TAR: TRAJECTORY ADAPTATION FOR RECOGNITION OF ROBOT TASKS TO IMPROVE TEAMWORK

A Dissertation
Presented to
The Academic Faculty

by

Michael Novitzky

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Robotics

School of Interactive Computing
Georgia Institute of Technology
December 2015

TAR: TRAJECTORY ADAPTATION FOR RECOGNITION OF ROBOT TASKS TO IMPROVE TEAMWORK

To those,

that came before me,

and those that have and continue to support me now:

my family, friends, and co-workers, and especially to the naysayers.

ACKNOWLEDGEMENTS

I truly believe that my success is a reflection of team support. In this case, my team is an extended group of people that have helped guide me through various points in my career at Georgia Tech. It all starts with my committee. First, I thank my advisor Tucker R. Balch for allowing me to pursue ideas outside of the box throughout my journey. As a former fighter pilot for the US Air Force, he generally had a refreshing take on the academic community. He always kept an even tone when speaking. Only once did I hear him speak gruffly, and it was in my direction: I will forever remember to catch the humanoid robot before it falls over and its head pops off - resulting in a crowd of gasping children. He often emphasized the bigger picture and guided my development throughout the bumps in the road. Henrik I. Christensen always impresses me by being such a rockstar in robotics, constantly jetting around the world. Yet, he still made time to meet in his busy schedule. I am grateful to have gotten to know him better through his love of Porsches while we drove Georgia Tech's autonomous car Sting on the Atlanta Motor Speedway, leading the procession of Porsches for the Porsche Mille 2011 parade. He was supportive of us RoboGrads by providing us endless coffee and tea, funds for team shirts for our RoboGrads FC, and RIM seminars complete with lunch. He was a great source of publications and ideas. Thomas R. Collins has been patient and supportive throughout my journey. I admire his calm and humble demeanor. A model citizen, he is regularly volunteering with children or animals. We enjoyed giving tours to children in hopes of inspiring them to be roboticist. I am grateful for the countless time he took out of his schedule so that I could bounce ideas off of him and all his help in polishing the various ways I presented my research. I thank Magnus Egerstedt for encouraging me to step outside

of my comfort zone to learn controls. I appreciate that he immortalized me on video: in every class I took with him, at least once a semester he would ask, “Does the computer scientist get it? Misha can we move on?” I valued his guidance and honest feedback. He was a pleasure to have on our RoboGrads FC team. James M. Rehg was one of the most practical people I have interacted with at Georgia Tech. While taking his probabilistic graphical models (PGM) class I asked whether he considers himself a “Bayesian”. His response was “I do what works, I guess you could say that I am a pragmatist.” He has an uncanny ability to be not only paying attention but have such deep insightful comments when you thought he wasn’t paying attention. Playing defense with him on our RoboGrads FC team is by far a highlight of my career at Georgia Tech. Lora G. Weiss always impressed me with her ability to critically analyze situations and her poise during presentations. I appreciated our office visits in which she took apart my work and helped me put it back together into a more coherent whole.

I am grateful to the RoboJackets that would both inspire me and show me the ropes, especially Andy Bardagjy, Phillip Marks, and Jean-Pierre (JP) de la Croix. Although we studied a lot, their late night antics in the library and trips to WingNuts were really great and necessary diversions. The three of them convinced me that if I were ever in need of undergraduates for a robotics project that I should hire from RoboJackets!

I thank my first contact at Georgia Tech, Frank Dellaert for encouraging me throughout my career and for introducing me to linear algebra, the Mac, and the BORG lab. The BORG lab at the time consisted of these great individuals: Dan Hou, Sangmin Oh, Richard Roberts, Michael Kaess, Kai Ni, Grant Schindler, Ananth Ranganathan, Adam Feldman, Keith O’Hara, Matt Powers, Jinhan Lee, Daniel B. Walker, José María Cañas Plaza, and Pablo F. Alcantarilla.

Manuela Veloso was the first person to show me competitive robotics. I will always

remember her questioning me: “Where is the science Misha?” I thank her for writing me a recommendation letter based on our RoboCup experience. I thank the whole RoboCup CMU/GT 2008 team for an amazing experience: Michael Phillips, Brian Coltin, Gaurav Gupta, Dominic Larkin, Pushkar Kolhe, Sonia Chernova, Douglas Vail, and Cheol-Woo Jung.

I thank Mike Stilman for teaching with passion and excitement. His course Robot Intelligence: Planning in Action is among my favorites while at Georgia Tech. I thank him for writing a recommendation letter for me to enter the PhD program. He inspired many of my cohorts to pursue robotics. His early passing is a great loss to the robotics community. I can only hope to inspire others around me as Mike Stilman once did. I thank many in the Golems Lab for late night writing sessions, notably: Jim Perkins, Jiuguang Wang (JW), Neil Dantam, Jon Scholz, and Tobias Kunz.

I thank Michael (Mick) E. West for hiring me at the Georgia Tech Research Institute (GTRI) and introducing me to marine robotics with the Yellowfin project. I got to work with a great team on Yellowfin including: John R. Bogle, Andrew Melim, Evan Seguin, Tedd C. Toler, Matthew P. Bradley, Ron Prado, Richard Reece and Andrew M. Henshaw. I am grateful for my experiences at GTRI which went above and beyond what the typical graduate research assistant ever gets to do. I got to experience meeting a wide variety of customers and had opportunities to participate in a wide variety of projects and experiments in awesome locations. I want to thank Charles Pippin for being such a fantastic project lead in the CUSTD project with a great team including: Gary Gray, Michael Matthews, Dave Price, Ai-Ping Hu, Ben Johnson, and Warren Lee.

I thank Michael R. Benjamin for being such a fantastic proponent for me. His support of MOOS-IvP led to much faster implementation of autonomy and data gathering for my experiments. Under his tutelage, my summer internship at the Battelle Marine Autonomy Lab at MIT was phenomenal. I got to work and interact with great people such as Alon Yaari, Kyle Woerner, and Arthur Anderson, just to name a few. The dock staff at the MIT Sailing Pavilion run a fantastic program and I have to give credit to Stewart Craig and Gerard Welch for making me feel welcome and always pointing in the right direction. I thank Eric Gibber and the volunteers there for providing a supportive learn to sail atmosphere. Sailing proved to be a relaxing and rewarding diversion.

The robotics graduate student group RoboGrads deserves a great amount of praise for providing an inclusive environment at Georgia Tech. As Outreach Chair, it was a pleasure to work with fellow robotics graduate students starting with our first president Jake Huckaby. The organization provided social, educational, and professional events that formed the glue of our community. I thank Tucker Hermans for founding CPL Drinks! which allowed for a weekly social gathering. I want to thank Carlos Nieto-Granda for brainstorming and helping me establish the RoboGrads football club (FC)! While I thought it would just be a great outlet I did not expect to see that level of athleticism and competitiveness out of my fellow RoboGrads!

My fellow academicians and friends deserve credit. Arya Irani for helping me get on my feet during my first semester. Maya Cakmak for being such a patient TA during my Introduction to Artificial Intelligence class. I cherish my friendship with her and her husband Raffay Hamid. Antonio Henrique for being a great representative of Brazil and making my mother happy by watching “Murder She Wrote.” Pushkar Kolhe for being a great friend while we travelled a similar path. Jon Scholz for helping me try to start traditions, even though I think we were the only ones to follow them. Paul Varnell, Kevin DeMarco, Paul Robinette, Andrew Melim, and Andrew Price for being a great cohort that encouraged each other in academic pursuits while working together at GTRI and beyond. John Rogers for his perspective of robotics from experience at high ranking institutions. I thank Alex Trevor for being a fantastic TA during 3D Reconstruction and a great source of sound advice. Crystal Chao for being a rockstar - literally have you heard her play the piano? Nick DePalma for great conversations on the state of robotics and the bigger questions. Tucker Hermans always impressed me with his knowledge of the field and I was not surprised that he was the first to graduate in my robotics class! He was a goto person while we studied for quals. Changhyun Choi for being an exemplary PhD student. Rowland O'Flaherty and Daniel Pickem are my goto controls experts. In order to get us in shape, Martin Levihn led tortuous running sessions in the mornings inspired by Arnold Schwarzenegger. Baris "The Wise" Akgun for being a good sounding board and teacher of robotics. Brian Hrolenok and Brian Goldfain for helping me sort out ideas at tea time. Alan Wagner for his advice in navigating the academic waters to graduation. Ben Amor Heni for his friendship and putting research into perspective! He was a fantastic sounding board and helped point me in crucial directions for research. Anthony Kuntz for reminding me to enjoy life and that we are all just people. Tim and Diane Crawford for great escapes to Colorado.

My extra-curricular activities helped keep me balanced. I thank my coaches and mentors for inspiring me, instilling me with work ethic, and keeping me focused on the end goal: Gary Dyals, Danny Dring, Mike Allen, Mike Lee Kanarek, Russell Wright, Craig Smith, Robert "Brock" Gaston, Matt and Dennis Scheib, Willard Jule, John Clarke, and Garret Machine. I especially want to thank Joe Lewis for his inspiration and being the unifying force for so many. He always emphasized that the greatest impact you can make is that of being a coach and mentor. His passing left a large vacuum as he was a coach, mentor, and idol to many. In Joe's legacy I want to thank the people I have had the privilege of working with and who have encouraged and inspired me: Dwayne Wright, Will Oswald, Aleksy Avila Vila, Scott Henson, Steve Kushner, Brad Walsh, Mercedes Gaffney, Shawn Wells, Kenneth Blackstone, Andrew Heymann, Eric Green, Chris Voso, Jason Brown, Mike Thompson, Robby Mixon, Angel Lopez, Chris Goddard, Dusty Pinion, Alex Westberry, and William

Rafael Ascencio.

My family instilled in me work ethic and an appreciation for education at an early age. My mother and father, Stella and Dimitri, were pillars of example with consistent determination in all that they approached. They never once wavered during the pursuit of my PhD. I hope to contribute to academia in robotics as they have in their respective fields. I thank my brothers Peter and Alexei for reminding me which personality qualities I should encourage in myself. Lastly I acknowledge my fiancée Kate Wagner and her family. Her parents Linda and Henry Wagner were welcoming and supportive of my odd work hours during family vacations. I thank her brothers Luke and Erik for being examples of service yet encouraging the nerd within. Kate provided epic support and understanding. Together we are such a great team because her strengths make up for my weaknesses. My friends and co-workers can agree that her cooking encouraged great success!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

One key to more effective cooperative interaction in a multi-robot team is the ability to understand the behavior and intent of other robots. Observed teammate action sequences can be learned to perform trajectory recognition which can be used to determine their current task. Previously, we have applied behavior histograms, hidden Markov models (HMMs), and conditional random fields (CRFs) to perform trajectory recognition as an approach to task monitoring in the absence of communication. To demonstrate trajectory recognition of various autonomous vehicles, we used trajectory-based techniques for model generation and trajectory discrimination in experiments using actual data. In addition to recognition of trajectories, we introduced strategies, based on the honeybee's waggle dance, in which cooperating autonomous teammates could leverage recognition during periods of communication loss. While the recognition methods were able to discriminate between the standard trajectories performed in a typical survey mission, there were inaccuracies and delays in identifying new trajectories after a transition had occurred. Inaccuracies in recognition lead to inefficiencies as cooperating teammates acted on incorrect data. We then introduce the Trajectory Adaptation for Recognition (TAR) framework which seeks to directly address difficulties in recognizing the trajectories of autonomous vehicles by modifying the trajectories they follow to perform them. Optimization techniques are used to modify the trajectories to increase the accuracy of recognition while also improving task objectives and maintaining vehicle dynamics. Experiments are performed which demonstrate that using trajectories optimized in this manner lead to improved recognition accuracy.

Acronyms

AxV - Autonomous Vehicles with the x standing in for U - underwater, G - ground, A - air, and S - surface.

ASV - Autonomous Surface Vehicle which operates on the surface of a body of water.

AGV - Autonomous Ground Vehicle

AAV - Autonomous Air Vehicle

AUV - Autonomous Underwater Vehicle

UAV - Unmanned Aerial Vehicle

UMS - Unmanned Maritime Systems can be defined as unmanned vehicles that displace water at rest and include two subcategories: UUV and USV.

UAS - Unmanned Aerial Systems

UUV - Unmanned Underwater Vehicles

USV - Unmanned Surface Vehicles

CHAPTER I

INTRODUCTION

The primary contribution of this dissertation is the adaptation of autonomous vehicle trajectories for improved recognition by autonomous teammates. The Trajectory Adaptation for Recognition (TAR) framework is introduced which is capable of adapting trajectories based on desired task properties, adhering to vehicle dynamics, and increasing recognition accuracy. Such capabilities are especially useful for autonomous vehicle team cooperation through task recognition. This dissertation begins by presenting foundational work by performing trajectory recognition of autonomous teammates with simulated and actual data while introducing strategies on how an observing teammate can perform such recognition. Circumstances in which recognition of an autonomous teammate are useful include when operating in environments where communication is intermittent or unavailable and when teammates are malfunctioning or have nefarious intent. Subsequently, further foundational experiments are performed in which robot teams leverage trajectory-based signaling, inspired by the honeybee "waggle" dance, enabling one autonomous vehicle to signal another to a location of interest. In this cooperation, an autonomous vehicle signals with an *InfinityPattern* while an observer performs recognition and visits the location of interest when it perceives the correct pattern. The TAR framework directly addresses the difficulty in creating or adapting trajectories for trajectory-based task recognition or signaling, which was performed in the foundational work as tedious manual iterations to achieve acceptable recognition accuracy. The framework automates the adaptation of trajectories for improved recognition accuracy while maintaining common task goals.

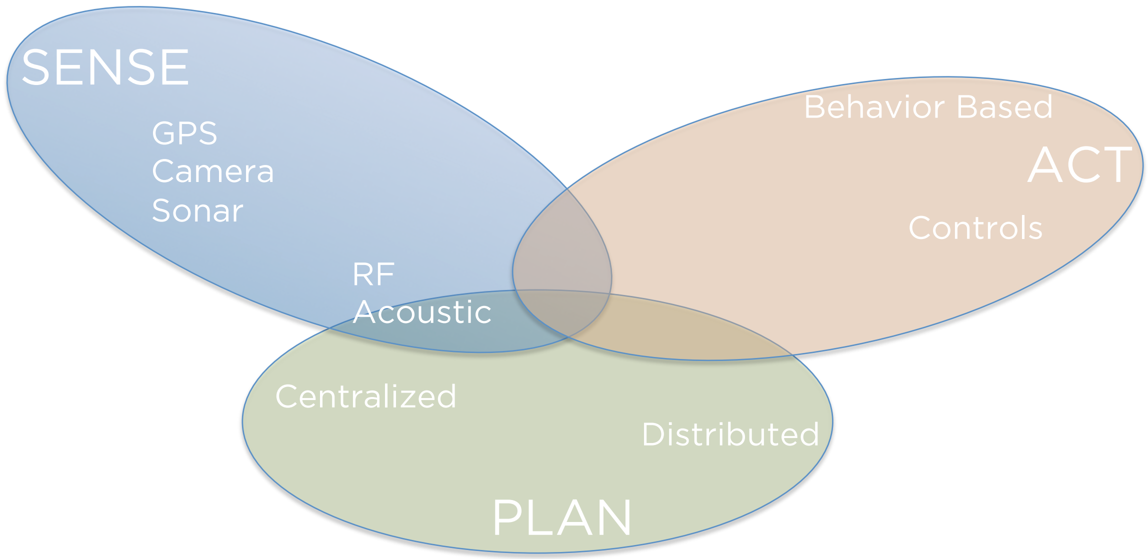


Figure 1: General Robot Autonomy Sense-Plan-Act. Robot sensing includes gathering information about the world though different sensors, such as a camera or sonar, as well as through communications, such as radio or acoustic modem. Robot planning includes creating world models and plans of action given the sensed information. Robot acting is the execution of a plan such as following a trajectory. While the general sense-plan-act paradigm considers each step separately, the reality is that there is overlap between all the areas.

1.1 Research Questions

The sense-plan-act paradigm is a general formulation of robot autonomy in which to situate the work in this dissertation [2]. The cycle begins with an autonomous robot gathering information about the world via sensors such as a camera and through communicating with other robots or operators. The plan step is where the robot creates a model of the world with the gathered information and formulates a plan of action. The act step is the execution of the plan produced in the previous step. While the sense-plan-act paradigm is a general formulation, the reality is that there are overlaps and integration between the three areas, depicted in Figure 1. Within the three major areas of the sense-plan-act paradigm there are subareas, depicted in Figure 2, such as camera calibration for sensing and trajectory tracking for acting. In this work, the focus in the area of sensing is recognition. Recognition can focus

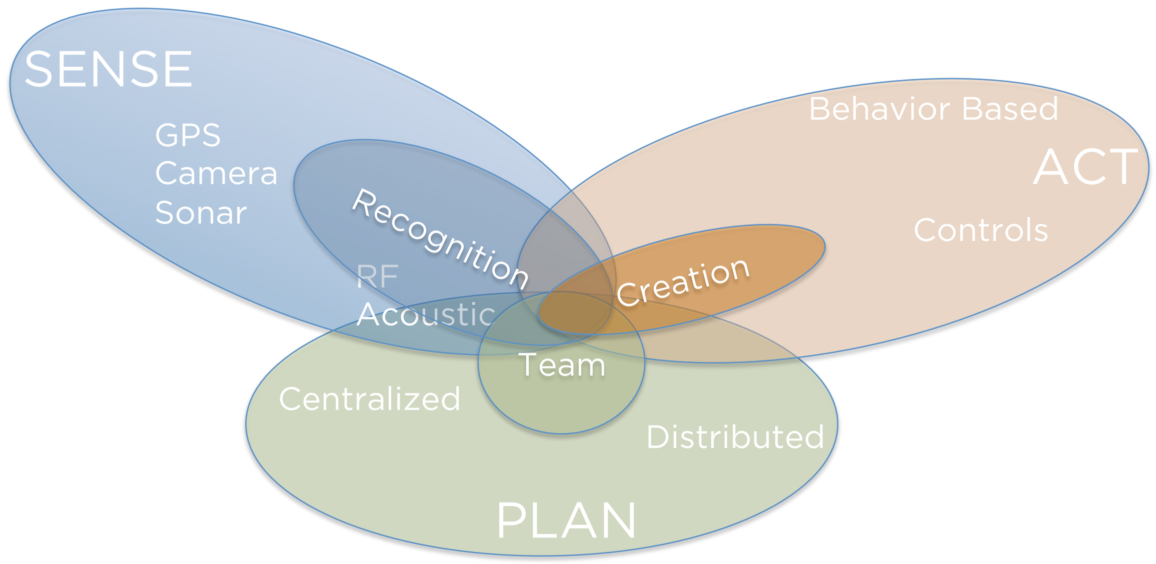


Figure 2: Subareas of Sense-Plan-Act. Relevant to the work presented in this dissertation are the subareas of recognition in sensing, team strategies in planning, and trajectory adaptation or creation in acting.

on objects, locations, or team plans. Specifically, recognition in this work focuses on trajectories created by autonomous vehicles such as cars, aircraft, and marine vehicles. The planning area can incorporate different world models and strategies. In particular this work focuses on the formulation of team strategies for cooperating robots. In the acting area there are several methods for generating trajectories and tracking generated trajectories. The focus of this work is that of trajectory adaptation. The first intersection that is addressed, as seen in Figure 3, is that of recognition of common trajectories to perform tasks such as search. The second focus of this work is the intersection of trajectory recognition and team strategies in which recognition of common task trajectories is leveraged for cooperation between two autonomous vehicles. The third focus is the intersection of trajectory recognition, team strategies, and trajectory creation in which trajectories are adapted for improved recognition which can be leveraged for team cooperation. Research questions addressed in this dissertation revolve around autonomous vehicle team coordination during periods of intermittent or denied communication.

1. Can trajectories for robots on a team be adapted to improve recognition such that the acting vehicle can perform them and yet still accomplish their desired task?

    (a) How should trajectories be described to support task accomplishment and recognition?

    (b) What are appropriate cost functions to achieve or maintain common autonomous vehicle tasks?

2. At what accuracy is it possible to recognize the common trajectories of autonomous vehicles?

3. What team strategies can be used to leverage trajectory recognition by autonomous vehicle team members?

1.2 Contributions

The following are contributions presented in this dissertation along with citations if previously published.

1. TAR: A methodology for evolving trajectories that simultaneously enables them to be separately recognizable while accomplishing their intended purpose [36].

2. Strategies for leveraging trajectory recognition for autonomous vehicle teams [35, 41, 38].

3. Expressing desired task using cost functions for common tasks.

4. Impact of strategies of an observing robot, stationary vs track and trail, on recognition accuracy [43].

5. Experimental results supporting TAR's ability to improve recognition accuracy while adhering to task objectives [37].

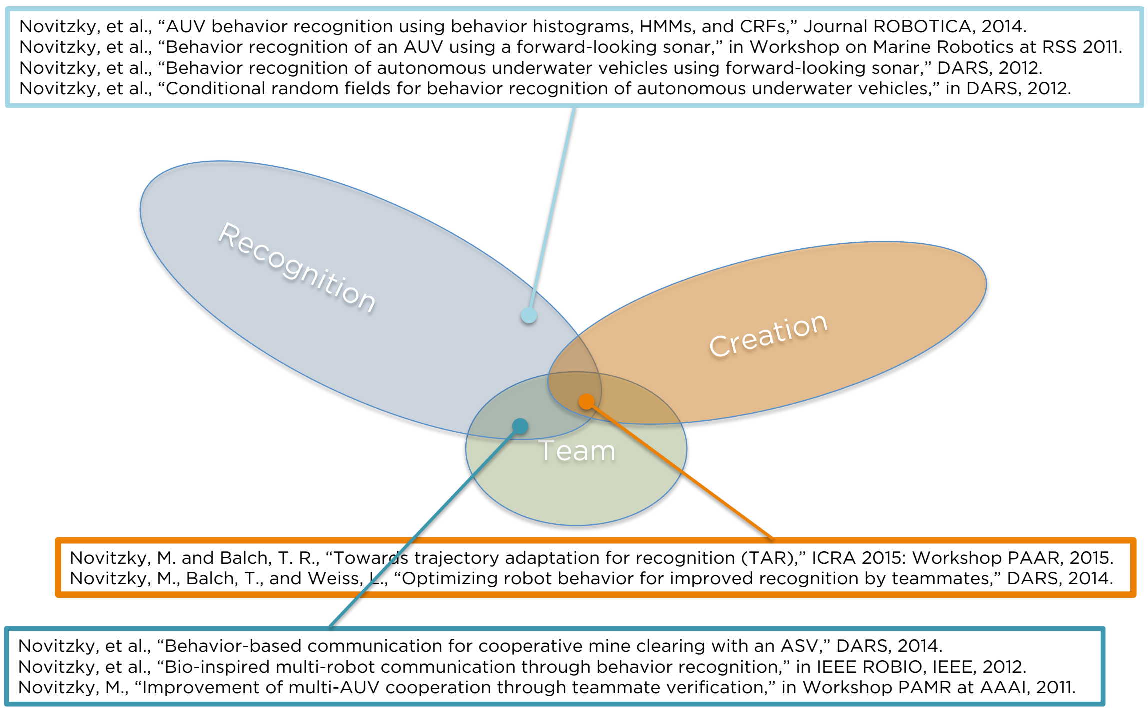

Novitzky, et al., “AUV behavior recognition using behavior histograms, HMMs, and CRFs,” Journal ROBOTICA, 2014.
Novitzky, et al., “Behavior recognition of an AUV using a forward-looking sonar,” in Workshop on Marine Robotics at RSS 2011.
Novitzky, et al., “Behavior recognition of autonomous underwater vehicles using forward-looking sonar,” DARS, 2012.
Novitzky, et al., “Conditional random fields for behavior recognition of autonomous underwater vehicles,” in DARS, 2012.
Recognition
Creation
Team
Novitzky, M. and Balch, T. R., “Towards trajectory adaptation for recognition (TAR),” ICRA 2015: Workshop PAAR, 2015.
Novitzky, M., Balch, T., and Weiss, L., “Optimizing robot behavior for improved recognition by teammates,” DARS, 2014.
Novitzky, et al., “Behavior-based communication for cooperative mine clearing with an ASV,” DARS, 2014.
Novitzky, et al., “Bio-inspired multi-robot communication through behavior recognition,” in IEEE ROBIO, IEEE, 2012.
Novitzky, M., “Improvement of multi-AUV cooperation through teammate verification,” in Workshop PAMR at AAAI, 2011.

