, M. (1992). Searching for a mobile intruder in a polygonal region. *SIAM J. Comput.*, 21(5):863–888.

Takei, R., Tsai, R., Shen, H., and Landa, Y. (2010). A practical path-planning algorithm for a simple car: a hamilton-jacobi approach. In *American Control Conference (ACC), 2010*, pages 6175–6180.

Tovar, B. and LaValle, S. M. (2008). Visibility-based pursuitevasion with bounded speed. *The International Journal of Robotics Research*, 27(11-12):1350–1360.

Tsalatsanis, A., Valavanis, K., and Yalcin, A. (2007). Vision based target tracking and collision avoidance for mobile robots. *Journal of Intelligent and Robotic Systems*, 48(2):285–304.

Wu, W. and Zhang, F. (2013). A switching strategy for target tracking by mobile sensing agents. *Journal of Communications*, 8(1):47–54.

Xu, Z., Fitch, R., and Sukkarieh, S. (2013). Decentralised coordination of mobile robots for target tracking with learnt utility models. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2014–2020.

Yu, X. and Hung, J. (2012). Optimal path planning for an autonomous robot-trailer system. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pages 2762–2767.

Zhang, Z. (2014). *Development of a Robot Combine Harvester Based on GNSS.* PhD thesis, Hokkaido University.