Figure 3: Big Picture Contribution. The figure displays the intersections of the sense-plan-act subareas of recognition-team strategy-creation addressed in this dissertation. A corresponding list of previously published work appears in break out boxes. The top-most light blue set of citations are previously published work in the area of trajectory recognition of common tasks performed by autonomous vehicles through simulation and various sensors. The bottom-most dark blue citations correspond to previously published work of leveraging trajectory recognition for team cooperation. The middle orange list of citations correspond to the TAR framework in which trajectories are adapted for improved recognition while still achieving common tasks.

6. Recognition of autonomous vehicle trajectories in simulation, actual sonar data, and ASV data [39, 40, 42, 43].

Figure 3 depicts where these contributions and their previously published citations fit in the subareas of the sense-plan-act paradigm.

1.3 Thesis Statement

Trajectory adaptation enables more effective task recognition while preserving task completion.

1.4 Dissertation Outline

The remainder of the dissertation is organized as follows. Chapter 2 contains the related work including multi-robot coordination strategies, recognition of robot tasks and trajectories, and the leveraging of recognition for robot team coordination. Chapter 3 describes the common tasks and trajectories used to perform them for both manned and unmanned vehicles. Chapter 4 describes foundational work using various recognition methods on common trajectories. Chapter 5 describes foundational work on leveraging trajectory recognition for multi-robot cooperation. Chapter 6 contains the description of the TAR Framework and experiments demonstrating its effectiveness on common tasks. Chapter 7 contains concluding remarks.

CHAPTER II

RELATED WORK

In some circumstances, autonomous vehicles should be able to recognize the task of another autonomous teammate simply through observation. This would be useful if a teammate needs to verify another member's current assigned task. This allows for the correction of an erroneous teammate or one that has nefarious purposes. Another scenario where task recognition can be useful is when RF communications are intermittent or unavailable. This is the type of environment in which marine robots find themselves as both radio frequency (RF) and acoustic communications can be difficult. Multi-robot teams rely heavily on stable communication so that efficiency is achieved by not repeating tasks. Task recognition would help in these environments, since one member could observe another is performing a task and therefore know to move on to another. Since most robot task recognition schemes focus solely on improving recognition capability, the schemes neglect that the actor can modify the paths or trajectories followed. An analogy for this work is two SCUBA divers about to dive in the water. Before the dive, they discuss on the boat which hand gestures they will use to communicate underwater. Because these gestures vary by country or organization, it takes some repetition and negotiation to come to an agreement on which gestures will be used and what they indicate.

This chapter will present background and relevant work in the areas of multi-robot coordination, recognition of robot activity, and leveraging robot task recognition for team cooperation. Section 2.1 will cover robot coordination schemes. Section 2.2 will cover activity or task recognition. Section 2.3 will cover robot trajectory creation. Section 2.4 will cover trajectory creation/recognition schemes. Section 2.5 covers how activity recognition is leveraged for multi-robot coordination.

2.1 Multi-Robot Coordination

Multi-robot teams, in comparison with single robot solutions, can offer solutions that are more economical, robust to failure, and more efficient than single robot solutions [14, 9]. A team of robots can work on tasks in parallel, perform distributed sensing, and operate in multiple locations at once. Furthermore, multiple robots add redundancy to the system. Unfortunately, a tradeoff is that these teams must communicate and work together with the added uncertainty regarding the behaviors of other robots. For instance, a team member may have trouble cooperating due to communication errors, because they are busy performing other tasks, or have conflicting goals [2].

Many different methods for performing distributed cooperation exist, including centralized optimization algorithms and game theoretic techniques. A centralized method requires at least one agent or a home base to make task or role assignments. Although this may be optimal when communication links are reliable, its efficacy degenerates with intermittent communication, and a central organizer makes the whole system come to a halt if it fails. Thus, a decentralized approach is much more viable because it is more robust to failures of communication. Auction-based algorithms generally have low communication requirements (where agents coordinate tasks through bid messages). Therefore, they can be well suited to environments with communication constraints. Auctions can perform computations in parallel and the methods take advantage of the local information known to each agent [12, 18].

However, auction-based methods can still degrade in overall efficiency as communication deteriorates [54]. Poor communication environments are encountered by autonomous underwater vehicles (AUVs) using traditional acoustic modems for underwater communication. The effectiveness of acoustic communications deteriorates in the presence of surface reflections, bottom reflections, ambient noise, and noise sources within the water column, such as emissions from other vessels. Sotzing and Lane [58] have demonstrated that using teammate prediction improves overall performance of a cooperative AUV system. However, this type of system still needs communication in order to avoid degradation as task predictions accumulate error over time without correction from teammates. Novitzky [35] proposed a multi-robot cooperation framework that combines the flexibility and robustness of an auction-based method similar to [54], the prediction of tasks of [58], along with the addition of task verification. During periods of low prediction confidence, an autonomous robot could go to a task area and perform task/activity recognition of a teammate. If a teammate was performing the appropriate task then the robot would move on to another task in the team's queue. However, if the teammate was not in the area or was performing the wrong task, then the erroneously performed task would be placed back in the tasks to be performed queue. This dissertation focuses on the ability to recognize an autonomous vehicle's task as a step towards implementing such a system.

2.2 Robot Task/Trajectory Recognition

There are several distinctions that are made in order to categorically study the recognition of teammate behaviors [60], including plan, activity, goal, and intent recognition. There are several approaches to activity recognition including logic-based, topologically invariant, and control theoretic techniques [61, 22, 11]. Probabilistic graphical models such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) are popular as they are useful for pattern recognition and have proven robust to uncertainty in many areas including speech recognition [48, 31] and handwritten script recognition [33].

The most prevalent use of behavior recognition with robots has been through the use of an overhead sensor, such as a camera. Original work was presented by Han and Veloso [20] in which an agent's behavior was recognized using a Hidden Markov Model (HMM). The authors discretized input from a vision system overseeing a Robocup small-sized league robot interacting with a ball. The vision system reported the location and velocity of both the agent and the ball to several concurrent HMMs. A limitation of this system is that if an HMM was started at the wrong moment, it could miss the appropriate activation of the initial states. To alleviate this problem the authors ran an HMM for a specific behavior at intervals in order to capture the correct linear order of events.

Of specific importance to this work is that performed by Vail et al. [65] in which the authors compared the accuracy of CRFs and HMMs for activity recognition on robot systems. Their chosen domain was simulated robot *Tag*. In their simulation, two robots were passively moving from waypoint to waypoint while a third was the *Seeker* searching for a robot to *Tag*. As part of the analysis of CRFs and HMMs, the authors tested the accuracy with different observations such as raw positions only, including velocities, and chasing features. The authors also examined the effect of incorporating features which violate the independence assumptions between observations. The results showed that a discriminatively trained CRF performed as well as or better than an HMM in their robot *Tag* domain.

Vail and Veloso [64] used CRFs for analyzing roles in multi-robot domains. The authors experimented with two approaches to feature selection and applied these methods to data recorded during RoboCup soccer small-size league games. The goal of their work was to create a classifier that can provide useful information to robots that are playing against a team whose roles are being classified. They found that using feature selection can dramatically reduce the number of features required by CRFs to achieve error rates that are close to or identical to the error rate achieved by the model with its full complement of features. Reducing the number of features dramatically speeds up online classification and training.

Focusing on autonomous mobile robots in the marine domain, Baxter et al. [5] performed behavior recognition using HMMs on post-mission analysis of self-localization provided by an AUV. The post-mission analysis converted GPS pose trajectories to low-level actions such as *track-west* and *left u-turn east*. The main drawback of this method is that it claimed to be agnostic to the environment yet still required the use of cardinal direction, which in itself is still somewhat constraining to the compass orientation within an environment. The authors improved upon their discretization methods where they also enhanced HMMs to deal with behavior sequences of variable length [6]. They began with AUV location information from simulated sonar data. These trajectories were fed into a maneuver-recognition algorithm capable of identifying an AUV's actions such as *straight* and *veer-left*, thus making it more environmentally agnostic. While the algorithm was attempting to recognize top-level goals such as *mine-countermeasure* (MCM), *mine-countermeasure inspection* (MCMI), and *point inspection* (PI), it further divided the top-level goals into recognizable sub-goals which included *dive*, *track*, *right u-turn*, and *left u-turn* along with *inspection*. Their results also included the observation that top-level goals are achieved via the AUV performing sub-goal behaviors.

As this dissertation focuses on enabling a multi-robot team to work without the use of traditional methods of communication, it uses insect communication as inspiration. Insects use many modalities of communication: tactile, chemical, acoustic, and visual. The modality of particular interest is that of the honey bee's "waggle dance" [66]. During the waggle dance, as seen in Fig. 17b, a sequence of motions occur: arcing right, waggling, and arcing left. Waggling is when the dancing bee walks in a straight line while oscillating its torso left and right. Current research indicates that the orientation of the waggle portion of the dance represents the angle between a food source and the sun. The length of the waggle indicates the energy required to reach the food source. While the waggle dance is a tactile form of communication between honeybees, scientists use computer vision techniques to aid biologists studying their behavior [16, 44, 4]. The typical workflow for these computer vision techniques is to label primitive actions which in turn make up a behavior. Oh et al. [44] used labeled tracks of a honeybee performing the "waggle dance" so that a parametric switching linear dynamic system could learn and then accurately label each primitive motion. Work by Feldman and Balch [16] used a technique in which kernel regression labeled the primitive motions of a bee's trajectory. A Hidden Markov Model (HMM) was used to smooth the labeled sequence and subsequently identify the most likely behavior such as *dancer* or *active bee*.

2.3 Robot Trajectory Creation

There are many methods for creating robot trajectories. In particular we note the work of Ratliff et al. [52] who introduced the Covariant Hamiltonian Optimization for Motion Planning (CHOMP) motion planning algorithm. CHOMP is based on covariant gradient descent and is able to construct trajectories which optimize over a variety of dynamic and task-based criteria. The second work to note is that of Kalakrishnan et al. [27] who introduced Stochastic Trajectory Optimization for Motion Planning (STOMP). STOMP is a stochastic trajectory optimization framework which allows for a combination of costs that do not have gradient information. More recently, Schulman et al. [56] introduced TrajOpt which is a trajectory optimization algorithm which incorporates collision avoidance. TrajOpt uses sequential convex optimization techniques with penalties for constraints. The work presented in this dissertation aligns more in the spirit of STOMP as a stochastic search method is used in the TAR framework in order to allow for flexible and general cost functions.

2.4 Trajectory Creation/Recognition Schemes

In this work we focus on not only recognizing autonomous robot tasks but also on potentially modifying their execution in order to be more easily recognized. Thus, a particular topic of interest is action/recognition schemes that are capable of both creating trajectories and recognizing them in the same framework. Mirror neurons have been a source of inspiration in the action/recognition area. Mirror neurons fire both during the generation and the observation of goal-directed actions [53, 17]. Goal-directed is described when an action is directed towards an object. This discovery indicates that mirror neurons are the circuitry used by the motor system to both recognize and generate actions. Inamura et al. [24] use mimesis theory from cognitive science which describes the existence of "mirror neurons" as evidence that behavior generation and behavior cognition are not independent. The authors propose a "mimesis model" that is a mathematical model that abstracts the whole-body motions as symbols, generates motion patterns from the symbols, and distinguishes motion patterns based on the symbols. An observer views a motion pattern of a performer and acquires a symbol of the motion pattern. The observer can generate the motion by performing it themselves. Hidden Markov Models (HMMs) are used for this mathematical framework for both recognition and generation.

One action/recognition scheme which is popular is the use of dynamic movement primitives (also called dynamic motor primitives, DMPs). In general, DMPs generate trajectories using a spring-damper system with a goal location as an endpoint and Gaussian basis functions as forcing functions along the motion [55]. Akgun et al. [1] studied how actions can both be recognized and generated by a Dynamic Movement Primitive (DMP) for online action recognition. The authors were able to demonstrate that their modified DMPs were capable of online action recognition with approximately the first one-third of the observed action with a success rate of over 90%. Akgun et al. used neural networks as their function approximators. The

TAR framework is similar to DMPs in that trajectories are approximated with radial basis functions (RBFs), explained in detail in Section 6.2. This allows for trajectory recognition to occur by comparing the weights of a template trajectory to that of a sample trajectory. The same RBF weights can be used to produce the trajectory that an autonomous robot should follow to perform a task. Trajectory adaptation can also be performed in the RBF weight space as perturbations of the weights create different trajectories.

2.5 Action Recognition for Robot Team Coordination

Lynne E. Parker [46] published work as early as 1995 investigating the impact of robot awareness of team member actions and its effect on cooperative team performance. The task investigated was a puck moving mission. Variations in the experiment include number of teammates (team size) and the level of teammate awareness of each other's actions. The metrics used for evaluation were time and energy. The results show that team performance varies by the size of the team, level of awareness, the amount of work available per robot, and the cost of replicated actions. Robot awareness is the same as recognition of team member actions through passive action recognition. The robots are required to locate the pucks, move them to a goal location, while reporting progress to human operators. Parker used the ALLIANCE software architecture which is a behavior-based, fully distributed architecture which allows for adaptive action selection to achieve fault tolerant cooperative control in robot missions. For this work, ALLIANCE was modified to have a consistent update by each robot through broadcast communication. This was the surrogate that later could be replaced by the use of passive action recognition. The five variations of the move puck experiment were: two robot team with full awareness, three-robot team with full awareness, two-robot team with no awareness, three-robot team with no awareness, and a single robot operator. Each experimental condition was run ten times while recording action selection and duration. Energy required was calculated by assuming that a robot at rest consumed no energy but any sort of actuator use equaled a unit of quantity of energy per unit time. The worst time requirement was the single robot scenario. Even adding robots to the team without awareness helps reduce the time required for this task. Overall, the three-robot team completed the task faster than the two-robot team regardless of awareness level. In both cases, the action aware teams performed better than the teams without awareness. It is interesting to consider why a three-robot team without awareness would perform better than a two-robot team with awareness. The answer is in the type of task being investigated. The pucks that are removed from the environment can be sensed without communication. the time requirement to move the puck is stable across both awareness and no awareness variations. The portions of the task that are sped up with awareness are find-locations and report-progress. If the cost of replicating these portions of the task is low then a non-aware team has less impact. With regards to energy usage, teams with no awareness are prone to duplicate tasks and thus waste energy. The larger the team without awareness, the greater the chance for needless duplication of tasks. Awareness increases energy efficiency as it reduces duplicated tasks. Overall, the puck moving experiments shows us that if the effects of a task cannot be sensed through the world, such as a puck removed, lack of awareness causes an increase in energy use. The more robots to a team without awareness means an increase in duplicate actions.

The most similar work to that presented in this dissertation is by Kuniyoshi [30] in which the author used binocular vision for teammate and resource observation. Kuniyoshi coined the term *Cooperation by Observation*, defined as "Observing other agent's action, and choosing appropriate actions regarding the observed action and the current task situation." In essence, the author extends a behavior-based architecture to allow for dynamic matching of behavior resources and behavior types. Kuniyoshi tested this framework on three tasks: posing, unblocking, and passing. In each task, a teammate was observed along with a common resource. For example in the unblocking task, robot i is pushing a block and robot j determines an obstacle is in robot $i's$ path. Robot j then moves the obstacle so that robot i can deliver its block to the appropriate location.

Dragan et al. [13] concentrate on human-robot collaboration. It is important for seamless collaboration that the robot make its intentions clear to the human collaborator. This is similar to our own work in the sense that Dragan et al. have termed that a robot's motion be legible. The authors distinguish between motion properties of predictability and legibility with a formal mathematical framework. As an example, the authors use the task of collaboratively clearing a table. Predictable motion is motion that is expected (observer knowing the goal action matches what they would predict to happen). Legible motion is intent-expressive (observer predict the goal based on the motion). The human observer watches a humanoid robot's motion trajectory and is modeled as running an inference trying to predict the robot's goal. A legible motion is one that a human observer can watch and determine the goal in the shortest amount of time. The authors give the robot a kinesthetic form of communication - intent through motion. They used a model based on the theory of action interpretation in psychology. This model allows for evaluating how legible a motion trajectory is and correlates with legibility in practice. In addition to being able to model/evaluate how legible a motion trajectory can be to a human, the authors also introduced a method of motion trajectory generation so that a human properly infers the robot's goal. "Trust region" of predictability limits the exaggeration of the humanoid robot's motion trajectory so that it is still predictable. Placing our work in context with that of Dragan et al. [13], we too are searching for motion trajectories that are legible. The difference is that the observer in our case is not a human but another robot. In our work, we assume the robot is limited by its sensor model and viewpoint of its teammate performing a legible motion trajectory. In our task we are not trying to max the $P(G|trajectory)$, or probability of the goal given the trajectory. Our goal is to recognize our teammate's task which can be either search, reach a waypoint, or loiter while waiting for a command.

Raghunathan and Baillieul have approached gesturing to signaling between mobile robots in a control theoretic manner [3, 49, 50, 51]. The work decomposed signaling into two parts: ability of the gesturing agent to generate a gesture as a motion and ability of the receiving agent to perceive the gesturing and perceive it according to a set. The work of Raghunathan and Baillieul [49] consider the relative motion of two Dubins' (two wheeled non-holonomic robots) vehicles in a plane and analyze the transmission of a signal. The authors formulate the two tasks as a pair of nonlinear control problems. The transmitting robot is required to track the states of the receiving robot and overlay the transmitted signal on the other motion mode for the receiver robot. This way the receiver robot can sense the transmitted signal with an onboard range sensor. The sensors considered are sonar, lidar, etc. and that the distance min and max are set and the relative motion of the transmitter is within these bounds. Between two robots, the transmitting robot must keep up with the receiver while also trying to send a space curve signal. In the analysis of this setup they assume the receiver only travels along the positive x-direction. The transmitting signals were required to be spatially periodic and not of infinite duration and bounded so that the separation between transmitter and receiver does not diverge. The transmitting robot was assumed to traverse the entire spatial signal in finite time. The authors do account for the gesturing limitations based on the dynamics of the non-holonomic robots. The slope of the function in which signaling occurs cannot exceed the velocity limits of the transmitting robot. A curve, C, is a feasible trajectory for the variable speed Dubbins vehicle if there exists a control law pair in velocity and angular rate which can steer it along C with no error. A finite extent curve C is observable under an observation model if one can reconstruct C perfectly after observing it using the observation model for a finite duration. The types of curves studied were sinusoidal signals [49], Fourier cosine series [50], and Chebyshev polynomials of the first kind [51].

An important facet of this work is the ability to distinguish signals between each other. Raghunathan and Baillieul [50] use the Euclidean distance (L_2 norm) between relative distances between two vehicles throughout a specific length trajectory. Then define a lower-bound between the distance between two signals that the receiver can identify under perfect observation and imperfect but bounded tracking control. This is all accomplished without noisy observations.

Raghunathan and Baillieul [51] extended their previous work to a fully distributed strategy. The system first requires a motion based signaling protocol which is a sequence of pre-determined motions. The next requirement is a signaling control law to achieve the signaling protocol. They explore several scenarios, including signaling to a stationary receiver. Other scenarios include two robots traveling inline and the modulation of distance of the trailing vehicle. A third scenario is a trailing transmitter trailing at an angle. At first a syncing period starts where the transmitting robot follows in parallel without any modulation. After this pre-specified time, the transmitter will modulate their trajectory to signal which will be followed by another purely parallel track to indicate termination of the signal. In this work the authors introduced that the receiver acknowledge, through some motion, to the transmitter that it has seen the herald (also called synching). The authors use a codebook β composed of a finite number of polynomials from the set of Chebyshev polynomials of the first kind. The polynomials have a desired property that guarantees bounds on the minimum and maximum separation between the two agents. The polynomials are scaled to ensure proper tracking performance which is determined by certain curvature and slope bounds. This work is improved over their own previous literature in that it concatenated 4 polynomials together rather than just one signal.

The work of Raghunathan and Baillieul is looking at specifically signaling between two robots using relative motion. They are not also attempting to perform a task such as search or circle loiter. Additionally, their formulation only specifies single valued functions. Any curve that can have two values along the x-axis is not available. For example, in our work we have a circle loiter as a task which is not capable of being represented.

Jones and Anderson [26] introduced control policies for trajectories that optimized a joint expression of control energy and robustness to observation noise for the linear time invariant case. The authors look at several aspects of such a system. The cost of transmission of a symbol is the energy of a control signal. They define the expected cost of transmission for the set of symbols as the summation of the cost to transmit a signal multiplied by the probability of transmitting that signal for all signals. The distinguishability of two symbols is the integrated euclidean distance between each timestep. Based on their formulation they assume that the two symbols are of equal duration. They use the L2 (Euclidean) norm. The total distinguishability then is the summation of every pair of distinguishability measures. The authors formulate the optimal control problem as minimizing the control input with weights emphasizing the importance of minimizing the expected control energy and maximizing the distinguishability. The authors use LQR framework to find minimum energy and maximum separation trajectories for encoding signals in motion. They do note that the LQR framework is limited in the number of symbols it can support. There was no observation noise and the trajectories were assumed to be fully controllable and fully observable.

This dissertation not only seeks to recognize trajectories for different tasks but also to modify them for improved recognition accuracy. We will use intended recognition as described by Kanno et al. [28] as occurring when the observed agent is aware of the observer and actively cooperates. The modification of trajectories is similar to state of the art in human gesture recognition schemes such as MAGIC 2.0 [29]. MAGIC allows designers and users to introduce and modify gestures for improved recognition accuracy. The MAGIC 2.0 system can even recommend alternative gestures to a human designer based on new gesture confusion. In a similar fashion, a team of autonomous vehicles will begin with a set of template trajectories that accomplish a set of tasks. If there is need for recognition accuracy improvement, then our framework will adapt those trajectories so that they are capable of being followed, still perform desired tasks, and increase recognition accuracy.

CHAPTER III

PROTOTYPICAL TASKS

The types of trajectories or behaviors used in this work are standard for manned, unmanned, and autonomous vehicles. In particular, we focus on maneuvers performed during Search and Rescue (SAR) and Intelligence Surveillance and Reconnaissance (ISR) missions. The standard for search and rescue is the International Aeronautical and Maritime Search and Rescue Manual (IAMSAR) [45] which is published by both the International Maritime Organization (IMO) and the International Civil Aviation Organization (ICAO). The IAMSAR is supplemented by organizations such as the U.S. National Search and Rescue Committee [57] and the U.S. Coast Guard [19]. Military branches also publish their own SAR procedures such as the NAVY' Search and Rescue Tactical Information Document (SAR TACAID) [34]. These documents outline the procedures for coordinating search and rescue operations which includes different search patterns based on target type and last location information along with the sensors being deployed including the number and type of searching vehicles. Of concern in this work are the search patterns performed by one vehicle such as parallel track search, seen in Figure 4, and creeping line search. The parallel track search pattern is used when the search area is large and the goal is uniform coverage. The creeping line pattern is performed when the search area is narrow and the target is assumed to be close to the closest start point. It is important to keep in mind that search patterns are also referred to as survey or broad-area coverage. These search patterns are standard and have been used for Anti-Submarine Warfare (ASW) as well [23]. For example, in addition to be used for search to conduct a radar/FLIR/MAD search, the same patterns can be used to lay down sonobuoys. In addition to search

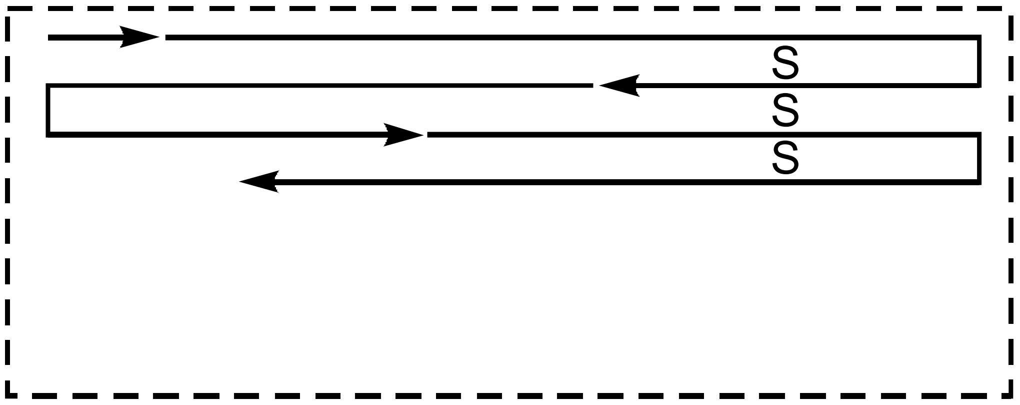


Figure 4: Parallel Track Search Pattern credit Navy SAR [34]

patterns we also use the loiter maneuver. In general, loitering is a maneuver that involves the vehicle cruising for a certain duration over a small region. In general aviation, loitering is a phase of flight that typically occurs at the end of a flight while the vehicle is waiting for clearance to land. In a military context, a loiter may occur over a known target location or during a search operation depending on the type of target and sensors involved.

Unmanned and autonomous vehicles have adopted the standard SAR and ISR maneuvers. The NATO Standardization Agreement (STANAG) 4586 [10] has standardized the way in which current and future UAVs in NATO Combined/Joint Services operational environments communicate with each other and their required capabilities. These capabilities include the standard communications and tracking of search patterns and loiter patterns such as circle, racetrack, figure 8, figure 8 with a bearing, racetrack with a bearing, etc. Example loiter patterns implemented in STANAG 4586 are seen in Figure 5. Loitering is of particular importance for autonomous aerial vehicles as it is one of their strengths that they can stay at station longer than manned vehicles while performing area overwatch. AUVs loiter for specific reasons due to their environment including difficulties in communication [68]. If an AUV were to simply stop its motion then the current may move the vehicle from its intended resting position. Additionally, an AUV may be neutrally buoyant at different depths making

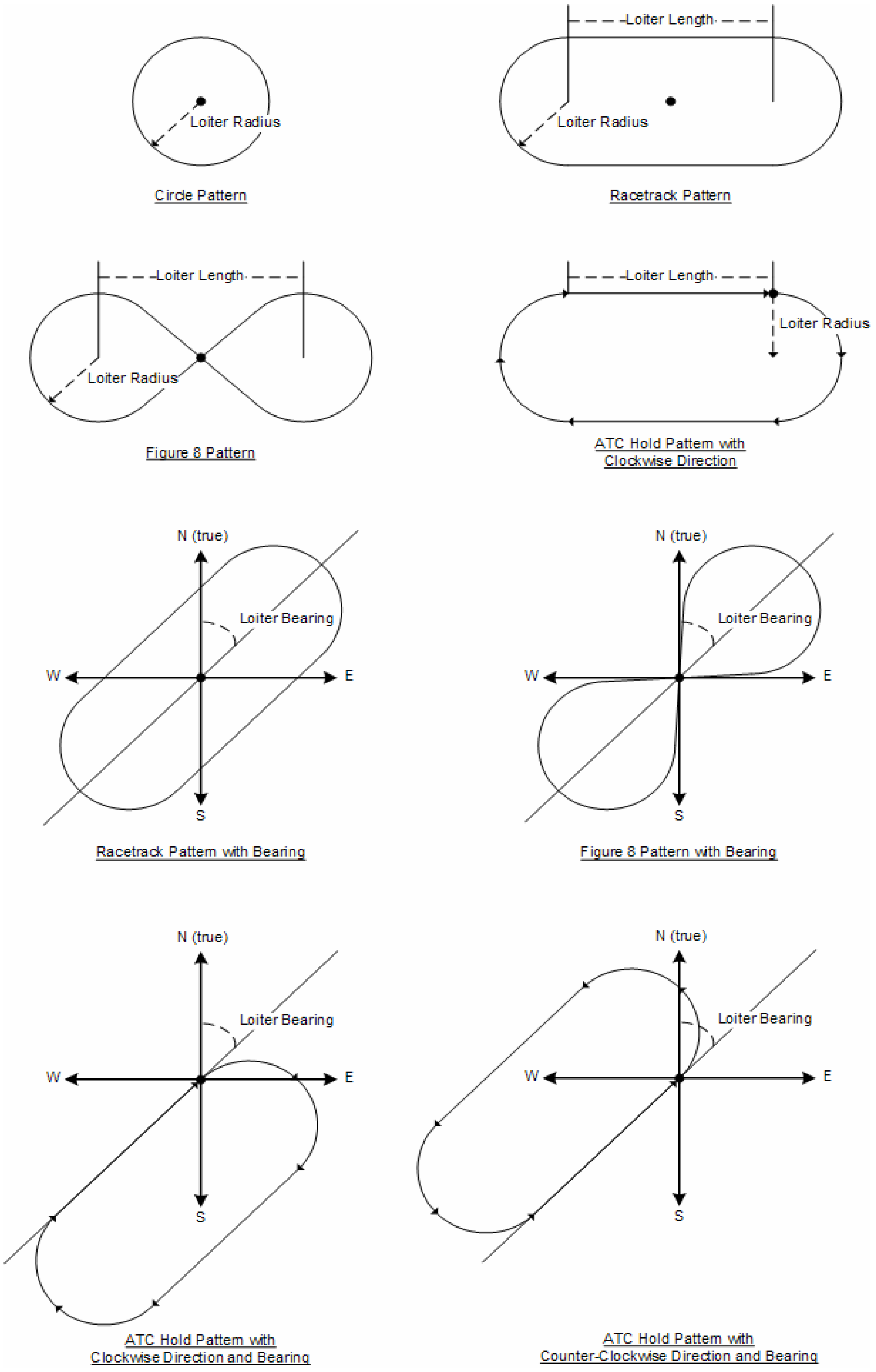


Figure 5: Implemented Loiter Patterns credit STANAG 4586 [10]

recovery more difficult if it stops moving and then drifts away. Loitering can be performed because it has been commanded and so that data transfer or further mission parameters can be communicated. When a mission aborts and is not required to immediately surface. Also, loitering occurs upon mission completion. During a mission, there may be several loiter locations based on time outs towards a waypoint. For example, a vehicle may not make it to a waypoint within 5 minutes and thus will switch into loiter at a specific location. The loitering at these specified locations allow for the mission not to be totally abandoned and allows for new mission parameters to be received from the control station. Beyond certain waypoints having a failsafe loiter location, segments of a trajectory may also have a loiter location specified. This allows for the AUV to switch to loitering in the shortest amount of time as the loitering points are themselves close to the current segment being travelled. This additionally allows command and control to quickly determine at which point in the mission the AUV was in based on the loiter point.

CHAPTER IV

FOUNDATIONAL WORK: RECOGNITION OF TEAMMATE TRAJECTORIES

In this chapter we present experiments in recognizing teammate trajectories from different sensors and team coordination strategies. The ability to recognize robot teammate tasks would be beneficial in communication denied environments [35]. This work is drawn from previous publications [39, 40, 42, 43]. While this foundational work demonstrates trajectory recognition of common autonomous vehicles tasks, there are situations in which the recognition accuracy is below an acceptable level for deployment. It is exactly these situations that motivate the concept of adapting the acting vehicle's trajectories for improved recognition accuracy. This is directly addressed with the introduction of the trajectory adaptation for recognition (TAR) framework in Chapter 6.

4.1 Introduction

This chapter focuses on trajectory recognition in an underwater application as a substitute for communicating through acoustic transmissions, which can be unreliable. The importance of this work is that sensor information regarding other agents can be leveraged to perform trajectory recognition, which is activity recognition of robots performing specific programmed behaviors, and task-assignment. This work illustrates the use of Behavior Histograms, Hidden Markov Models (HMMs), and Conditional Random Fields (CRFs) to perform trajectory recognition. We present challenges associated with using each recognition technique along with results on individually selected test trajectories, from simulated and real sonar data, and real-time recognition through a simulated mission.

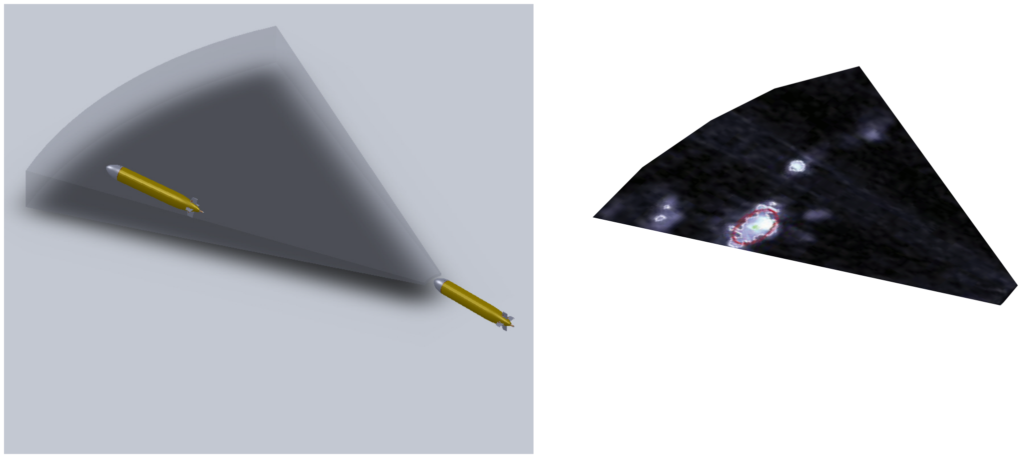


Figure 6: An AUV using its forward-looking sonar to *Track & Trail* a leader AUV in order to perform trajectory recognition.

The motivation for this work is the need for multiple small AUVs to perform autonomous research operations in underwater environments such as Georgia Tech Research Institute's Yellowfin AUV [67] research platform, seen in Figure 7. Because of the vehicle's size, power constraints, and operating environment, communication bandwidth is limited. We envision a cooperation system similar to that proposed by Novitzky [35] which will utilize auction-based methods augmented with prediction of teammate tasks during periods of degraded communication. If the confidence in a prediction of a teammate's task is low, then an AUV can perform prediction verification, illustrated in Figure 6, through trajectory recognition, as suggested by Baxter et al. [5]. Additionally, this handles the situation where an AUV may have been assigned a task only to discover another agent already performing the task but not communicating. The work in this chapter specifically focuses on whether trajectory recognition of an AUV is possible using Behavior Histograms, Hidden Markov Models (HMMs), or Conditional Random Fields (CRFs) and which is more suitable for the system described above.

Unlike previous trajectory recognition work in the AUV domain, our work does

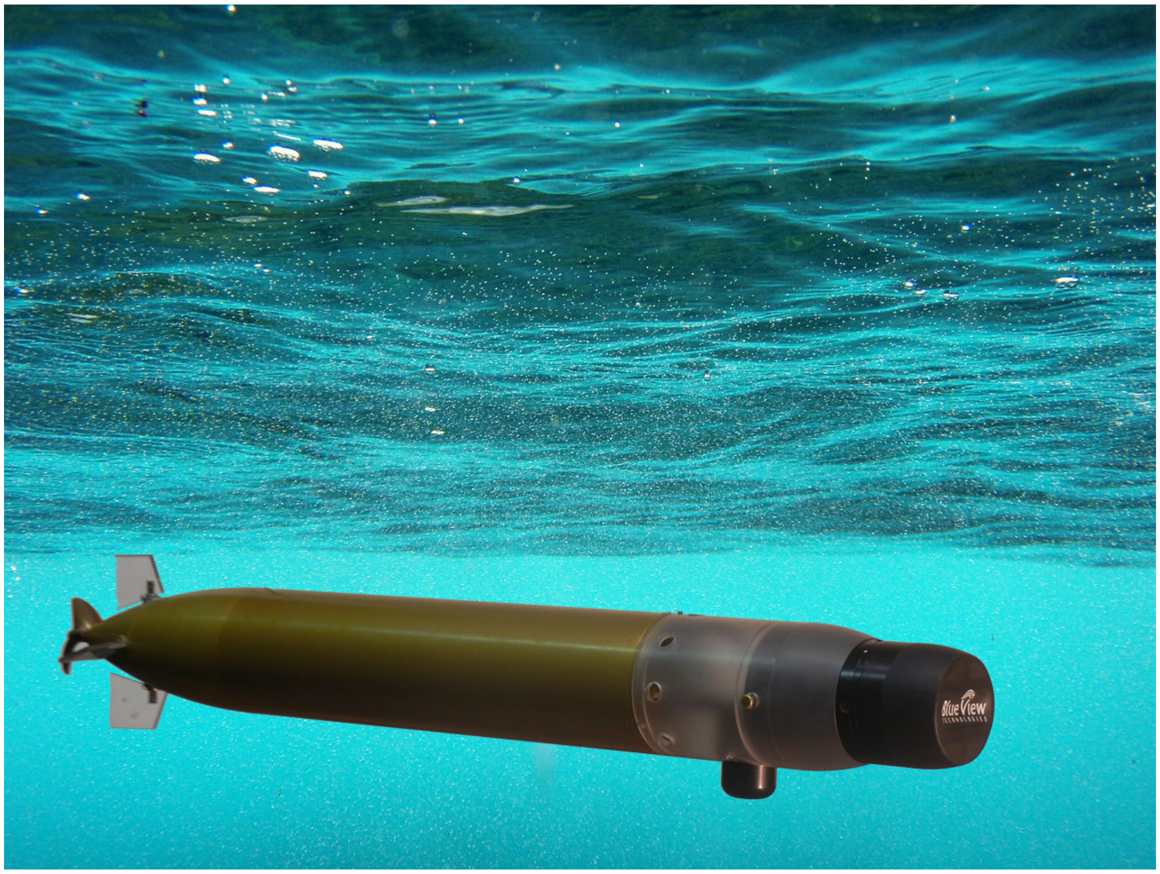
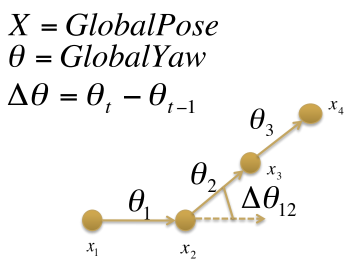


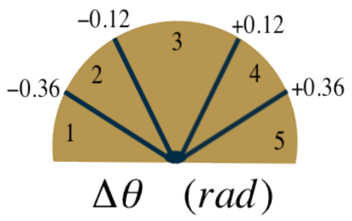
Figure 7: Yellowfin Autonomous Underwater Vehicle - designed to be man-portable for oceanographic observation.

not rely on trajectories provided through post-mission analysis nor only through simulation. Furthermore, it performs trajectory recognition of an AUV through the use of a simple discretization method, resulting in only one feature, on both simulated trajectories and actual sonar data comparing the results of a method using Behavior Histograms, HMMs, and a CRF.

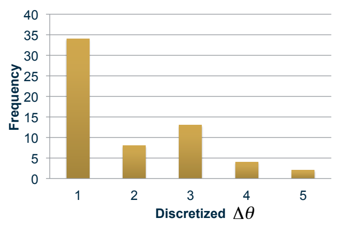
While this work uses high-frequency sonar data to validate the ability to perform trajectory recognition of autonomous underwater vehicles, it is feasible to use other sensor modalities. Airborne LIDAR has been used for bathymetry, underwater mine detection from helicopters, and shallow underwater target detection from surface vessels [25, 32, 63]. Thus, it is feasible that Airborne LIDAR could be used by a heterogenous teammate in the form of an autonomous aerial platform to detect the



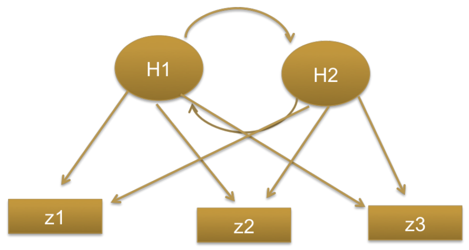
(a) Location to $\Delta\theta$.



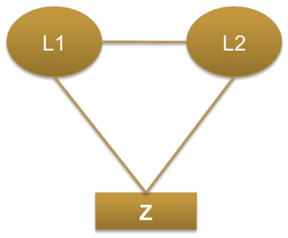
(b) $\Delta\theta$ discretization.



(c) Behavior Histogram.



(d) HMM.



(e) CRF.

Figure 8: In (a) an AUV's location over time is used to determine its global yaw. The change in global yaw from one time step to the next is encoded as an integer value which represents a given range, as seen in (b). The three trajectory recognition methods are Behavior Histograms, Hidden Markov Models, and a Conditional Random Field, as seen in (c), (d), and (e), respectively.

location of an AUV. Another source of teammate localization data underwater is through the use of hydrophone arrays which have been capable of localizing targets and AUVs [15]. Thus, a teammate capable of interfacing with a hydrophone array can perform trajectory recognition from a remote location.

4.2 Trajectory Discretization

The encoding method used is agnostic to any environment. The only measurement required is the location $\mathbf{x} = (x, y)$ coordinates of an AUV in a fixed 2D plane, as seen in Figure 8a. The motion model of the AUV is assumed to be non-holonomic and always moving with a forward motion similar to a tricycle model. The yaw of the AUV is calculated from the vector of motion from one measurement to the next.

$$\Delta\mathbf{x}_{(t-1,t)} = \mathbf{x}_t - \mathbf{x}_{t-1} \tag{1}$$

$$\theta_t = \arctan(\Delta\mathbf{x}_{(t-1,t)}) \tag{2}$$

$$\Delta\theta_t = \theta_t - \theta_{t-1} \tag{3}$$

The encoding used in this research is the change in yaw between measurements. Possible changes in yaw are discretized according to bins, as seen in Figure 8b. Each bin corresponds to a range of values. Bin 3, for example, represents a change in yaw between -0.12 and +0.12 rad. As seen in Figure 8b, an AUV moving straight ahead is observed as having no change in yaw and thus encoded as a 3 while one turning by 0.26 rad is encoded as a 2. A series of these encodings are combined into a trajectory string for input into the Behavior Histograms, Hidden Markov Models (HMMs), or the Conditional Random Field (CRF).

4.3 Recognition Methods

This chapter focuses on trajectory recognition of an AUV performing the trajectories *GoToWaypoint*, *Loiter*, and *SearchPattern*, which can be seen in Figure 9. The *GoToWaypoint* is the simplest of the three trajectories as the AUV proceeds from its current location to a defined waypoint. The *Loiter* trajectory is performed while waiting for instructions and is created by the AUV following a set of waypoints placed along the perimeter of an octagon. The *SearchPattern* trajectory is typically performed while trying to cover a rectangular area by following waypoints placed along evenly separated segmenting lines. In this chapter, we assume the trajectories are performed atomically, meaning no other behaviors are running in parallel. In general, observations are labeled as $Z = \{z_1, ..., z_T\}$ where the index represents successive time steps. In our domain z_t contains an integer value of the change in yaw of the AUV, described above. In the HMM method each hidden state H may not have an explicit definition. In the CRF method the labels L_t are drawn from one of the three trajectory labels. Trajectory recognition is the ability for each method to determine one of the three trajectories as the most likely being observed. Accuracy is defined as the recognition method properly identifying the trajectory being observed.

4.3.1 Histogram Matching

The baseline recognition method uses histogram matching. After a given trajectory is discretized, as described previously, a histogram is created from the possible encodings. As an example, if a trajectory is discretized into five possible changes in yaw then a histogram of that trajectory will have five bins, as seen in Figure 8c. The height of each bin will reflect the frequency of that change in yaw throughout the trajectory. Each histogram is normalized.

4.3.1.1 Template Histogram

A template histogram is made for each trajectory that we wish to recognize. A training set of trajectory instances are discretized and a normalized histogram is made for each trajectory. A template histogram is then created by taking the mean of all the training instances for each bin.

4.3.1.2 Recognition using Histogram Intersection

Histogram intersection is used to determine the match value for each template trajectory histogram B_j to a given trajectory instance histogram B_k:

$$Intersection : B_j, B_k = \sum_{i=1}^{n} min(B_j(i), B_k(i)) \tag{4}$$

where each histogram has n total bins. A score of 1 means both histograms are an exact match, while a score of 0.5 is a partial match, and a score of 0 is a total mismatch. Trajectory recognition is performed by taking the intersection of a trajectory instance with each template histogram. The given trajectory instance is classified as the template histogram resulting with the highest histogram intersection.

4.3.2 Hidden Markov Model

In this approach to the recognition problem, each trajectory is modeled using a separate Hidden Markov Model (HMM). Each HMM is first trained on example trajectories of a specific behavior. The trained HMM is then given test trajectories to determine the log-likelihood that the test trajectory was generated by that trajectory.

4.3.2.1 Training

The Hidden Markov Model (HMM), as seen in Figure 8d, is composed of hidden states and observations [48]. In Figure 8d the hidden states are labeled with $H_1...H_n$ while the observations are labeled $z_1...z_t$. A random process can be in any one of the hidden states and can emit any one of the observations. In this work the observations consist of the labeled changes in yaw, $\Delta\theta$. The number of hidden states for each HMM are empirically determined. An HMM must learn the transition probabilities between hidden states, $a_{i,j} = P(h_{t+1} = j|h_t = i)$, the probabilities that a hidden state may produce an observation, $b_{j,k} = P(z_t = k|h_t = j)$, and the initial state distribution, $\pi_j = P(h_1 = j)$. The compact notation $\lambda = (A, B, \pi)$ represents the complete parameter set of an HMM. The Baum-Welch algorithm estimates the maximum likelihood of the parameters, λ, when given a corpus of training data, Z, $\bar{\lambda} = max_{\lambda} P(Z|\lambda)$.

4.3.2.2 Testing

The probability of an observation sequence, Z, given an HMM trained on a trajectory, λ,

$$P(Z|\lambda) = \sum_{allH} P(Z|H,\lambda)P(Z|\lambda) \tag{5}$$

is efficiently produced by the forward algorithm [48]. This produces a log-likelihood that a test trajectory instance was produced by the trajectory it was trained upon [48]. A trial consists of an instance of a trajectory being tested against each possible HMM. At each trial, the HMM producing the maximum log-likelihood is determined as the representative trajectory of the trial. If the representative trajectory matches the true test instance label, then it is logged as a positive identification. The accuracy of each trained HMM is the number of positive identifications over the entire corpus of similarly labeled instances.

4.3.3 Conditional Random Field

As seen in Figure 8e, conditional random fields (CRFs) are undirected graphical models and are commonly used for structured classification [31]. CRFs are built from a vector of weights and a vector of features. Features take the form $f_i(t, l_{t-1}, l_t, Z)$ where i is an index into the feature vector f and t is an offset into the sequence, l_{t-1} and l_t are values of the label pair at time $t-1$ and t, respectively. Z represents the entire observation sequence across all values of t. In both the following subsections,

$$\Phi_Z = \sum_L \prod_{t=1}^{T} exp(w^T f(t, l_{t-1}, l_t, Z)) \tag{6}$$

is a normalizing constant.

4.3.3.1 Training

Training of CRFs is performed by finding a weight vector w^* that maximizes the conditional log-likelihood of labeled training data:

$$\mathcal{L}(L|Z;w) = w^T f(t, l_{t-1}, l_t, Z) - log(\Phi_Z) \tag{7}$$

$$w^* = \arg\max_w \mathcal{L}(L|Z;w) \tag{8}$$

4.3.3.2 Testing

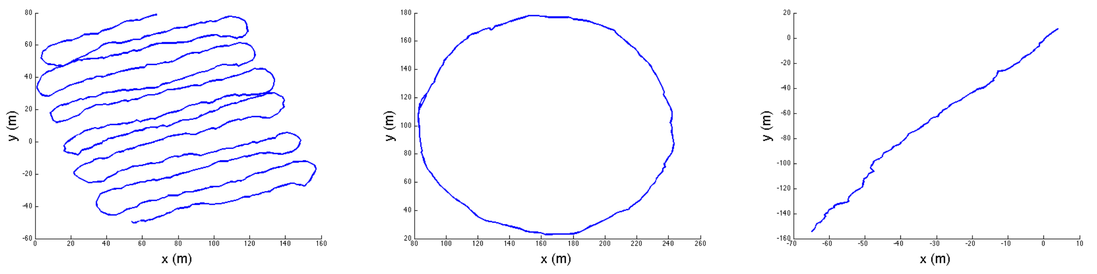
The conditional probability of a label sequence given an observation sequence is computed from the weighted sum of the features as:

$$P(L|Z) = \frac{1}{\Phi_Z} \prod_{t=1}^{T} exp(w^T f(t, l_{t-1}, l_t, Z)) \tag{9}$$

The most likely trajectory label l is assigned to each observation for each test instance presented to the trained CRF.

4.4 Experiments

We performed experiments in two scenarios: a stationary observer and a *Tracking & Trailing* observer that followed the vehicle of interest. The stationary experiments



(a) Noisy *SearchPattern*. (b) Noisy *Loiter*. (c) Noisy *GoToWaypoint*.

Figure 9: Noisy versions of the template trajectories are depicted in (a), (b), and (c).

were first performed using trajectory data gathered through simulation and then using a stationary forward-looking sonar. In order to test our method with two AUVs, trajectory data was gathered in simulation with one *Tracking & Trailing* a leader vehicle. The final set of experiments was performed with the three trajectory recognition techniques receiving data in real time from a simulated mission.

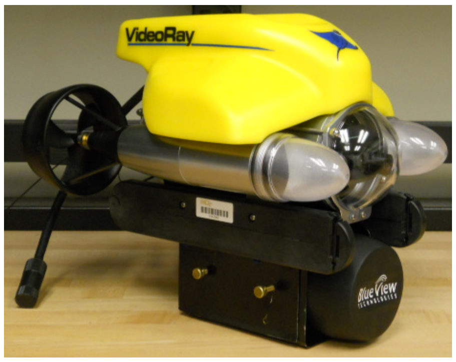
4.4.1 Stationary Observer

4.4.1.1 Simulation

MOOS-IvP, a widely used behavioral architecture for real and simulated AUVs, is used to generate the simulated trajectory data [8]. The trajectories *GoToWaypoint*, *Loiter*, and *SearchPattern* are run within iMarineSim and viewed through pMarineViewer, which are tools included in MOOS-IvP. The locations of the AUVs are recorded as each trajectory is executed, providing a template trajectory. In order to create more realistic results, the template trajectories undergo rotation and translation transformations and are injected with Gaussian noise. Variations of each trajectory template are created as they undergo a random assignment of transformations, including changes in rotation and translation along with an injection of cumulative Gaussian noise with random assignments of standard deviation ranging from 0 to 0.25, as seen in 19a, 19b, and 19c. This will demonstrate that our methods are agnostic to the environment as they are robust to rotations and translations and environmental noise. The global change in yaw for these experiments is discretized into seven bins with a spread of four degrees per bin.

4.4.1.2 Real Sonar Data

For our real sonar data experiments, a surrogate vehicle called the VideoRay ROV is used instead of the Yellowfin AUV due to space limitations in our testing tank, the Acoustic Water Tank facility of the Georgia Tech Woodruff School of Mechanical Engineering. The testing tank is 7.62 meters deep, 7.62 meters wide, and 10.36 meters



(a) VideoRay ROV.

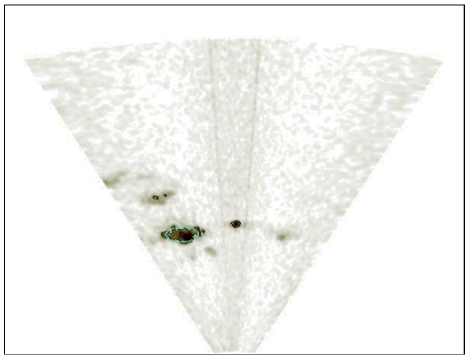
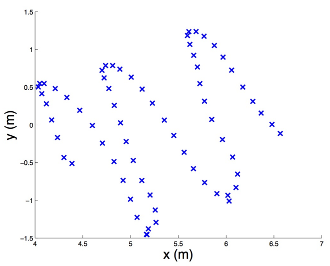
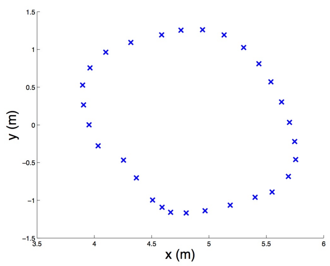

(b) Inverted Sonar Image.

Figure 10: The VideoRay ROV is seen in (a). An inverted sonar image of the VideoRay ROV along with false positive noise is seen in (b). The largest object is assumed to be the target ROV while the smaller objects are noise.

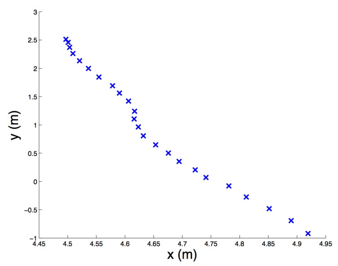
long. The VideoRay is a modified Pro 4 model ROV from Video Ray LLC. which includes the addition of Yellowfin subsystems such as the WHOI acoustic micro-modem and a BlueView forward looking sonar, as seen in Figure 10a. The experiments were conducted with a BlueView forward-looking sonar positioned statically in a corner while it recorded the location of a human piloted VideoRay ROV. Throughout the experiment, the VideoRay ranged between 1 to 10 meters from the BlueView sonar. For these experiments, the perception algorithm makes the simplifying assumptions that there is only one relevant object in the scene, the VideoRay, and that it will always be in the FOV of the sonar. The VideoRay operators were asked to perform multiple runs of three trajectories, *GoToWaypoint*, *Loiter*, and *SearchPattern*.



(a) Sonar *SearchPattern*.



(b) Sonar *Loiter*.



(c) Sonar *GoToWaypoint*.

Figure 11: Real trajectories captured by a BlueView forward-looking sonar of an AUV performing *SearchPattern*, *Loiter*, and *GoToWaypoint* are seen in (a), (b), and (c), respectively.

The BlueView forward-looking sonar provides an image with intensity values corresponding to the acoustic response of a surface, as seen in Figure 10b. The more intense a pixel, the more likely that an object exists at that location. In order to smooth the ROV's trajectory, only every *ith* frame is used, here $i = 10$. This reduces the number of outliers significantly as the sonar data is extremely noisy. Edges are found in the sonar image which are used to create contours. The contour with the largest area is assumed to be the ROV, as we assume that only the ROV is in the image and the smaller contours are noise. The API of the BlueView sonar then produces the range and bearing of the center pixel relative to the sonar itself. Range and bearing are then converted to x and y coordinates to produce trajectories, as seen in Figure 11. In this form, the discretization process converts location to global yaw then to change in yaw as described above. In this experiment, the global change in yaw is discretized into five bins with a spread of four degrees each.

4.4.2 *Track & Trail*

In order to test our methods with a non-stationary observer, testing was performed on simulated data with one AUV performing *Track & Trail* of a leader performing a trajectory, as seen in Figure 6. As in the experiments above, the MOOS-IvP simulator is used to generate template trajectories of a leading AUV performing *GoToWaypoint*, *Loiter*, and *SearchPattern* while an observing AUV performs *Track & Trail*.

In order to use the change in yaw method of encoding, the template trajectories of each vehicle are used to produce the pose (x, y, θ) of the trailing vehicle along with range and bearing to the lead vehicle. Using this information allows the trailing AUV to reconstruct the leading AUV's trajectory which will be discretized for use in trajectory recognition. In order to create more realistic results, the original measurements of the trailing AUV's location (x, y, θ) along with range and bearing to the leader AUV are injected with Gaussian noise, similar to those seen in 19a, 19b, and 19c. This more accurately represents the uncertainty an AUV will have of its own location and the uncertainty of the location of the target AUV present in sonar data. In this experiment, the global change in yaw is discretized into seven bins with a spread of four degrees each.

4.4.3 Real-Time Recognition

In order to test our methods performing recognition in real-time, an example survey mission was simulated in MOOS-IvP, as seen in Figure 12. The mission consisted of an AUV performing a repeated survey of an area. First, the AUV performs the *Loiter* trajectory as it awaits the start mission order. In transit to the survey location, the AUV performs the *GoToWaypoint* trajectory. Once it arrives at the appropriate coordinates, the AUV begins the *SearchPattern* trajectory of the area. At the termination of the *SearchPattern* trajectory, the AUV performs *GoToWaypoint* trajectory to return to the start of the survey coordinates to perform a second *SearchPattern* trajectory.

Each trajectory recognition method had slight modifications for real-time recognition. The CRF produces a histogram of each behavior ID and if a single trajectory was above a threshold of 70% then it was labelled as that trajectory. If no trajectory was above 70% then the CRF system returned *no label*. The HMM method produces

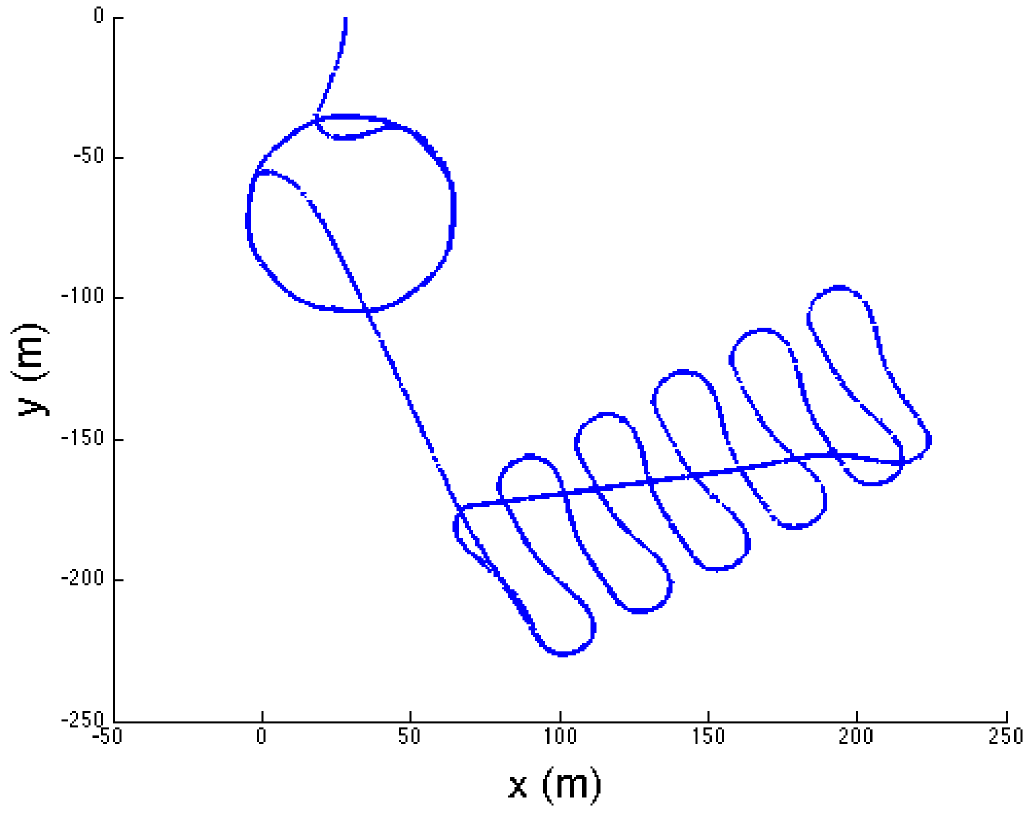


Figure 12: Simulated survey mission in which the AUV loiters, proceeds to a location, and then performs *SearchPattern* of a specific area. The three trajectory recognition techniques label the trajectories in real-time.

a negative log-likelihood that a learned trajectory HMM produced the trajectory found in the trajectory window. The larger the negative log-likelihood, closer to zero, the more likely that trajectory template produced the trajectory seen in the window. The HMM method was sensitive to the number of hidden states used to represent a trajectory. A manual tuning process resulted in all the HMMs having the same number of states. The behavior histogram method produced an intersection value for each trained trajectory. A value of one meant that the trajectory in the window was a perfect match for the behavior histogram and anything less meant it was less like the trained behavior histogram. In the simulated survey mission each trajectory was labelled by hand, including an extra label for transitions between trajectory. The results for the real-time trajectory recognition experiments are the percentage of correctly labeled trajectories versus those incorrectly labeled.

4.5 Results

The results of three different sources of data are analyzed and presented below. The accuracy of the Behavior Histogram, Hidden Markov Model (HMM), and Conditional Random Field (CRF) methods are considered for each data source. Accuracy is defined as the trajectory recognition method correctly identifying the trajectory with its true label.

4.5.1 Stationary Observer

4.5.1.1 Simulation

As seen in Table 1, each method was trained on a specific trajectory using a corpus of 600 instances of trajectories generated by running the behavior offline. A total of 400 trajectories from each template, for a total of 1200 instances, were presented to the three methods for classification. The trajectories were generated by inserting Gaussian noise with a standard deviation of 0.25 on the location. As is seen in Table 1, the Behavior Histogram method was able to achieve an accuracy of 84.25%, 99.25%, and 65% for *SearchPattern*, *Loiter*, and *GoToWaypoint*, respectively. The HMMs performed well for *SearchPattern*, *Loiter*, and *GoToWaypoint* as they were able to accurately discriminate trials by 100%, 99.25%, and 96%, respectively. The CRF had 100% accuracy on all three trajectories. As seen in Table 2, the confusion matrix for the Behavior Histogram shows that 63 instances of *SearchPattern* were confused with *GoToWaypoint*, while *GoToWaypoint* was confused with *SearchPattern* and *Loiter*. For the three trajectories, the HMM method had the most difficulty in discriminating *GoToWaypoint*, recognizing 16 instances of that trajectory as *SearchPattern*.

Table 1: Accuracy of Simulated Stationary Trajectory Recognition.

Trajectory	Training	Testing	Histogram	HMM	CRF
SearchPattern	600	400	84.25%	100.00%	100.00%
Loiter	600	400	92.25%	99.25%	100.00%
GoToWaypoint	600	400	65.00%	96.00%	100.00%

Table 2: Confusion Matrices Simulated Stationary Data.

			Recognized Trajectory		
			SearchPattern	*Loiter*	*GoToWaypoint*
Histogram	Actual Trajectory	*SearchPattern*	337	0	63
		Loiter	31	369	0
		GoToWaypoint	121	19	260
HMM		*SearchPattern*	400	0	0
		Loiter	0	397	3
		GoToWaypoint	16	0	384
CRF		*SearchPattern*	400	0	0
		Loiter	0	400	0
		GoToWaypoint	0	0	400

4.5.1.2 Sonar Data

As seen in Table 3, each method was trained on real sonar data while an ROV performed a specific trajectory using a corpus of 21 instances for *SearchPattern*, 23 instances for *Loiter*, and 14 instances for *GoToWaypoint*. A total of 38 instances were presented to each method for testing. The HMM discrimination method had the best accuracy of 100%, 68.75% and 100%, respectively. The Behavior Histogram method had an accuracy of 75%, 68.75%, and 90%. The CRF performed worse than both the other methods with recognition of *SearchPattern*, *Loiter*, and *GoToWaypoint* with accuracy of 75%, 68.75%, and 80%, respectively. As seen in Table 4, the Behavior Histogram method misidentified one instance of *SearchPattern* with *Loiter* and two as *GoToWaypoint*. Its worst performance was misidentifying four instances of *Loiter* as *SearchPattern* and one instance as *GoToWaypoint*. The HMM method only had errors in recognizing *Loiter* with five instances being identified as *SearchPattern*. The CRF method performed similarly to the HMM method in misinterpreting *Loiter* as *SearchPattern*. Additionally, the CRF method identified one instance of *SearchPattern* as *Loiter* and two instances as *GoToWaypoint*. The CRF method's best performance on the real sonar data was in recognizing *GoToWaypoint* as it only mis-identified two instances as *Loiter*.

Table 3: Accuracy of Sonar Trajectory Recognition.

Trajectory	Training	Testing	Histogram	HMM	CRF
SearchPattern	21	12	75.00%	100.00%	75.00%
Loiter	23	16	68.75%	68.75%	68.75%
GoToWaypoint	14	10	90.00%	100.00%	80.00%

4.5.2 *Track & Trail*

As seen in Table 5, using the change in yaw of the leading vehicle as a discretization method resulted in sufficient accuracy as most of the accuracy was above 90%. The results are from inserting Gaussian noise with a standard deviation of 0.75 on the location (x, y, θ) of the trailing vehicle, range and bearing to the leader. The Behavior Histogram method had an accuracy of 97.5%, 96.25%, and 56.25% of *SearchPattern*,

Table 4: Confusion Matrices Stationary Sonar Data.

			Recognized Trajectory		
			SearchPattern	*Loiter*	*GoToWaypoint*
Histogram	Actual Trajectory	*SearchPattern*	9	1	2
		Loiter	4	11	1
		GoToWaypoint	1	0	9
HMM		*SearchPattern*	12	0	0
		Loiter	5	11	0
		GoToWaypoint	0	0	10
CRF		*SearchPattern*	9	1	2
		Loiter	5	11	0
		GoToWaypoint	0	2	8

Loiter, and *GoToWaypoint*, respectively. The HMM discrimination method had accuracy of 97.25% with *SearchPattern*, 94.75% with *Loiter*, and 95.25% with *GoToWaypoint*. The CRF discrimination method had the best accuracy of discrimination of *SearchPattern*, *Loiter*, and *GoToWaypoint* with accuracy of 99.50%, 99.75%, and 99.75%, respectively. As seen in Table 6, the CRF method only had at the worst case two mis-recognitions of *SearchPattern* versus the HMM method which had a best case of only 11 mis-recognitions and the Behavior Histogram method had 175 mis-recognitions of *GoToWaypoint*.

Table 5: Accuracy of Simulated *Track & Trail* Trajectory Recognition.

Trajectory	Training	Testing	Histogram	HMM	CRF
SearchPattern	600	400	97.50%	97.25%	99.50%
Loiter	600	400	96.25%	94.75%	99.75%
GoToWaypoint	600	400	56.25%	95.25%	99.75%

4.5.3 Real-Time Recognition

As seen in Tables 7, 8, 9, and 10, the three trajectory recognition methods were able to recognize the behaviors in real-time during the simulated survey mission with varying success. The discretization of the change in yaw for each run was performed every two seconds with nine bins each representing eight degrees of change in yaw. Experiments were performed with varying trajectory window sizes. In Table 7, all

Table 6: Confusion Matrices Simulated *Track & Trail* Data.

			Recognized Trajectory		
			SearchPattern	*Loiter*	*GoToWaypoint*
Histogram	Actual Trajectory	*SearchPattern*	390	0	10
		Loiter	8	385	7
		GoToWaypoint	116	59	225
HMM		*SearchPattern*	389	0	11
		Loiter	1	379	20
		GoToWaypoint	13	6	381
CRF		*SearchPattern*	398	2	0
		Loiter	1	399	0
		GoToWaypoint	0	1	399

three trajectory recognition techniques were tested with a trajectory window size of 40 observations. The HMM method fared the worst of the three methods as it was unable to recognize the *GoToWaypoint* behavior while also only recognizing 75.46% of the *Loiter* trajectory. The CRF performed better as it was able to identify 93.14%, 86.79%, and 50.97% of *SearchPattern*, *Loiter*, and *GoToWaypoint* behaviors, respectively. The behavior histogram method fared the best overall as it was above 90.0% accuracy for both the *SearchPattern* and *Loiter* trajectories but only had a 65.05% accuracy for the *GoToWaypoint* trajectory. Increasing the trajectory window size to 55 observations for all three methods decreased their performance across the board, as seen in Table 8.

The HMM method proved to be fragile in relation to the trajectory window size. Several experiments were performed with varying trajectory window sizes for each behavior HMM. As seen in Table 9, by adjusting the trajectory window size of the trajectory presented to the *SearchPattern*, *Loiter*, *GoToWaypoint* HMMs to 70, 70, and 25, respectively, increased the accuracy of the HMM method in recognizing *GoToWaypoint* from 0.0% to 32.91%. The accuracy in determining the *GoToWaypoint* trajectory increased to 49.71% by decreasing the trajectory window size even more to 10 observations, as seen in Table 10. There was a slight decrease in the accuracy of the HMM method in recognizing both *SearchPattern* and *Loiter* in the 70,70,25 and 70,70,10 window size conditions compared to 40 and 55 window sizes which is a reflection of the window size and interplay between the negative log-likelihood produced by each behavior HMM.

Table 7: Percentage Confusion Matrices Real-Time Simulated Survey Mission (Window Size 40, HMM with 7 states).

			Recognized Trajectory			
			SearchPattern	*Loiter*	*GoToWaypoint*	*NoLabel*
Histogram	Actual Trajectory	*SearchPattern*	91.49	0.00	8.51	0.00
		Loiter	8.16	90.31	0.00	1.53
		GoToWaypoint	8.74	26.21	65.05	0.00
		Transition	61.46	0.00	0.00	38.54
HMM		*SearchPattern*	95.82	0.00	0.00	4.18
		Loiter	21.63	75.46	0.00	2.91
		GoToWaypoint	52.43	2.52	0.00	45.05
		Transition	58.33	0.00	0.00	41.67
CRF		*SearchPattern*	93.14	0.36	5.21	1.29
		Loiter	4.80	86.79	0.00	8.42
		GoToWaypoint	14.47	24.85	50.97	9.71
		Transition	53.33	4.90	0.00	41.77

4.6 Discussion

The work presented here demonstrates the ability of the three trajectory recognition systems to work relatively well when static testing data is provided. The Hidden Markov Models (HMM) method resulted in sufficient performance with static data as most of the recognition accuracy was above 90%. Using Behavior Histograms resulted in the worst performance of the three methods in both the simulated stationary data and the simulated *Track & Trail* data while performing slightly better than the CRF method in the sonar data. It is not surprising that the baseline method, Behavior Histogram, is the worst performer when compared to the other two methods as it only takes the frequency of the changes in yaw into account making it less suitable for time series data. Using the Conditional Random Field method resulted in better

Table 8: Percentage Confusion Matrices Real-Time Simulated Survey Mission (Window Size 55, HMM 7 states).

			Recognized Trajectory			
			SearchPattern	*Loiter*	*GoToWaypoint*	*NoLabel*
Histogram	Actual Trajectory	*SearchPattern*	86.08	0.00	13.92	0.00
		Loiter	5.10	85.71	0.00	9.18
		GoToWaypoint	15.53	35.92	48.54	0.00
		Transition	61.46	0.00	0.00	38.54
HMM		*SearchPattern*	94.33	0.00	0.00	5.67
		Loiter	22.55	66.94	0.00	10.51
		GoToWaypoint	42.14	1.75	0.00	56.12
		Transition	55.52	0.00	0.00	44.48
CRF		*SearchPattern*	84.90	0.00	10.88	4.23
		Loiter	4.80	78.47	1.33	15.41
		GoToWaypoint	16.89	24.56	41.94	16.60
		Transition	47.29	0.00	2.92	49.79

Table 9: Percentage Confusion Matrix Real-Time HMM 7 States Variable Window Sizes 70,70,25

		Recognized Trajectory			
		SearchPattern	*Loiter*	*GoToWaypoint*	*NoLabel*
Actual	*SearchPattern*	92.73	0.00	0.08	7.19
	Loiter	20.87	60.97	0.00	18.16
	GoToWaypoint	31.07	0.87	32.91	35.15
	Transition	58.54	0.00	0.00	41.46

performance than the HMM method when there was ample training data available, as in the simulated stationary observer data or the simulated *Track & Trail* data. However, the CRF method performed poorly in discrimination of the real sonar data. Due to the small sample size of real sonar data it may be an indication of under-training the CRF. However, all the methods struggled with discriminating the *Loiter* trajectory in the real sonar data set. It is possible that the sonar-captured *Loiter* trajectory should be further separated into *counter-clockwise* and *clockwise Loiter* as that could be the reason for the methods performing poorly. This is in contrast to the simulated *Loiter* trajectories which only performed them in the *counter-clockwise* direction.

Table 10: Percentage Confusion Matrix Real-Time HMM 7 States Variable Window Sizes 70,70,10

		Recognized Trajectory			
		SearchPattern	*Loiter*	*GoToWaypoint*	*NoLabel*
Actual	*SearchPattern*	90.75	0.00	4.51	4.74
	Loiter	20.87	60.97	0.00	18.16
	GoToWaypoint	28.93	0.87	49.71	20.49
	Transition	56.35	0.00	2.19	41.46

The three trajectory recognition techniques did not fare as well during real-time recognition experiments as an ideal system would have above 90% accuracy for all the trajectories. In general, the behavior histogram method performed the best. This may be due to the lack of noise in the simulated mission trajectory. Both the CRF and HMM methods poorly recognized the *GoToWaypoint* trajectory. When using one trajectory window size for all trajectories, the CRF method performed with more accuracy overall than the HMM method. In general, the HMM method is more difficult to tune because the number of hidden states must be determined in conjunction with the trajectory window size for each trajectory for best performance. The poor overall results for all the trajectory recognition methods may be due to the method in which the simulated mission trajectory is labelled. The real-time data was labelled according to which trajectory the vehicle was running at the time of discretization. This does not correspond to what a human can distinguish by looking at the trajectory window. This will be a focus of our future research.

Discretization parameters for the change in yaw observations play a crucial role in the success of trajectory recognition in these experiments. For example, an experiment that discretizes the global change in yaw with five bins each with a spread of four degrees has a limited resolution. Any change in yaw greater than six or less than negative six degrees is placed into bins one and five, respectively. Thus, if a crucial distinction between two trajectories occurs beyond these terminal edge bins they will not be properly discriminated. The discretization parameters were empirically determined for each experiment. While simple, global changes of yaw by time as an observation method is very susceptible to changes in speed. Alternatives are to use an observation of changes in yaw by distance or describe higher level primitives such as *straight*, *left turn*, and *right turn* as observations for our trajectory recognition methods.

Future work includes further investigation of discretization and trajectory recognition methods along with a larger data set. It may be possible that each trajectory recognition method requires different change in yaw discrimination parameters for improved accuracy. Alternatively, higher level motion primitives may increase accuracy and robustness. Alternative recognition methods may be more appropriate. Handwriting recognition has similarities with trajectory recognition with a long history and may yield improved methods [47]. Recognition should be verified with more trajectories than the ones used in these experiments, as they are a small sample representation. Future research will include adding noise to the simulated real-time missions and a real-time mission with data gathered from actual vehicles.

4.7 Conclusion

This chapter demonstrates the use of Behavior Histograms, HMMs, and CRFs for recognizing common trajectories performed by autonomous marine vehicles. The common trajectories or tasks are *SearchPattern*, *Loiter*, and *GoToWaypoint*, which can be seen through sonar data in Figure 11. The three trajectory recognition methods were compared on simulated trajectories and on actual sonar data. Two observing teammate strategies were explored. The first was a static observing teammate which had an overwatch of its acting teammate. The second strategy is the observing teammate performing *Track&Trail* of the acting teammate. Both strategies are plausible given different sensing and teammate capabilities. While the experiments in this chapter demonstrate the feasibility of performing trajectory recognition of teammates, it only focuses on the parameters and methods of the recognition algorithms. The experiments in which the recognition algorithms have lower than preferred recognition accuracy are what motivate the concept of adapting trajectories for improved recognition performance. The trajectory adaptation for recognition (TAR) framework is introduced in Chapter 6.

CHAPTER V

FOUNDATIONAL WORK: ROBOT COOPERATION THROUGH TRAJECTORY-BASED COMMUNICATION

This chapter explores the use of trajectory-based communication. Similar to the Honeybee's waggle dance, autonomous robots can signal a location of interest through a performed trajectory. Recognition of these location signals can then be leveraged by an observing autonomous teammate for continued mission success even in sporadic communication. This work is drawn from previous publications [41, 38]. This foundational work on leveraging recognition for trajectory-based communication demonstrates the feasibility of such a team strategy. The difficulty though is that the trajectory used for communicating the location of interest is created by a tedious manual process. The cycle consists of manual adaptation of the trajectory, training of the trajectory recognition method, and analysis of the results. The first issue with the newly adapted trajectory is that the autonomous vehicle could not track the desired waypoints due to either the vehicle dynamics or waypoint tracking parameters. The second issue is that the adapted trajectory would not achieve the desired recognition accuracy. These issues motivated the autonomous adaptation of trajectories for improved recognition using the TAR framework which is described in detail in Chapter 6.

5.1 Introduction

This chapter focuses on enabling multi-robot teams to cooperatively perform tasks without the use of radio or acoustic communication. One key to more effective cooperative interaction in a multi-robot team is the ability to understand the task and intent of other robots. This is similar to the honey bee “waggle dance” in which a bee can communicate the orientation and distance of a food source, seen in Figure 17b. In this similar manner, our heterogenous multi-robot team uses a specific trajectory to indicate the location of mine-like objects (MLOs). Observed teammate action sequences can be learned to perform trajectory recognition and task-assignment in the absence of communication. We apply Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) to perform trajectory recognition as an approach to task monitoring in the absence of communication in a challenging underwater environment. In order to demonstrate the use of trajectory recognition of an Autonomous Underwater Vehicle (AUV) in a cooperative task, we use trajectory-based techniques for model generation and trajectory discrimination in experiments using simulated scenario data. Results are presented demonstrating heterogenous teammate cooperation between an AUV and an Autonomous Surface Vehicle (ASV) using trajectory recognition rather than radio or acoustic communication in a mine clearing task.

The purpose of this research is to enable robot teams to cooperate in environments without communications. Many current decentralized coordination methods, such as auctions or self-assignment, require teammates to broadcast their self-assigned task/roles along with costs. By defining a task/role as a robot performing a trajectory in a certain location, trajectory recognition can be used as task/role identification. In the previous chapter, we have demonstrated the ability to perform trajectory recognition of a limited number of static trajectories using simulation and real sonar data. The research presented in this chapter extends our previous work by focusing on a mine clearing task which includes teammate trajectory recognition so that implicit communication can be leveraged. Similarly to honey bees performing a “waggle dance” to indicate the direction and distance to a food source, our AUV will perform it’s own dance trajectory called the *InfinityPattern* to indicate the location of

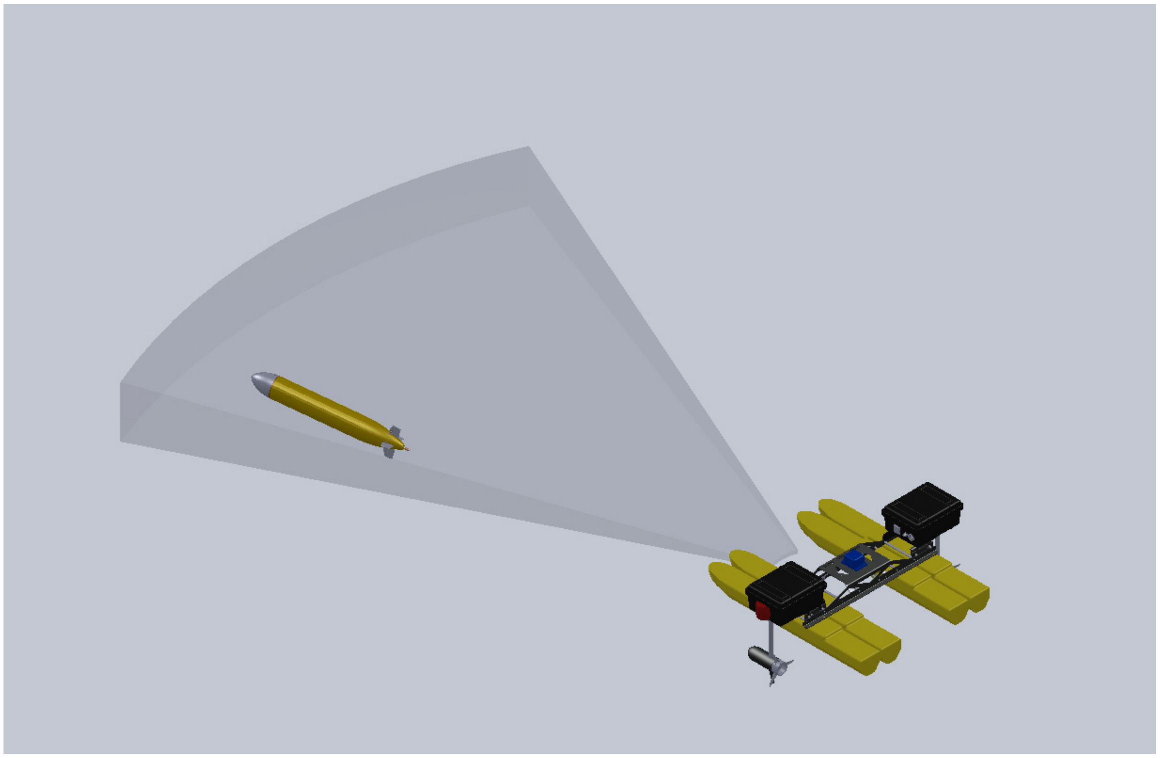


Figure 13: A Kingfisher ASV using its sonar to observe a Yellowfin AUV. The ASV can use these observations to perform trajectory recognition of the AUV.

a mine-like object (MLO). Instead of traditional communications, the AUV uses intended recognition [28] with the *InfinityPattern* trajectory, shown in Figure 18a, to communicate the location of an MLO.

The goal of this work is to create a system that can efficiently operate with as little explicit communication as possible. This chapter investigates the feasibility of an ASV performing trajectory recognition of an AUV through a sonar, as illustrated in Figure 13. Although [6] has used simulated sonar data and [5] has used post-mission GPS trajectory analysis of an actual AUV for trajectory verification, little or no research has attempted to perform cooperative behaviors based on trajectory recognition in the underwater domain. Our method is presented using a simulation of an Autonomous Surface Vehicle (ASV) performing cooperative trajectories by using trajectory recognition, with an HMM and CRF, of an Autonomous Underwater Vehicle (AUV) in a mine-clearing task.

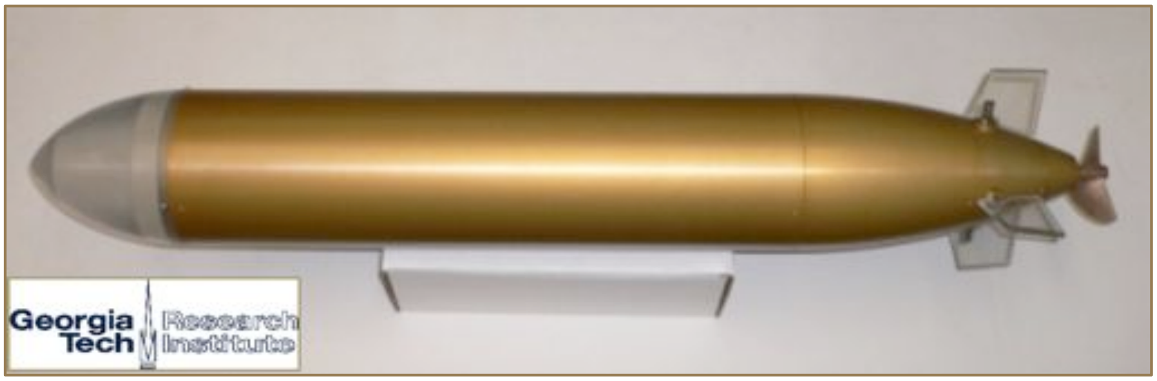


Figure 14: Yellowfin Autonomous Underwater Vehicle - designed to be man-portable for oceanographic observation.

5.2 Hardware Platform

The motivation for this work is the need for multiple small AUVs and ASVs to perform autonomous research operations in underwater environments. For example, the Georgia Tech Research Institute (GTRI) has developed the Yellowfin AUV research platform, as seen in Figure 14, and because of the vehicle's size, power constraints, and operating environment, communication bandwidth is limited.

The heterogeneous teammate of the Yellowfin AUV is the Kingfisher M100 ASV made by Clearpath Robotics. The Kingfisher, as seen in Figure 17a, is designed for environmental and civil engineers to be quickly deployable. It weighs 30 kg and its dimensions are 1.27x1.27x0.52 meters. While they are not the fastest vehicles, with a max speed of two meters a second, they do provide a stable platform capable of keeping a station over a specified location. In addition, their size does allow for a 11.5 lbs payload which is used by more capable sensors. In these experiments a Garmin GPS module is used for localization and a Bullet WiFi system is used for communications.

The autonomy software leverages the open-source MOOS-IvP software suite [8]. MOOS-IvP allows for rapid deployment of autonomous vehicles. The MOOS portion, also called the MOOSDB, is a centralized database scheme for message passing between separate modules or programs. The IvP portion contains all the modules needed for autonomy. These IvP modules includes the IvP-Helm which is responsible for coordinating active behaviors. Other modules in the IvP tree include modules for interfacing with the Kingfisher hardware such as the GPS, compass, and motors.

The base station for our experiments is a shoreside computer running the MOOS-IvP module pMarineViewer [7]. The pMarineViewer displays a geo-referenced image along with the locations of the autonomous vehicles. From the GUI, messages can be sent to the IvP module of each vessel.

5.3 Methods

5.3.1 Trajectory Discretization

The encoding method used is agnostic to any environment. The only measurement required is the location $x = (x, y)$ coordinates of an AUV in a fixed 2D plane, as seen in Figure 8a. The motion model of the vehicle is assumed to be non-holonomic and always moving with a forward motion similar to a tricycle model. The yaw of the vehicle is calculated from the vector of motion from one time-step to the next.

$$\Delta x_{(t-1,t)} = x_t - x_{t-1} \tag{10}$$

$$\theta_t = \arctan(\Delta x_{(t-1,t)}) \tag{11}$$

$$\Delta\theta_t = \theta_t - \theta_{t-1} \tag{12}$$

The encoding used in this research is the change in yaw between measurements. Possible changes in yaw are discretized according to bins. Each bin corresponds to a range of values. Bin 3 in our example represents a change in yaw between -0.12 and +0.12 rad. For instance, as seen in Figure 8b, an AUV moving straight ahead is observed as having a 0 rad change in yaw and thus encoded as a *3* while one turning by .26 rad is encoded as a *2*. A series of these encodings are combined into a trajectory string for input into the HMM or CRF.

5.3.2 Recognition Methods

In this work we compare the performance of a Hidden Markov Model (HMM) method and a Conditional Random Field (CRFs).

5.3.3 Hidden Markov Model

In this approach to the recognition problem, each trajectory is modeled using a separate Hidden Markov Model (HMM), seen in Figure 8d. Each HMM is first trained on example trajectories of a specific trajectory template. The trained HMM is then given test trajectories to determine the log-likelihood that the test trajectory was generated by that template trajectory. The Hidden Markov Model (HMM) is composed of hidden states and observations [48]. In Figure 8d the hidden states are labeled with $H_1 ... H_n$ while the observations are labeled $z_1 ... z_t$. A random process can be in any one of the hidden states and can emit any one of the observations. In this work the observations consist of the labeled changes in yaw, $\Delta\theta$. The number of hidden states for each HMM are empirically determined. An HMM must learn the transition probabilities between hidden states, $a_{i,j} = P(h_{t+1} = j | h_t = i)$, the probabilities that a hidden state may produce an observation, $b_{j,k} = P(z_t = k | h_t = j)$, and the initial state distribution, $\pi_j = P(h_1 = j)$. The compact notation $\lambda = (A, B, \pi)$ represents the complete parameter set of an HMM. The Baum-Welch algorithm estimates the maximum likelihood of the parameters, λ, when given a corpus of training data, Z, $\bar{\lambda} = max_\lambda P(Z|\lambda)$. The probability of an observation sequence, Z, given an HMM trained on a behavior, λ,

$$P(Z|\lambda) = \sum_{allH} P(Z|H, \lambda) P(Z|\lambda) \tag{13}$$

is efficiently produced by the forward algorithm [48]. This produces a log-likelihood that a test trajectory instance was produced by the trajectory template it was trained upon [48]. A trial consists of an instance of an instance trajectory being tested against each possible HMM. At each trial, the HMM producing the maximum log-likelihood is determined as the representative trajectory template of the trial. If the representative trajectory matches the true test instance label, then it is logged as a positive identification. The accuracy of each trained HMM is the number of positive identifications over the entire corpus of similarly labeled instances.

5.3.4 Conditional Random Field

Conditional random fields (CRFs) are undirected graphical models for structured classification [31], seen in Figure 8e. CRFs are built from a vector of weights, w, and a vector of features, f. Features take the form $f_i(t, x_{t-1}, x_t, Y)$ where i is an index into the feature vector f and t is an offset into the sequence, x_{t-1} and x_t are values of the label pair at time $t-1$ and t respectively. X represents all the output variables for a sequence, in this work the trajectory labels. Y represents the entire observation sequence across all values of t, which in this work are the sequence of changes in yaw. T is the termination time index for a given sequence. Z_Y is a normalizing constant required to have an actual probability distribution:

$$Z_Y = \sum_{X'} \prod_{t=1}^{T} exp(w^T f(t, x'_{t-1}, x'_t, Y)) \tag{14}$$

Training of CRFs is performed by finding a weight vector w^* that maximizes the conditional log-likelihood of labeled training data:

$$l(X|Y;w) = w^T f(t, x_{t-1}, x_t, Y) - log(Z_Y) \tag{15}$$

$$w^* = \arg\max_{w} l(X|Y;w) \tag{16}$$

The conditional probability of a label sequence given an observation sequence is computed from the weighted sum of the features as:

$$P(X|Y) = \frac{1}{Z_Y} \prod_{t=1}^{T} exp(w^T f(t, x_{t-1}, x_t, Y)) \tag{17}$$

The most likely trajectory label x is assigned to each observation in a test sequence presented to the trained CRF.

5.3.5 Real-time Trajectory Recognition

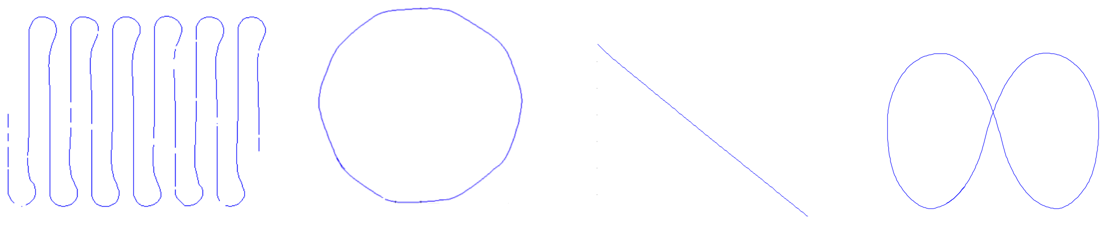
Real-time trajectory recognition, as was used in [41], is an extension of instance testing. As a vehicle performs a task, only a limited portion of the trajectory is presented to the trajectory recognition methods, called the trajectory window. Each method produces a trajectory label, the label with the largest conditional probability, for each observation in the trajectory window. The trajectory label at the head of the trajectory window is logged as the trajectory which has the greatest percentage representation in the trajectory window, as long as it is above a threshold. Starting at $t = windowsize$ the entire trajectory label is logged as the trajectory window frame moves along it.

5.3.6 Trajectory Label Smoothing

The real-time trajectory recognition described above and used in our previous work [41] resulted in poor trajectory recognition accuracy on the actual ASV data and thus needed to be extended. Instead of only recording the label of the most represented trajectory in the trajectory window at position t, each time the trajectory position t is in the trajectory window and receives a trajectory label from the recognition method it was saved. This results in the position t in the trajectory to receive a total of $windowsize$ labels from the recognition method. The trajectory label logged for position t is the label with the highest frequency. This essentially allows the observation to have votes counted toward its trajectory label each time the trajectory recognition method produced a label.

5.3.7 MLO Location Estimation

The *InfinityPattern* was created to convey 1) the location of an object of interest and 2) be distinguishable from the other normal trajectories performed by maritime vehicles during a survey mission. Ideally, as the ASV performs the *InfinityPattern*, the intersection of the pattern is directly over the MLO. A teammate estimates the



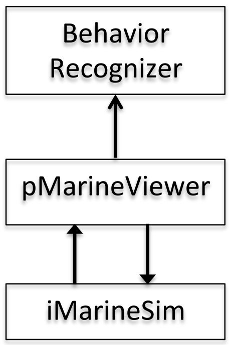
(a) *SearchPattern.* (b) *Loiter* (c) *GoToWaypoint* (d) *InfinityPattern*

Figure 15: Simulated trajectories of an AUV performing *SearchPattern*, *Loiter*, *GoToWaypoint*, and *InfinityPattern* are seen in (a), (b), (c), and (d), respectively.

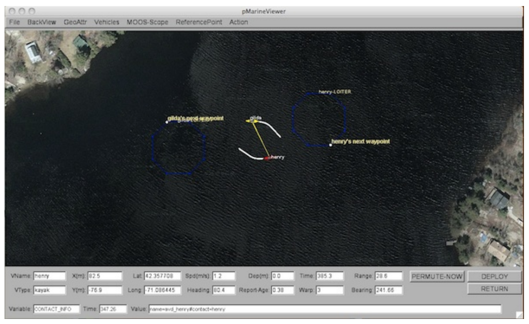
location of an MLO by calculating the center of a segment labelled as *InfinityPattern*.

5.4 Experiment I: Simulation

The simulation environment for our experiments is provided by iMarineSim and pMarineViewer which are part of the MOOS-IvP open source autonomy package [8]. The pMarineViewer module is a GUI-based tool, as seen in Figure 16b, that renders 2D overhead maps of the vehicles performing behaviors. The iMarineSim is a single-vehicle simulator that updates vehicle state based on actuator values. Actuator values are produced by the IvP Helm, which is a coordinator over activated behaviors. The simulator tools allow for verification of vehicle behaviors and interactions. The experiments are performed using trajectory data gathered through simulation. The only behavior employed is the go to waypoint behavior which allows



(a) Simulation workflow.



(b) An example mission running in pMarineViewer.

Figure 16: Simulation Experiment Setup

for the specification of trajectories using multiple waypoints. The trajectories *GoToWaypoint*, *Loiter*, *SearchPattern*, and *InfinityPattern* are run within iMarineSim and viewed through pMarineViewer which shares its plotted trajectories with our trajectory recognition module, as seen in Figure 16a. The locations of the AUVs are recorded as each trajectory is performed. For these experiments, the perception algorithm makes the simplifying assumptions that there is only one relevant object in the scene, the Yellowfin, and that it will always be in the FOV of the sonar.

In this mine-clearing scenario, our heterogenous team consists of a Yellowfin AUV and our Kingfisher ASV, as described above. While the Yellowfin quickly performs a *SearchPattern* behavior over the designated area of operation, the Kingfisher watches silently. The Yellowfin AUV is quick and therefore can cover a large area much faster than the Kingfisher ASV. However, due to Yellowfin's dynamics it must be continuously in motion lest it sink to the bottom. As it moves quickly through the water column, it uses its BlueView forward-looking sonar to detect mine-like objects (MLOs). Because acoustic communications are restricted, the Yellowfin communicates the potential discovery of an (MLO) by performing an *InfinityPattern*, where the center of the infinity marks the spot of the MLO. Once the Kingfisher observes the *InfinityPattern* it quickly calculates the center of the trajectory and proceeds to investigate whether the MLO is truly a mine to be cleared. Ten trials are performed, results are seen in Table 11, of an MLO in a different location in a 40 meter by 40 meter area. In each of the trials, elapsed time is measured from a Yellowfin AUV detecting an MLO to the time when a Kingfisher ASV arrives at it's location.

5.4.1 Results

Training

Training of the CRF was performed with 600 static trajectories of each template trajectory: *SearchPattern*, *Loiter*, *GoToWaypoint*, and *InfinityPattern*, as seen in

Figure 15. After training, the CRF was verified against static trajectories of each template trajectory. The CRF was able to distinguish each trajectory with 100% accuracy.

Testing

As seen in Table 12, it took the Kingfisher ASV an average of 42.5 seconds to reach the MLO after being detected by the Yellowfin AUV when using acoustic communication. This includes the time for the Yellowfin to communicate the MLO's location and travel time of the Kingfisher ASV to that location. In the no acoustic communication with trajectory recognition scenario, it took the ASV an average of 201.5 seconds to identify the *InfinityPattern* trajectory and travel to the MLO's location.

The time between the Yellowfin AUV detecting an MLO and the Kingfisher recognizing the *InfinityPattern* varied. This variation occurred due to the location of the MLO with respect to the *SearchPattern*. If the *SearchPattern* smoothly transitioned into the *InfinityPattern* then recognition was quick. If the transition required a turn or a loop before initiating the *InfinityPattern* then recognition took longer. As examples, the quickest recognition of the *InfinityPattern* took 127 seconds while the longest recognition time took 238 seconds. In general, the *InfinityPattern* was recognized on its first iteration. However, in one trial it took two full iterations of

Table 11: Arrival time of the Kingfisher ASV.

Trial	w/ Comms	w/ Beh. Rec.
1	32	158
2	34	203
3	40	195
4	43	167
5	45	177
6	51	207
7	55	219
8	44	279
9	46	205
10	35	205

the *InfinityPattern* before it was recognized. This is all due to the fact that there was varying amounts of confusion while the Yellowfin AUV was transitioning to the *InfinityPattern*. If the transition between *SearchPattern* and *InfinityPattern* occurred during a long leg of the *SearchPattern* then confusion would result with the leg being identified as *GoToWaypoint*. If a small loop was required to transition between *SearchPattern* and *InfinityPattern* then there would be confusion with *Loiter* or even *SearchPattern*. However, these confusions would resolve themselves as more of the *InfinityPattern* would appear. If the transition between trajectories naturally looked like it could have come from an *InfinityPattern* then recognition was quick. On the other hand, if the transition was in stark contrast then it would require a second iteration of the *InfinityPattern* so that only that pattern was observable to the algorithm. If the observation window was too small the *InfinityPattern* was mistakenly identified as the *Loiter* trajectory as only a portion of the infinity is visible. At certain points during the *InfinityPattern* the trajectory recognition algorithm also detected portions of *GoToWaypoint*.

5.4.2 Simulation Conclusion

The work presented in this section demonstrates the feasibility of performing heterogenous cooperation without explicit communication through trajectory recognition. In this implementation, the trajectory of importance being recognized was the *InfinityPattern* which was used to indicate the location of a mine-like object (MLO) which is comparable to the honey bee's "waggle dance" to indicate food. While it is much faster for the heterogeneous team in these experiments to communicate an MLO's

Table 12: Average arrival time of the Kingfisher ASV.

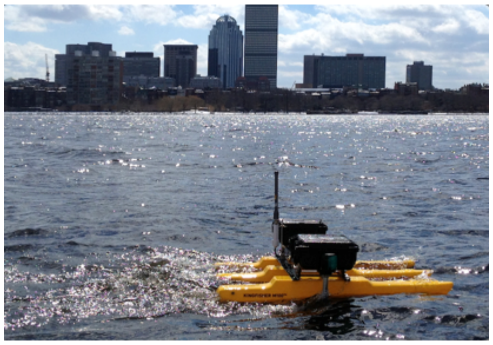
	w/ Comms	w/ Traj. Rec.
Average	42.5	201.5

location acoustically, that may not be feasible in a communication restricted environment. The most restrictive portion of the no communication/trajectory recognition arrival time was the length of time required to recognize the *InfinityPattern* as in each trial the trip time for the ASV was the same. This indicates that the *InfinityPattern* may not be the best trajectory to indicate the location of an MLO. Future work includes further investigation of more optimal parameters for both discretization and for the trajectory CRF as accuracy can be improved. This may include the use of more features than just the change in yaw of the Yellowfin AUV. Ultimately, these methods should be verified with more trajectories than the ones used in these experiments, as they are a small sample representation. In order to verify that this method scales, experiments in simulation will include a larger number of ASVs and AUVs. We are currently gathering real GPS trajectories produced by the Kingfisher ASV performing the above mentioned behaviors so that the presented technique can be tested on real-world data.

5.5 Experiment II: Real ASV Data

In this experiment we demonstrate the approach using real autonomous surface vehicles (ASVs) operating in a marine environment. The mission performed by the ASVs is more realistic and therefore more complex than experiments in the previous section. We compare Hidden Markov Models (HMMs) and a Conditional Random Field (CRF) approach with trajectory-based features for model generation and trajectory discrimination. Results from experiments using ASVs on the Charles River demonstrate the feasibility of the approaches.

In this experiment, we extend our previous trajectory recognition work [41] which is reported in the previous section using trajectory-based communication in a mine clearing task. In normal operations, an AUV would communicate via radio or acoustic modem to its teammates the location of an MLO so that they may safely verify



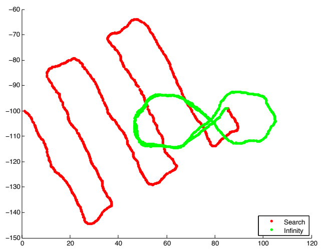
(a) Kingfisher M100

(b) Honeybee Waggle Dance

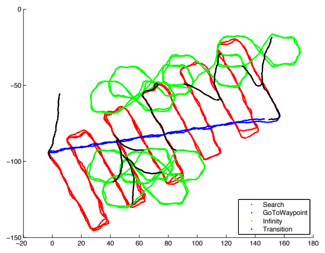
Figure 17: Clearpath Robotics' Kingfisher M100 ASV, seen in (a), is designed for environmental and civil engineers and is easily deployable. In (b) the honeybee can be seen performing its "waggle dance."

and dispose if necessary. Instead of traditional communications, the AUV uses intended recognition [28] with the *InfinityPattern* behavior, shown in Figure 18a, to communicate the location of an MLO. Here we extend our scenario to actual ASV data gathered through self-reported GPS localization and a more complicated mission with seven MLOs, seen in Figure 18b. The repeated trajectory transitions between *SearchPattern* and *InfinityPattern* created poor trajectory recognition accuracy in our original algorithm from [41]. Our key contributions of this research are that we:

- demonstrate our methods on real data gathered by autonomous robots instead of synthetic data

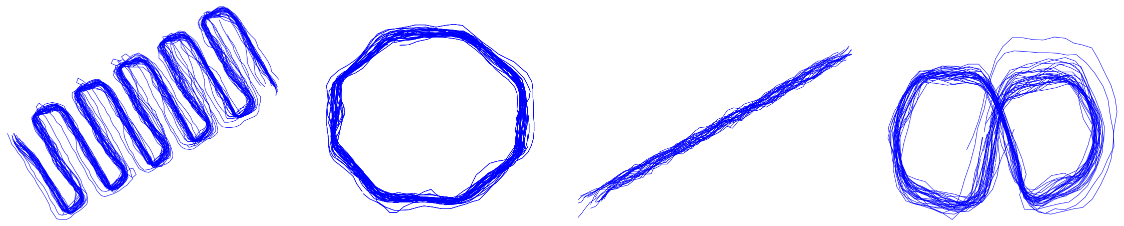


(a) Basic Mission.



(b) Complete Mission.

Figure 18: The ASV performing *SearchPattern*, *GoToWaypoint*, *InfinityPattern*, and *inTransition*, have been hand-labelled, for ground truth, and are colored red, blue, green, and black, respectively. In (a), is an example of the ASV searching and finding an MLO. In (b), the entire mission can be seen.



(a) *SearchPattern*. (b) *Loiter*. (c) *GoToWaypoint*. (d) *InfinityPattern*.

Figure 19: Overlaid GPS trajectories, not to scale, of the Kingfisher ASVs performing *SearchPattern*, *Loiter*, *GoToWaypoint*, and *InfinityPattern* are seen in (a), (b), (c), and (d), respectively.

- extend our previous trajectory recognition algorithm to apply to realistic, complex and dynamic missions

The current experiment uses GPS data as a surrogate for other sensor modalities, since perception is not the focus, but it is feasible to use other sensor modalities such as Airborne LIDAR [25, 32, 63] or hydrophone arrays [15]. A teammate capable of interfacing with either of these sensors can perform trajectory recognition from a remote location.

Data was gathered by deploying the Kingfisher ASV on the Charles River off of the MIT Sailing Pavilion[1]. A base station consisting of an onshore computer running MOOS-IvP's pMarineViewer [7] module connected to the Kingfisher ASV via a Bullet WiFi system. The base station has a drop-down menu which sends appropriate behavior commands to the IvP-Helm module running on the ASV. For all of the experiments, the trajectory of each Kingfisher ASV is recorded via a Garmin GPS18X-5hz with a typical accuracy of 3 meters. The training trajectories of *SearchPattern*, *Loiter*, *GoToWaypoint*, and *InfinityPattern* are seen in Figure 19. To gather data for our mock mine-clearing scenario, the base station was modified so that a user could simulate the detection of a mine-like object (MLO) by triggering an *InfinityPattern* centered at the location of a mouse-click on the pMarineViewer's map. Once the

[1]courtesy of the Center for Ocean Engineering at MIT

Table 13: Confusion matrix for the smoothed HMM trajectory recognition method.

		NoLabel	*SearchPattern*	*Loiter*	*GoToWaypoint*	*Infinity*
Actual	*SearchPattern*	0%	**81.00%**	0%	0.70%	18.30%
	GoToWaypoint	0%	3.90%	0%	96.00%	0%
	Infinity	0%	13.00%	0%	0%	**87.00%**
	InTransition	0%	53.00%	0%	0.87%	46.00%

ASV had performed the *InfinityPattern* twice, it would return to its *SearchPattern* trajectory at the location where it had diverged to perform the *InfinityPattern*.

To test the capability of our methods on a more complicated mission than in [41] and reported in the previous section, two and a half complete tracks of *SearchPattern* were collected in which a total of seven *InfinityPatterns* were performed. As seen in Figure 18b, the trajectory was hand-labeled, to be used as ground truth, into the four trajectories and a fifth label *inTransition* was used when the ASV was interrupting a trajectory to start another. Figure 18a illustrates the hand labelled trajectory of the ASV performing *SearchPattern* and transitioning to *InfinityPattern*. The final mine-clearing trajectory data was normalized to account for the fact that the training data was collected from vessels traveling at a speed of 1.7 m/s while the mock mine-clearing mission data was collected from a vessel traveling at a speed of 1.4 m/s.

5.5.1 Results

The results presented here are obtained using a trajectory with each observation performed every two seconds of runtime. Our original algorithms from [41] and reported in the previous section had poor trajectory recognition results with the real ASV data, described above, and thus needed to be extended with trajectory label smoothing.

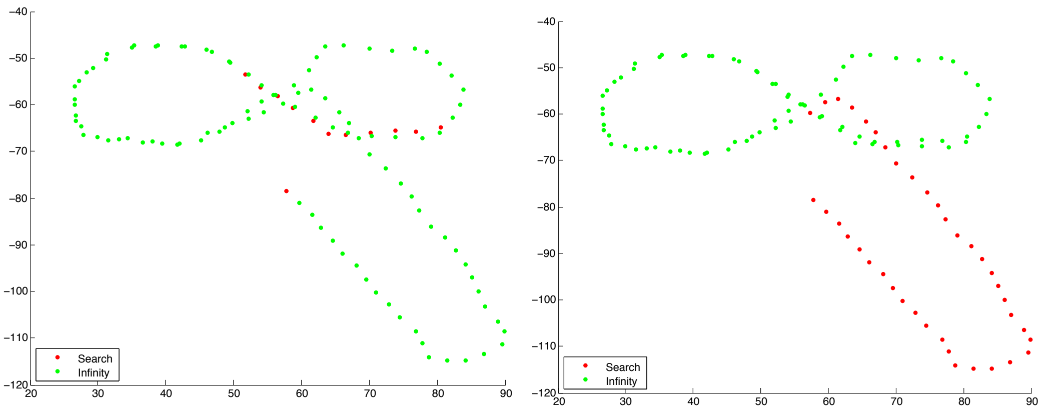
Hidden Markov Model Labels The HMM method is sensitive to the number of states used for each behavior. In the results reported here each trajectory HMM had two states. Other HMMs with more states had better labeling accuracy yet they had more false positives in MLO locations. The HMM method is also sensitive to the trajectory window size for each behavior HMM. The results presented below have trajectory window sizes of 35, 60, 30, and 30 for *SearchPattern*, *Loiter*, *GoToWaypoint*, and *InfinityPattern*, respectively. As seen in Table 13, the HMM method had a trajectory recognition accuracy of 81% for *SearchPattern* which was confused with *InfinityPattern* for 18.30%. *GoToWaypoint* had the highest accuracy with 96% and a small amount of confusion with *SearchPattern* of 3.9%. The *InfinityPattern*, which is of utmost importance in this work, had a recognition accuracy of 87% while being confused with *SearchPattern* for 13%.

Conditional Random Fields Labels The CRF results here are obtained with a trajectory window size of 50 observations, a number empirically determined but large enough to encompass an entire *InfinityPattern*, which represents 100 seconds of operation time. The confusion matrix for the smoothed behavior labels is seen in Table 14. The accuracy of recognizing *SearchPattern* is 96.13%. *GoToWaypoint* a low recognition accuracy of 41.17% while being mis-identified as *SearchPattern* for 58.82%. The *InTransition* label is identified, in decreasing order, as *SearchPattern*, *GoToWaypoint*, and *Infinity*. The accuracy of *InfinityPattern* is important to this work since each segment labelled *InfinityPattern* indicates the location of an MLO. The CRFs recognition accuracy of *InfinityPattern* is 89.07% and confused with *SearchPattern* for only 10.92%. As seen in Figure 20b, the CRF produces much better trajectory labeling results with *SearchPattern* colored correctly in red and *InfinityPattern* correctly colored green.

Estimated MLO Location The proposed system would have teammates respond to every segment that is identified as an *InfinityPattern*. Of the seven actually

Table 14: Confusion matrix for the smoothed CRF trajectory recognition method.

		NoLabel	*SearchPattern*	*Loiter*	*GoToWaypoint*	*Infinity*
Actual	*SearchPattern*	0%	**96.13%**	0%	0.70%	3.16%
	GoToWaypoint	0%	58.82%	0%	41.17%	0%
	Infinity	0%	10.92%	0%	0%	**89.07%**
	InTransition	0 %	40.86%	0%	29.56%	29.56%



(a) HMM Labels. (b) CRF Labels.

Figure 20: The trajectory labelled *SearchPattern* and *InfinityPattern* are colored red and green, respectively. The HMM labels, seen in (a), demonstrate errors in labeling as a segment of *SearchPattern* is green and a segment of *InfinityPattern* is red. The CRF labels, seen in (b), show improvement, where *SearchPattern* and *InfinityPattern* are labelled properly.

performed by the ASV they were all properly labelled. However, both methods only properly segmented the first five *InfinityPatterns*. The sixth and seventh *InfinityPattern* trajectories were correctly labeled as *InfinityPattern* but the two were performed within close proximity of each other. This proximity did not allow for the transitioning segment to be labelled to something other than *InfinityPattern*. This resulted in the two *InfinityPatterns* labelled as one segment and the estimated MLO location was in the middle of the two MLOs. The HMM method would incorrectly label a segment of *SearchPattern* as *InfinityPattern* just prior to an actual performance of *InfinityPattern*, as seen in 20a. A teammate visiting an MLO's location based on these errors would go to a location shifted by the error of labeling a portion of *SearchPattern* as *InfinityPattern*. The CRF method, seen in 20b, more consistently labelled *InfinityPattern* for better MLO location estimation.

5.5.2 Real ASV Data Conclusion

The goal of this work was to use trajectories as a method of communication among autonomous cooperative robots. Using alternative methods of communication beyond radio or acoustics is applicable where communications are denied by active or natural interference. In our presented mock mine clearing mission an ASV searches an AO for MLOs. When the ASV spots an MLO it would normally signal a teammate more suited to deal with the MLO through acoustic or radio communications. In our work, the ASV instead uses intended recognition by performing a maneuver so that a teammate can cooperate by observation. The teammate, using trajectory recognition, can distinguish the intended maneuver and extract information from it. This is similar to that of the honey bee “waggle dance” which indicates the location of a food source to other bees. In this context, the corresponding dance is the *InfinityPattern* which indicates the potential location of a mine. We presented how our algorithms were extended for use with an actual ASV and a much more complicated and realistic mission. The results demonstrate the feasibility of using a trajectory recognition system with either an HMM or CRF for communication with maritime vehicle data. The CRF had higher recognition accuracy for *SearchPattern* and *InfinityPattern* but the HMM had a higher recognition accuracy for *GoToWaypoint*. The estimated MLO locations provided by the HMM labelled segments for *InfinityPattern* were offset by a false labeling of trajectories prior as *InfinityBehavior*. The improved CRF trajectory labeling increased the accuracy of the estimated MLO location based on *InfinityPattern* labelled segments. Both recognition methods failed to segment two *InfinityPatterns* which were performed too close to each other and both of them were labelled as one continuous segment with a teammate then estimating the location of one MLO in-between the two MLOs. The major conclusion is that these preliminary tests show promise for this approach, but more work must be performed to achieve robust performance. Using cooperation by observation and intended recognition allows at least one robot to actively use vehicle motion to communicate information to an observing teammate. This is useful if other communication modalities are temporarily or permanently unavailable. Future work includes experimenting on how this trajectory-based communications algorithm scales to the number of observers or "dancers" along with the size of the AO.

5.6 Conclusion

This chapter demonstrates the use of trajectory-based communication on both simulated and real ASV data. Similar to the Honeybees waggle dance, autonomous robots can signal a location of interest through a performed trajectory. Recognition of these location signals can then be leveraged by an observing autonomous teammate for continued mission success even in sporadic communication. This foundational work on leveraging recognition for trajectory-based communication highlighted issues with manually creating a trajectory. The trajectories manually created were prone to tracking error as the vehicle might not be able to follow waypoints without looping. This was due to either the vehicle dynamics or waypoint tracking parameters. The second issue is that the adapted trajectory would not achieve the desired recognition accuracy. As described in this chapter, the *InfinityPattern* would be mis-identified based on the acting vehicle's entry vector to initiate the trajectory. Lower than desired recognition accuracy and the difficulty in manually adjusting trajectories to also cope with vehicle dynamics lead to the concept of automatically adjusting trajectories and the introduction of the TAR framework which is described in detail in Chapter 6.

CHAPTER VI

TRAJECTORY ADAPTATION FOR TEAMMATE RECOGNITION

Autonomous robots should be able to recognize the task being performed by their autonomous robot teammates. Such task recognition capabilities would assist in task verification, which might be needed if a teammate is reporting one task but perceived to be doing another whether due to error or due to deliberate misrepresentation. Additionally, this capability would help when communications are sporadic or not available. Typically, recognition of robot trajectories or tasks requires the tuning of parameters to get acceptable accuracy by the observer. Experiments in Chapter 4 demonstrated that although different methods are employed and parameters tuned, recognition accuracy can still be below acceptable rates. Experience in manually creating trajectories for trajectory-based signaling in Chapter 5 proved tedious and time-consuming as the newly adapted trajectory waypoints were difficult for the autonomous vehicle to follow or still had below acceptable trajectory recognition accuracy. In this chapter, we consider the ability to automatically modify the trajectory of the actor. We introduce a novel negotiation framework named TAR: Trajectory Adaptation for Recognition. The trajectories that an actor performs during a task is negotiated between the observer and actor which allows for the consideration of the acting vehicle's dynamics, task requirements, sensor properties and recognition algorithms of the observer. We present results of one example instantiation of our TAR framework. Some of this work draws from previously published material [36, 37].

6.1 Introduction

In some circumstances, autonomous vehicles should be able to recognize the task of another autonomous teammate simply through observation. This would be useful if a teammate needs to verify another member's current assigned task. This allows for the correction of an erroneous teammate or one that has nefarious purposes. Another scenario where task recognition can be useful is when RF communications are intermittent or unavailable. This is the type of environment in which marine robots find themselves as both radio frequency (RF) and acoustic communications can be difficult. Multi-robot teams rely heavily on stable communication so that efficiency is achieved by not repeating tasks. Task recognition would help in these environments, since one member could observe another is performing a task and therefore know to move on to another. Since most robot task recognition schemes focus solely on improving recognition capability, the schemes neglect that the actor can modify the paths or trajectories followed. An analogy for this work is two SCUBA divers about to dive in the water. Before the dive, they discuss on the boat which hand gestures they will use to communicate underwater. Because these gestures vary by country or organization, it takes some repetition and negotiation to come to an agreement on which gestures will be used and what they indicate. Using such an analogy as inspiration we introduce the Trajectory Adaptation for Recognition (TAR) Framework.

The overall idea of the TAR Framework can be seen in Figure 21. In this simplified figure the axes represent two parameters that can encode a space of trajectories. For example, width and length of an oval. A single trajectory in this figure is represented as a red dot. Let us assume trajectory recognition were to use these two parameters. In order to get the highest recognition accuracy for two different trajectories then we would want them to be as far away as possible in this space. For example, one trajectory would have the parameters as 0,0 or the origin while the second trajectory would be at the upper right hand corner. Because any vehicle is restricted by its dynamics, represented as a blue oval projection onto this space, a trajectory chosen must also be viable. We also must make sure that any trajectory performed by the actor will be perceptible by the observer. The projection of the sensor's properties onto the trajectory space is shown as a white oval in the figure. Requiring that a trajectory be identifiable means that any trajectory we chose must not only be viable for the actor but perceptible by the observer. The last consideration we will apply is that of task properties. For example, we may require that a trajectory cover a search area. The task properties for a trajectory are represented as brown ovals in the figure. In essence, we require for the TAR Framework to automatically find trajectories that are highly recognizable given the observer's sensor, viable for the actor to perform, and conform to the task properties.

The TAR framework, seen in Figure 22, relies on teammates performing trajectory negotiation as part of a locker room agreement [59]. The term locker room agreement was coined by Peter Stone and is described

> "as set by the team when it is able to privately synchronize. It defines the flexible teamwork structure and the inter-agent communication protocols, if any."

Before a mission starts, the teammates that will use the TAR framework will begin by exchanging important information. The observer must have a description of its sensor characteristics and a trajectory recognition method. The actor must have a set of template trajectories, a way to describe (if not simulate) its own vehicle dynamics, and task requirements which the template trajectories should fulfill. The trajectory adaptation process begins with a simulation of the actor performing a trajectory. This trajectory is then passed through a simulation of the observer's sensor. Recognition is then performed on the trajectory that the observer has sensed. Acceptable results are determined by the recognition accuracy and the task requirements. If the results are

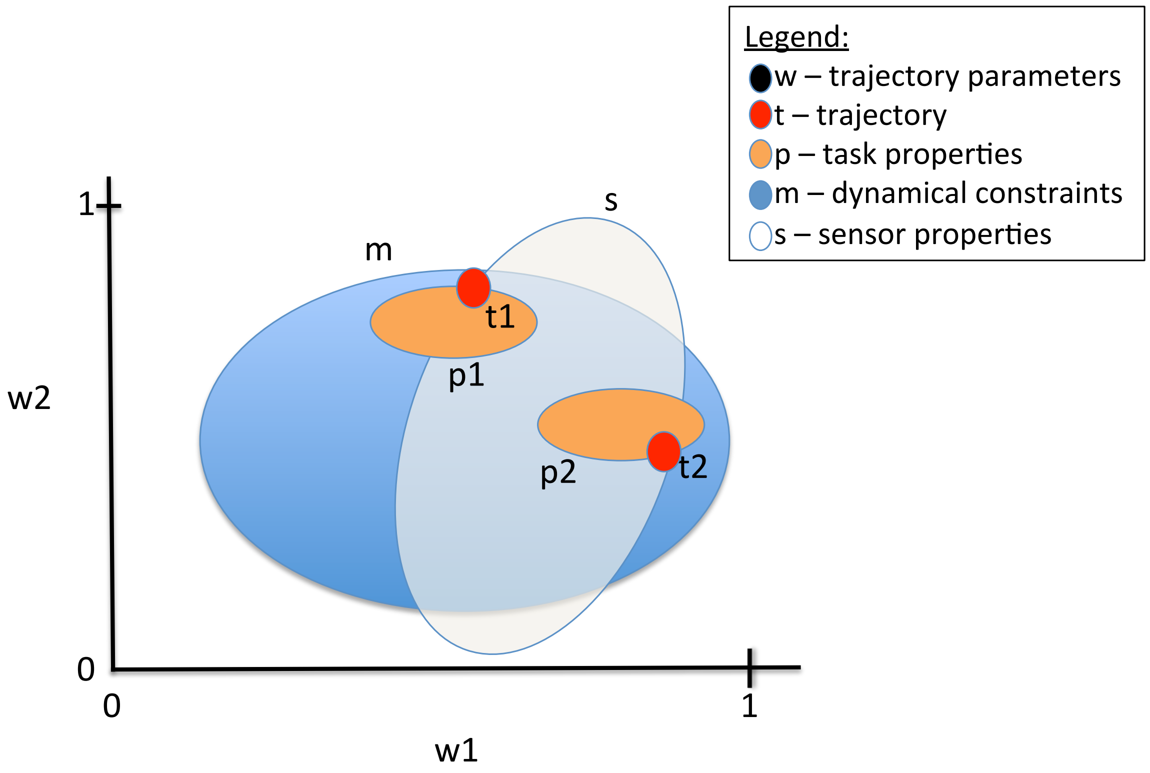


Figure 21: In the simplified big picture, trajectories, shown as red dots, are defined by a set of parameters, for ease of clarity the w1 and w2 axis. The **w** parameters are used for trajectory following, modification, and recognition. The actor is an autonomous vehicle and therefore is constrained by its dynamic model, which is represented as a blue oval. The sensor model of the observer is represented by a white oval. The task requirements or properties are represented by brown ovals. The problem is to find the trajectories that are most recognizable by the observer and that the actor can actually perform while still meet the task properties.

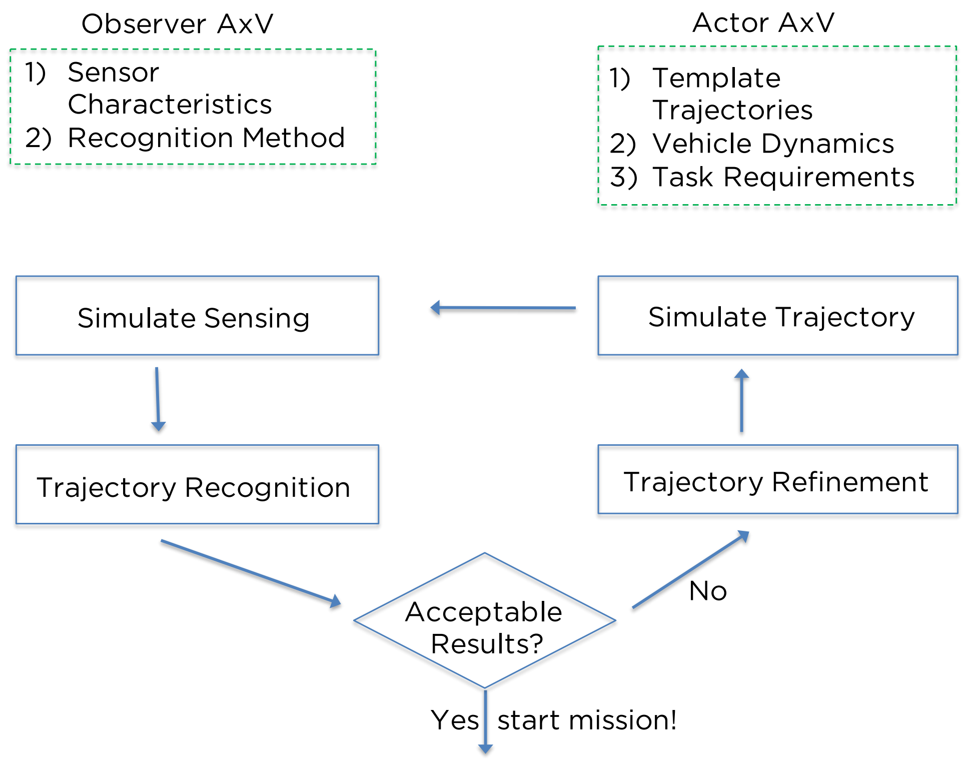


Figure 22: The General Trajectory Adaptation for Recognition (TAR) Framework. The TAR Framework begins with a locker room agreement exchange of the actor and observer information, depicted as dashed green boxes. This allows for trajectory adaptation to loop through simulated dynamics, sensing, recognition, and refinement until acceptable results are obtained.

acceptable, then the mission may start using any multi-robot coordination strategy. If the results are not acceptable, then the trajectory(s) are refined according to a chosen process. Trajectory refinement accounts for the vehicle's dynamics and presents an achievable trajectory(s) to the simulator so the process can begin another loop. The TAR framework is flexible in that this entire process can be performed on a central server, on just one of two teammate robots, or iteratively between two robots.

Every module of the framework is flexible in that different parts can be implemented according to requirements. For example, one simulator can perform both the vehicle dynamics and the sensing. Trajectory recognition can use any number of algorithms from probabilistic graphical models to more geometric approaches. The refinement can be performed with gradient descent or evolutionary algorithms. These decisions can be made to be centralized or decentralized based on the autonomous vehicles and mission at hand.

6.2 Canonical Representation

In the following notation a scalar is denoted as a lower case letter, x. A column vector is denoted by a bold lowercase letter $\mathbf{x}$ with a matrix denoted by a bold capital letter $\mathbf{X}$. A transpose is indicated by an apostrophe $'$. The column vector of ones is denoted by $\mathbf{1}$.

The original trajectory $\mathbf{X}^o$ is an $n \times 2$ matrix where n is the number of points in the trajectory. $\mathbf{x}$ and $\mathbf{y}$ are column vectors representing the x and y coordinates, $\mathbf{x} = [x_1 \ldots x_n]'$, $\mathbf{y} = [y_1 \ldots y_n]'$, then $\mathbf{X}^o = [\mathbf{x}, \mathbf{y}]$. The empirical mean for each dimension of the trajectory is calculated using, $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$ and $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i$. The trajectory is transformed to the origin by subtracting the empirical means for each dimension for each point in the trajectory using,

$$\mathbf{X}^c = \mathbf{X}^o - \mathbf{1}[\bar{x},\ \bar{y}]. \tag{18}$$

The eigenvectors and eigenvalues of the $\mathbf{X}^c$ matrix are found using SVD where $\mathbf{X}^c = \mathbf{USV}'$. Here $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices and $\mathbf{S}$ is a diagonal matrix containing the singular values. The columns of $\mathbf{U}$ and $\mathbf{V}$ are eigenvectors corresponding to eigenvalues in $\mathbf{S}$. The eigenvectors and eigenvalues of $\mathbf{X}^c$ are rearranged in descending order. The trajectory $\mathbf{X}^c$ is rotated using the ordered eigenvectors so that the trajectory has its dominant variance along the x-axis,

$$\mathbf{X}^{cr} = \mathbf{X}^c\mathbf{V}. \tag{19}$$

The trajectory is shifted into a positive frame by first finding the minimum for both $\mathbf{x}$ and $\mathbf{y}$ column vectors using $x_{\text{min}} = \min(\mathbf{x})$ and $y_{\text{min}} = \min(\mathbf{y})$. Subtract these to place the trajectory into a positive frame using,

$$\mathbf{X}^p = \mathbf{X}^{cr} - \mathbf{1}[x_{min},\ y_{min}]. \tag{20}$$

Each dimension of the trajectory is analyzed individually. Let l_i be arc length up to index i, where

$$l_i = \sum_{j=1}^{n-1} \|\mathbf{X}^p_j,\ \mathbf{X}^p_{j+1}\|. \tag{21}$$

Here l_n will be the total arc length of the trajectory. We then perform analysis on X^* given by augmenting $\mathbf{X}^p$,

$$\mathbf{X}^* = [\mathbf{X}^p,\ \mathbf{l}] = [\mathbf{x}^p,\ \mathbf{y}^p,\ \mathbf{l}], \tag{22}$$

where $\mathbf{l}$ is a normalized column vector of arc lengths, normalized using total arc length $\mathbf{l}_n$. This will allow us to later compare trajectories of varying arc lengths.

A form of trajectory representation must be selected. Originally, waypoints were selected as a representation. However, given two waypoints that represent a straight line, their perturbation still produces a straight line. Thus, a representation is chosen that allows for a description of the trajectory with parameters. Approximations of trajectories can be performed with Fourier basis or Cheybyshev polynomials, as examples. In these experiments we use radial basis functions (RBFs) to approximate each dimension in the trajectory. A general radial basis function based on a Gaussian is denoted as $\phi(r) = e^{-(\epsilon r)^2}$ where r is the euclidean distance metric from a point to the center, $r = \|x - center\|$. We specifically use the radial basis $e^{-(\frac{(x-mean)^2}{2*variance})}$. Radial basis functions were chosen because they allow for local control of perturbing a trajectory with the modification of one corresponding weight.

In order to fit each dimension of the trajectory, radial basis functions are spread along the normalized arc length evenly starting with one RBF centered at 0 and then evenly spaced until the last RBF is located centered at 1. The variance is chosen so that there is enough support to approximate the trajectory. For example, if we have only two RBFs to approximate a straight line and their variance is too narrow then the approximation will not fit a line well as there will be a dip or curve to the line. In practice, a minimum of four RBF means were required to approximate common trajectories within acceptable error. The following experiments varied the number of radial basis means starting with a minimum of six for flexibility.

The matrix Φ is an $n \times b$ matrix where n is the number of points in the original trajectory and b is the number of RBF means, or centers, along the normalized arc length [0,1] of the trajectory. The location $\Phi(i, j)$ in the matrix represents the value of the j^{th} radial basis function with respect to the i^{th} position along the trajectory defined as,

$$\Phi(i, j) = e^{-(\frac{(x_i - center_j)^2}{2*variance})}. \tag{23}$$

The original trajectory can be approximated using the general form $y(x) = \sum_{i=1}^{b} \omega_i \phi(r)$ where ω_i is a weight corresponding to the i^{th} RBF. Because these weights are linear with respect to ϕ we can find them using the pseudo inverse, $\mathbf{w}_Y = (\Phi^T\Phi)^{-1}\Phi\mathbf{x}^{*'}$ and $\mathbf{w}_X = (\Phi^T\Phi)^{-1}\Phi\mathbf{y}^{*'}$. In the reverse direction, given the weights the trajectory can be approximated and reconstructed with the original Φ matrix using $\mathbf{y}^* = \Phi \mathbf{w}_Y$ and $\mathbf{x}^* = \Phi \mathbf{w}_X$.

6.3 Recognition

Recognition of trajectories can be performed with the weights of the approximating radial basis functions (RBFs), described in the previous section. The weights of template trajectories are compared to the weights of a new sample trajectory with a distance measure of choice, as seen in Figure 23. The new sample trajectory is labelled as one of the templates if it is the closest, in weight space, and only if within a threshold. Using a threshold removes the issues of some false positives. Just because a template trajectory is the closest, in weight space, to a sample trajectory among all the other templates does not indicate that it is the correct label. The overall trajectory recognition algorithm using radial basis functions is seen in Algorithm 1.

Algorithm 1: Trajectory Recognition with Radial Basis Functions

```
Input: $k$ Template Trajectories (T), Sample Trajectory (S), Number of Basis
       Centers(M), Variance for Each Basis(V)
Output: Sample Trajectory Label
THRESHOLD = 0.1?;
// All trajectories are in x,y form:
for $i \leftarrow 1$ to $k$ do
    $\mathbf{W}_i \leftarrow$ findRadialBasisFunctionWeights($\mathrm{T}_i, M, V$);
$\mathbf{W}_s \leftarrow$ findRadialBasisFunctionWeights(S,M,V);
// Measure distance of sample trajectory to all templates
d $\leftarrow \infty$;
label$_s \leftarrow$ 'no label';
for $i \leftarrow 1$ to $k$ do
    temp $\leftarrow$ distance($\mathbf{W}_i$, $\mathbf{W}_s$);
    if temp $< d$ && $temp < THRESHOLD$ then
        d = temp;
        label$_s \leftarrow i$;
```

In order to determine the recognition accuracy of all the trajectories together, a simulated mission is performed. During the simulation, as the autonomous vehicle

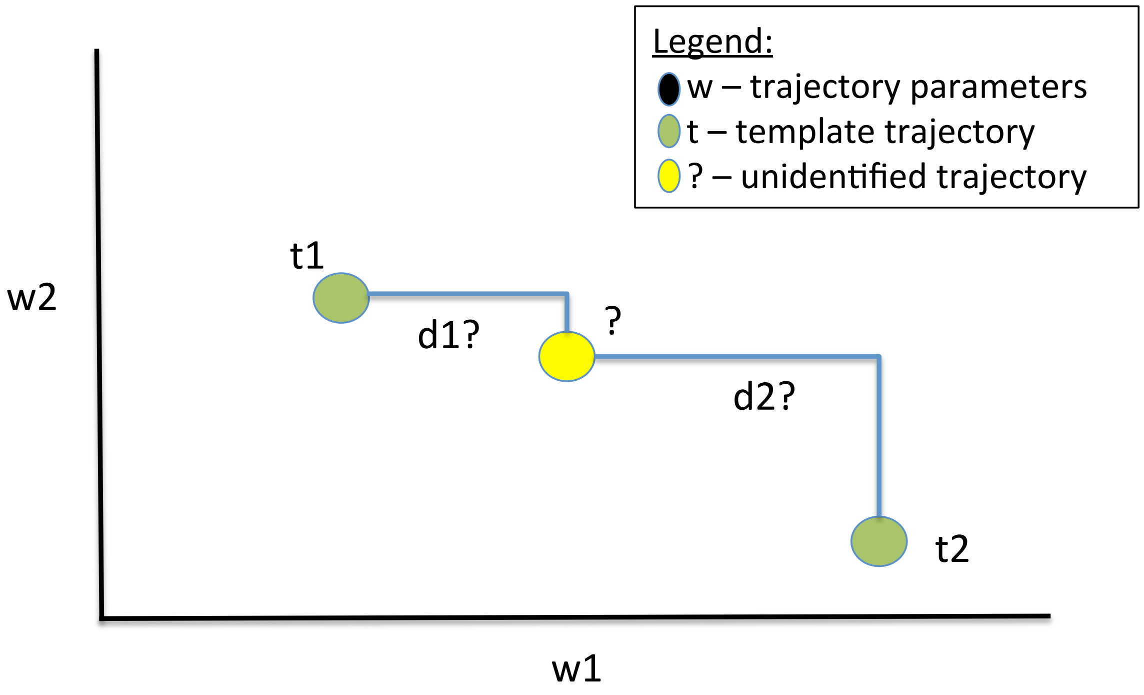
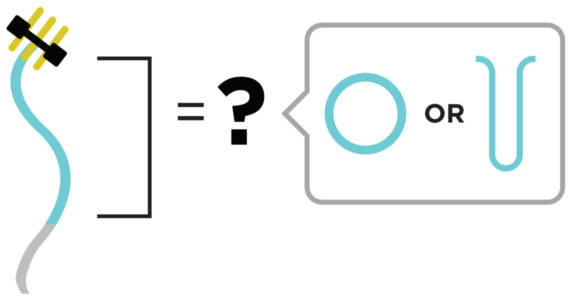
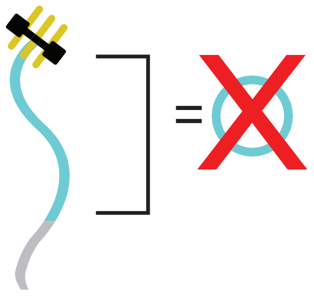


Figure 23: Trajectory recognition with DMP weights. Template trajectories are created during the negotiation process, are represented as green dots. A new trajectory, represented as a yellow dot, is performed by the actor. The L1 norm or manhattan distance is used in order to determine which trajectory it most resembles, within a threshold.



(a) Trajectory Labelling



(b) Recognition Error

Figure 24: In (a) trajectory recognition is performed by presenting a portion of a simulated mission to the recognition algorithm. The closest trajectory template in weight space is given a label if it is within a threshold. Recognition error is the mis-identification of a trajectory in the window such as in (b).

performs each of the trajectories, it labels them for ground truth. Then the recognition algorithm uses a trajectory window to scan over the simulated mission, as seen in Figure 24a. Using Algorithm 1, the entire mission is labelled automatically. The results of the recognition algorithm can be displayed as a confusion matrix, seen in Table 15. The rows are the trajectory that has been performed, also known as ground truth labels, and the columns represent the labels produced by the algorithm. Ideally, the confusion matrix should look like an identity matrix in that the only entries in the matrix that have numbers are the diagonals and the off-diagonal entries are zeros. For clarity, Table 15 has the diagonal entries which indicate accuracy are highlighted grey and the off-diagonal entries which indicate incorrect labels are in white. Incorrect labels are the result of recognition error, seen in Figure 24b, and are what the TAR framework aims to reduce.

6.4 Task Objectives

Desired properties of common trajectories must be expressed in terms of cost functions of that trajectory or set of trajectories. For example, a prominent task that autonomous vehicles perform is search of an area. Ensuring that a trajectory still properly searches an area must be put in terms of a cost so that the optimization mechanism can produce trajectories that in this case maximize the search coverage. The following are task objectives expressed as cost functions. The task costs are clamped between zero and one so that they are all on the same scale. As the system

Table 15: Example Confusion Matrix. Correct Labels are highlighted grey and white signifies error

		Label			
		SearchPattern	*Loiter*	*GoToWaypoint*	*NoLabel*
Actual	*SearchPattern*	92.73	0.00	0.08	7.19
	Loiter	20.87	60.97	0.00	18.16
	GoToWaypoint	31.07	0.87	32.91	35.15
	Transition	58.54	0.00	0.00	41.46

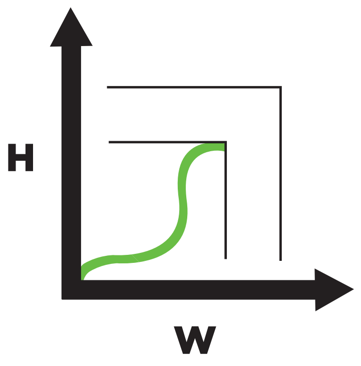
is minimizing the costs, a one indicates the worst possible value or threshold and a zero represents the best possible value.

Within Area: It is desirable to keep a trajectory within a specified area, otherwise the trajectory may occupy a space that is larger than even the operating area. As seen in Figure 25a, in order to achieve this objective the bounding box of the trajectory is calculated. The total area is then linearized between zero and one based on the smallest acceptable area and the largest acceptable area.

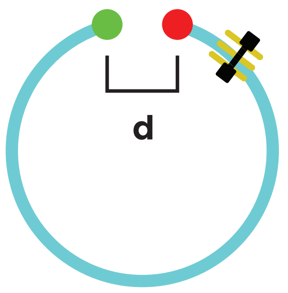
Min Distance First and Last Point: Some tasks require that the start and end location be sufficiently close, such as most loiter patterns. For this objective, the distance between the first and last point of a trajectory is calculated, as seen Figure 25b. The distance is linearized between zero and one based on the smallest and largest, respectively, acceptable distance.

Orientation Similarity Between First and Last Pose: For repeating trajectories, such as loiters, it is desired to have the final orientation of the vehicle align with the initial orientation which facilitates repetition, as seen in Figure 25c. The direction of the first position is the difference between the second location in the trajectory and the first. The direction of the last location is the difference between the last location and the one prior. The cosine between the two orientations is calculated using the dot product of the two vectors divided by the multiplication of each vector's norm. This produces the cosine value between the two orientations which range between -1 and 1. The cosine value is 1 if the orientations are aligned, a zero if orthogonal, and -1 of opposite.

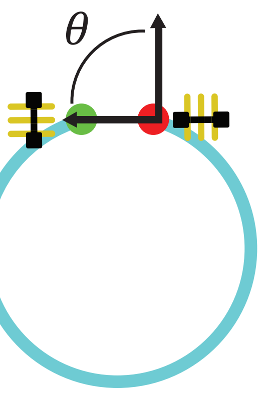
Within Radii: In order to maintain a circle loiter a trajectory is rewarded with a lower cost if it's points are within a radius of the center of the bounding box of the trajectory. In order for the system to have some latitude, a minimum and maximum radius are defined, see in Figure 25d. The distance for each point in the trajectory from the center is calculated. One minus the ratio of points within the radii versus



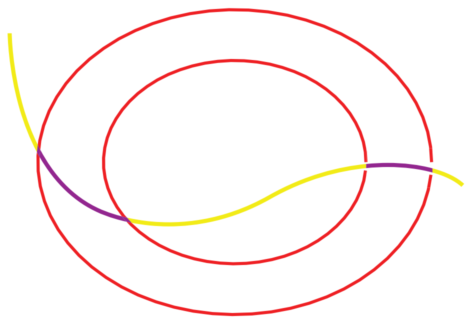
(a) Within Area



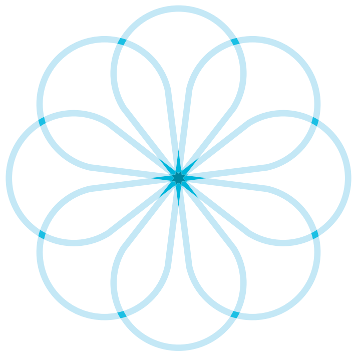
(b) Min Distance First and Last Point



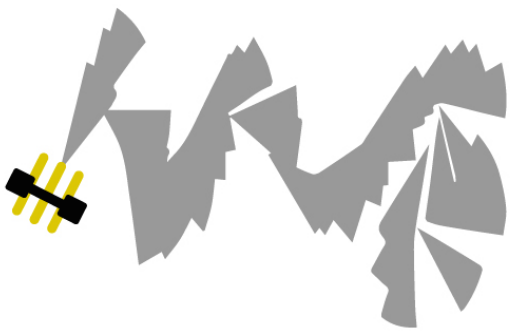
(c) Orientation Similarity



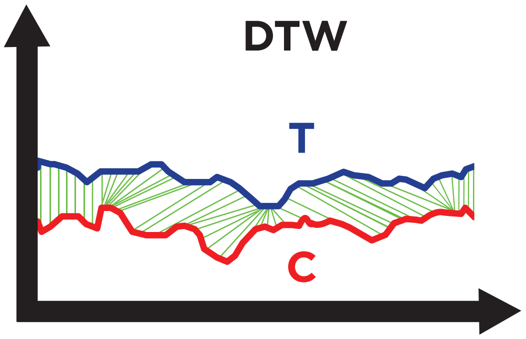
(d) Within Radii



(e) Mark Spot



(f) Sensor Coverage



(g) Actionability (DTW)

Figure 25: Example Task Objectives

the entire trajectory is calculated,

$$cost = 1 - \frac{pts\_within\_radii}{total\_num\_pts}. \tag{24}$$

This corresponds to a value between zero if all the points are within the radii and a one if none of the points are within the radii.

Mark Spot: In this team cooperation framework, some trajectories have been used to indicate an important location. The location was indicated using a *figure 8 loiter* where the important location is the intersection of the trajectory in the middle. The goal is to find a trajectory with the highest density of trajectory visits at the location of interest and when not close to the location to spread the density of the trajectory points to as low as possible, an example is seen in Figure 25e. The bounding box of the trajectory is discretized. The bin with the most trajectory points is marked as the location of interest. Then the average of all the points in the remaining bins in the discretization is calculated. The score for this objective is the ratio of the average number of points remaining divided by the number of points in the interest location. In some cases the MarkSpot task objective is called MaxPoints.

Sensor Coverage: Sensor coverage is an important objective for most autonomous vehicle applications. A sensor's field of view is simulated along the robot's trajectory. As seen in Figure 25f, the area covered by the sensor is depicted in grey and the area not covered is in white. The cost is one minus the ratio of covered area divided by the total area of the trajectory's bounding box.

Actionability: To ensure that an autonomous vehicle can follow a trajectory some measure must be employed. In this work dynamic time warping (DTW) is used to measure the difference between the ordered trajectory and the one actually followed by the autonomous vehicle, illustrated in Figure 25g. DTW calculates an optimal match between two sequences [62]. The distance between each point in the commanded trajectory and each point in the trajectory followed is calculated. Starting at the first two points, the cumulative distance to get to a corresponding point pair is calculated by adding the minimum distance to get to the pair of points and the distance between them. The final cumulative distance value calculated for the final two points is the distance required to make perturbations between corresponding points so that the two trajectories match exactly. It is this final distance that is used as a similarity measure between the commanded trajectory and that followed by the autonomous vehicle. The lower the measure indicates that the vehicle is closer to following the ordered trajectory. A zero indicates a perfect reproduction while a larger number reflects a poor following. This is important as some trajectories are difficult for an autonomous vehicle to follow and may make loops in order to visit waypoints it may have missed due to dynamics earlier.

6.5 Total Cost

The total cost is,

$$C = R + A + O \tag{25}$$

where R is the sum of the recognition error, A is the actionability cost, and O is the sum of the different task objectives. The recognition error, *rec_error*, is the summation of the off-diagonal entries of the confusion matrix, described in Section 6.3. The actionability cost in all of these experiments is the dynamic time warping (DTW) cost between the commanded trajectory and that followed by the autonomous vehicle, described in Section 6.4. The objectives or task properties, O, may include any of the objectives described above, i.e. sensor area coverage, within a radius of the center, and min area.

6.6 Objective Weights

In general, the total cost is a linear combination of costs that each have a value between zero and one. Zero indicates the best possible value and a one represents the worst possible value. In certain cases it became difficult to maintain or induce the desired properties for a given task. In such cases, a weighting function was applied to a cost to give it a higher priority. The desired properties of the weighting function are that it start at the origin, be piece wise continuous, and positive between the domain of zero to one. An example weighting function is a quadratic: $w(x) = ax^2$, where a is a positive coefficient and x is the original cost between zero and one. The total cost remains a linear combination of costs but some of them might be prioritized by a weighting function indicated by w as in the following example:

$$C = R + A + w_1(O_1) + O_2 \tag{26}$$

which equates to

$$C = R + A + a_1(O_1)^2 + O_2. \tag{27}$$

Where task objective two O_2 was not weighted and kept its range of zero to one but task objective one O_1 was prioritized by a quadratic weight function, w_1.

6.7 Optimization

In general, the TAR Framework allows for any choice of optimization algorithm. Because it was desired to be as flexible as possible, an evolutionary strategy was chosen. This allows for the use of task objectives to be specified as cost functions without the need for explicit derivatives. Additionally, an evolutionary algorithm allows for the exploration of a complex fitness landscape. In these experiments trajectory exploration was performed using Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) [21]. CMA-ES is an evolutionary algorithm for non-linear non-convex black-box optimization. It is a second-order approach estimating a covariance matrix within an iterative procedure. CMA-ES does not compute just the best set of parameters per iteration. It in fact calculates a mean and covariance matrix of possible best trajectories. It does this through repeated iterations with several samples at each iteration. For this work, the parameters for each sample were the radial basis function weights for the trajectory to be adapted. The fitness score of each sample was the total cost which is a linear combination of the recognition error and the objective functions for each task, as described in Section 6.5.

6.8 Experiments

This section describes the experimental results of the Trajectory Adaptation for Recognition (TAR) Framework. The first experiments, reported in Section 6.8.3, demonstrate the TAR Framework's ability to modify trajectories for the purpose of improving recognition accuracy. Simply modifying trajectories for recognition improvement does not ensure that the commanded trajectory is the one followed by the autonomous vehicle. Therefore an actionable objective is added to the overall cost function which helps guide the TAR Framework to produce trajectories that a specific vehicle can accomplish, reported in Section 6.8.4. This then leads to individual experiments, reported in Section 6.8.5, attempting to produce commonly performed tasks in the autonomous vehicle domain including sensor coverage and cirlce loiter to ensure that the task objectives are appropriate measures for maintaining task properties. Following common maneuvers, creating a mark the spot task is attempted, reported in Section 6.8.6, which is important to the type of autonomous vehicle cooperation described in Chapter 5. Once these objectives have been validated they are attempted on more complex scenarios. The first is a common scenario found in the autonomous underwater vehicle domain in which the same circle loiter is performed for various states of the vehicle such as waiting for commands, error, and mission termination. In order to disambiguate these states, the TAR Framework attempts to perturb them for improved recognition accuracy while maintaining a maneuver similar to a circle loiter, reported in Section 6.8.7. The final experiments, reported in Section 6.8.8, push the expressive power of the TAR Framework by attempting to start from simple shapes into maneuvers that accomplish tasks such as sensor coverage, circle loiter, and mark the spot while improving recognition accuracy. Key insights from these experiments include the efficacy of certain task objectives to accomplishing tasks, balancing many tasks in one optimization with the use of priority weighting functions, the number of parameters to represent the trajectories, and the impact of varying initial starting locations for improved task objective success.

6.8.1 Experimental Setup

The simulation environment for our experiments is provided by iMarineSim and pMarineViewer which are part of the MOOS-IvP open source autonomy package [8]. The pMarineViewer module is a GUI-based tool, as seen in Fig. 16b, that renders 2D overhead maps of the vehicles performing behaviors. The iMarineSim is a single-vehicle simulator that updates vehicle state based on actuator values. Actuator values are produced by the IvP Helm, which is a coordinator over activated behaviors. The simulator tools allow for verification of vehicle behaviors and interactions. The experiments are performed using trajectory data gathered through simulation. The trajectories are run within iMarineSim and viewed through pMarineViewer which shares its plotted trajectories with our trajectory recognition module. The locations of the AUVs are recorded as each trajectory is performed.

6.8.2 Measures of Success

There are many different criteria for termination of trajectory exploration. A generally accepted terminating condition is when the total cost falls bellow a certain user defined threshold. Because CMA-ES is a stochastic process, it can get stuck in local minima and thus the cost does not vary and plateaus. Such a situation calls for a termination when the cost does not vary more than a threshold for a period of iterations. Although this termination criteria may inadvertently stop the optimization process from locally perturbing to an improved total cost. A third criteria is when the algorithm reaches a predetermined number of iterations. A strategy for such evolutionary algorithms is to run repeated experiments in order to find the resulting evolution with the lowest cost or qualitative properties.

Measures of success depend on the task and can be either quantitative or qualitative. For example, when only recognition accuracy is the requirement then once the desired recognition accuracy has been reached then the system terminates. If the system terminates for multiple runs without reaching the desired cost then it is considered a failure. On the other hand, in tasks such as mark the spot, the TAR Framework will terminate runs and a user may decide that qualitatively the resulting trajectories are not desirable. This begs the question of whether the appropriate cost objectives to reach their desired tasks are included.

6.8.3 Recognition Improvement

In order to demonstrate that the TAR framework can increase recognition accuracy by modifying a trajectory, an experiment starting with two straight lines is performed. Starting from two similar trajectories demonstrates the worst possible recognition error.

Experiment 1 Setup

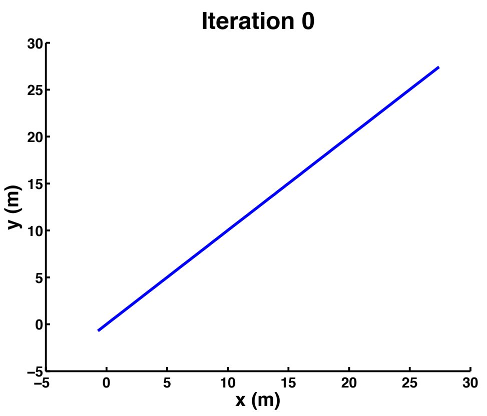
Table 16: Experiment 1 Setup

Task	Initial Trajectory	Perturb	Cost Functions
1) Straight Line	Straight Line	N	N/A
2) Adapted Line	Straight Line	Y	R

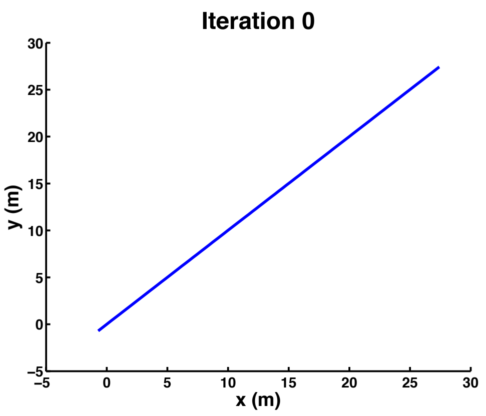
# RBFs Per Dim:12	Initial Sigma: 2

In this experiment the TAR Framework begins with two straight lines, as seen in Figure 26, and modifies one of them to improve the recognition accuracy. The experimental setup is contained in Table 16. The trajectories are approximated by 12 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectory being adapted. The total cost is defined as,

$$C = R \tag{28}$$



(a) Trajectory 1.



(b) Trajectory 2.

Figure 26: Initial set of trajectories to be recognized for experiment 1. Both trajectory one and two, seen in (a) and (b), have been initialized as straight lines to induce recognition confusion. The TAR Framework will perturb the second trajectory in order to reduce recognition error.

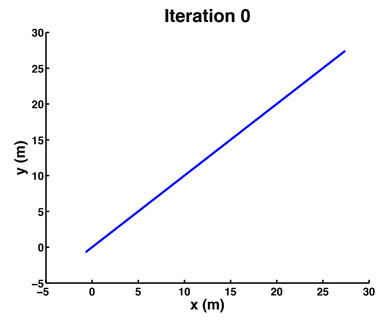
Table 17: Experiment 1 Iteration and Cost

iteration	0	6	53	58	77	92
cost	1.23	0.798	0.709	0.647	0.617	0.485

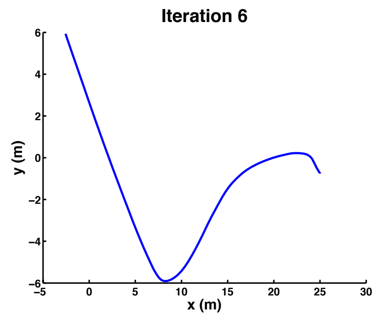
where R is defined as,

$$R = \sum_{i=1}^{p} recognition\_error(i). \tag{29}$$

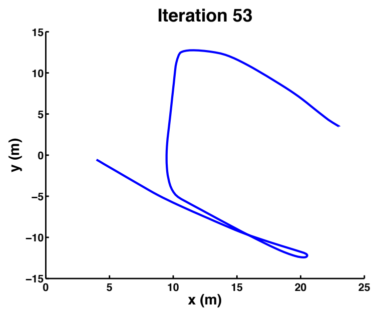
Results The evolution of the second trajectory is seen in Figure 27. Figure 28a displays the best cost per iteration throughout the experiment. Table 17 contains highlighted iteration costs seen starting at 1.23 and ending at 0.485 on iteration 92. Table 18 contains the confusion matrices for the highlighted iterations. The off diagonals of the confusion matrix for the *ith* row corresponds to the $recognition\_error(i)$ for the *ith* trajectory. The original confusion matrix in Table 18a demonstrates that there is a lack of recognition accuracy as they are all below 90%. By Iteration 53 the confusion matrix demonstrates that the accuracy for both trajectories is about 90% as seen in Table 18d. At iteration 92 the overall cost or recognition error has reduced for all labels however it has reduced the recognition accuracy for the first trajectory below 80%, as seen in Table 18f. The set of trajectories to be recognized at iteration



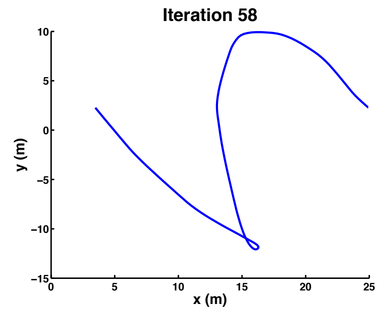
(a) Iteration 0.



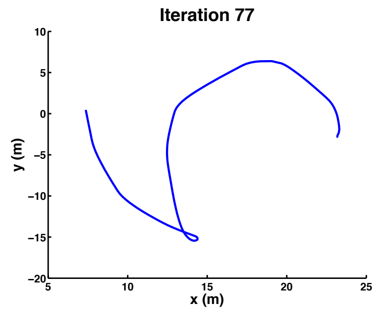
(b) Iteration 6.



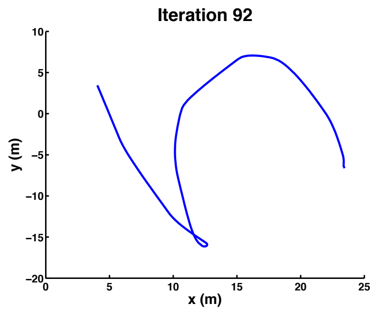
(c) Iteration 53.



(d) Iteration 58.

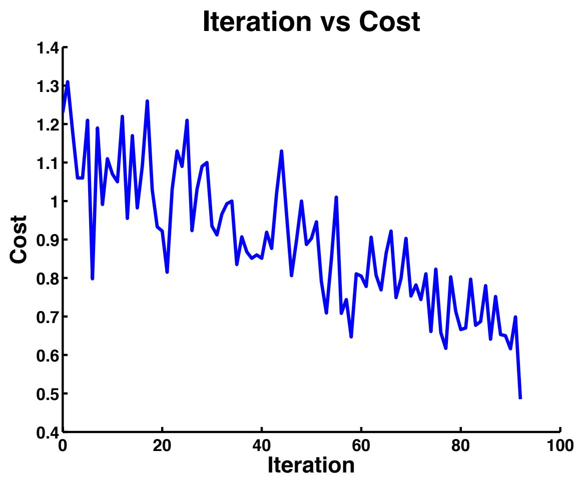


(e) Iteration 77.

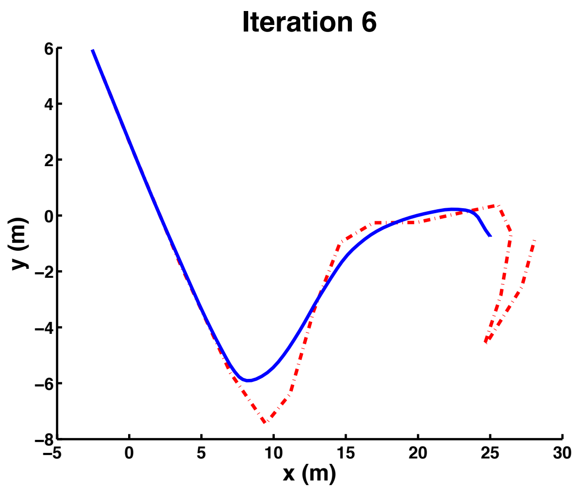


(f) Iteration 92.

Figure 27: Experiment 1 Trajectory Adaptation. The TAR Framework is designed to improve recognition accuracy by modifying trajectories. The two trajectories are initialized as straight lines, seen in (a). The first trajectory remains a straight line yet the second trajectory is allowed to be adapted and it's evolution can be seen in (b), (c), (d), (e), and (f).



(a) Iteration vs Cost



(b) Example Tracking Error

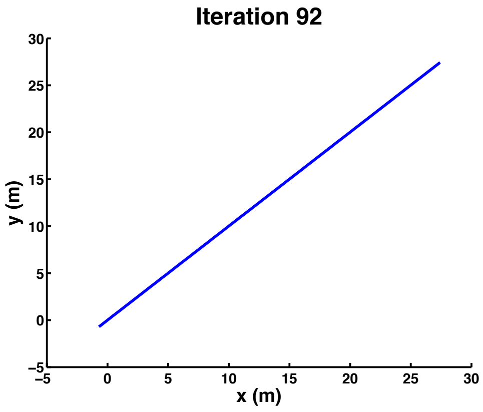
Figure 28: Experiment 1 Illustrative Figures

92 are seen in Figure 29.

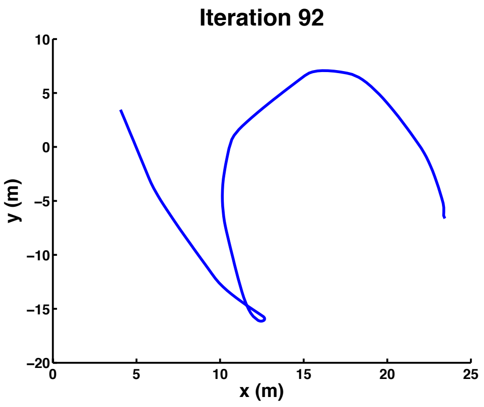
Discussion

The improvement of the total cost or in this case only recognition error is seen in Figure 28a. It is important to note the use of *recognition_error* for all the possible labels. By iteration 58 the accuracy for both trajectories has improved over 90% for both labels. However, there is labelling error for when the vehicle is in transition by being labelled as either Trajectory 1 or 2. This error in labelling transitions as either trajectory is improved in iteration 92. However, this improvement in NoLabel for Transitions decreases the accuracy for labelling trajectory 1 below 80%. This raises an important point about which error to include for recognition accuracy improvement. In some situations the accuracy of recognizing just one of the trajectories is most important and reducing the confusion error for just that trajectory is vital. In such a situation the recognition error may only need to be the summation of the off diagonals for the row and column corresponding to that one important trajectory. It is a balance between the relevant recognition for the task.

A second important observation is the evolution of the single trajectory. As seen in Figure 28b there is a difference between the commanded waypoints in red and the actual trajectory followed by the autonomous vehicle in blue. The error in tracking the commanded waypoints is a combination of autonomy behavior parameters and vehicle dynamics. For example, the behavior-based autonomy in MOOS-IvP has a parameter called capture radius. The capture radius being too large accounts for gaps between the red trajectory and the blue. The error in tracking also stems from the vehicle dynamics. The end of the commanded trajectory in Figure 28b has a sharp hook. Even if the capture radius was small, the vehicle dynamics would allow it to reach the tip of the hook but not be able to track tightly the curve to the end. Instead, the vehicle would have performed a loop to return in the direction needed to track the last waypoints, such as the loops found in the later iterations.



(a) Trajectory 1.



(b) Trajectory 2.

Figure 29: The final set of trajectories to be recognized at iteration 92 in experiment 1. Both trajectory one and two were initialized as straight lines. In order to reduce recognition error, the TAR Framework adapted trajectory two, seen in (b), and kept trajectory one stable, seen in (a).

Conclusion

Although trajectory recognition accuracy was improved, the commanded trajectories were not well followed. This leads to the inclusion of an actionable objective to the trajectory which is investigated in the next experiment.

Table 18: Experiment 1 Confusion Matrices

		Label		
		NoLabel	*GoToWaypoint*	AdaptedTraj
Actual	*Transition*	23.70	46.20	30.10
	GoToWaypoint	0.00	89.40	10.60
	AdaptedTraj	0.00	35.70	64.30

(a) Iteration 0.

		Label		
		NoLabel	*GoToWaypoint*	AdaptedTraj
Actual	*Transition*	31.30	48.20	20.50
	GoToWaypoint	0.00	100.00	0.00
	AdaptedTraj	0.00	11.10	88.90

(b) Iteration 6.

		Label		
		NoLabel	*GoToWaypoint*	AdaptedTraj
Actual	*Transition*	41.30	52.80	6.74
	GoToWaypoint	10.20	89.80	0.00
	AdaptedTraj	0.00	1.15	98.90

(c) Iteration 53

		Label		
		NoLabel	*GoToWaypoint*	AdaptedTraj
Actual	*Transition*	40.90	44.30	14.80
	GoToWaypoint	4.08	95.90	0.00
	AdaptedTraj	0.00	1.49	98.50

(d) Iteration 58

		Label		
		NoLabel	*GoToWaypoint*	AdaptedTraj
Actual	*Transition*	40.20	42.70	17.10
	GoToWaypoint	1.96	98.00	0.00
	AdaptedTraj	0.00	0.00	100.00

(e) Iteration 77

		Label		
		NoLabel	*GoToWaypoint*	AdaptedTraj
Actual	*Transition*	73.70	17.50	8.77
	GoToWaypoint	22.20	77.80	0.00
	AdaptedTraj	0.00	0.00	100.00

(f) Iteration 92

6.8.4 Recognition Improvement and Actionable

In the previous experiment, the TAR Framework was modifying trajectories without regard to whether or not a vehicle could follow them which resulted in commanded trajectory tracking errors. In order to address this issue, an objective is added called actionable (Act). This objective encourages the TAR framework to produce trajectories that the autonomous vehicle can track within acceptable limits. The actionable (Act) objective is implemented in these experiments using Dynamic Time Warping (DTW) comparing the commanded waypoints to the trajectory actually followed by the autonomous vehicle. DTW gives a measure between two trajectories in the form of what changes of distance would need to be made between points in one trajectory to match the points in another. For example, two identical trajectories will result in a DTW cost of zero. As two trajectories differ more and more the DTW cost increases. A second task objective is added to the total cost called Within Area, which is designed to keep the trajectories within a reasonable operating area. More importantly, the Within Area objective keeps the trajectories from growing too large to be useful. The following experiments will both have the same cost function but will only differ in the turn rate of each simulated vehicle. A vehicle that has a higher turn rate can produce more intricate trajectories in a smaller area than a vehicle with a lower turn rate. It is predicted that by including the Within Area and Act task costs will encourage the TAR Framework to produce trajectories that are recognizable and actionable based on each vehicle's dynamics in a small operating area. In these experiments the cost function is defined as,

$$C = R + Act + Area \tag{30}$$

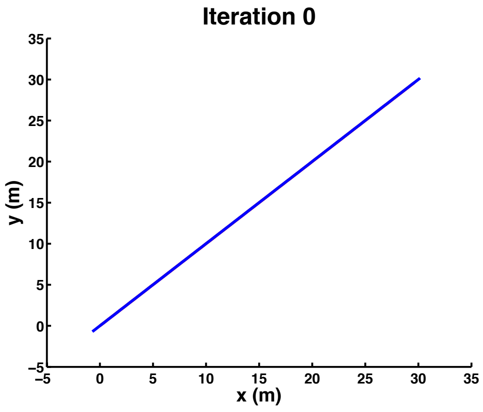
where R is defined as,

$$R = \sum_{i=1}^{p} recognition\_error(i). \tag{31}$$

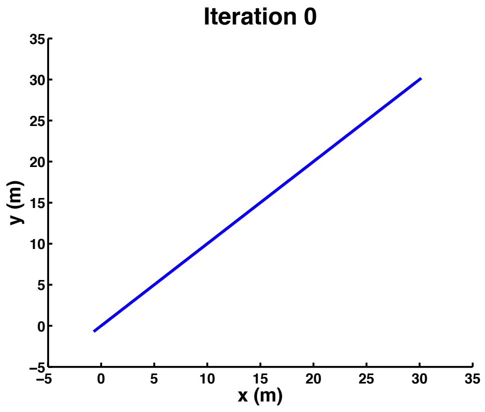
Experiment 2 Setup

Table 19: Experiment 2 Setup

Task	Initial Trajectory	Perturb	Cost Functions
Straight Line	Straight Line	N	N/A
Adapted Line	Straight Line	Y	$R + Act + Area$
# RBFs Per Dim: 12		Initial Sigma: 2	



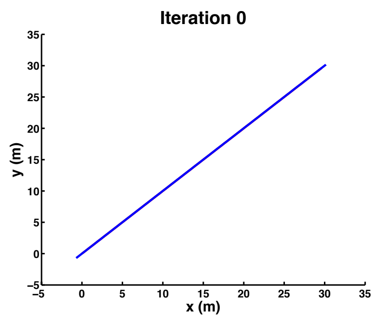
(a) Trajectory 1.



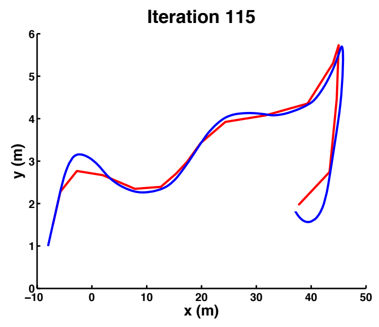
(b) Trajectory 2.

Figure 30: Initial set of trajectories to be recognized for experiment 2. Both trajectory one and two, seen in (a) and (b), have been initialized as straight lines to induce recognition confusion. The TAR Framework will perturb the second trajectory in order to reduce recognition error while maintaining vehicle dynamics and keeping within an area.

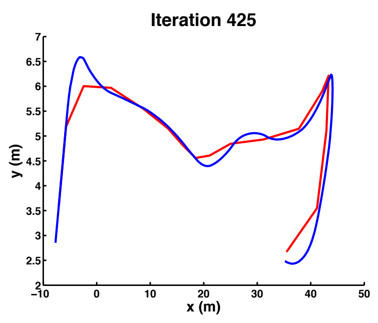
In this experiment the TAR Framework begins with two straight lines, seen in Figure 30, and modifies one of them to improve the recognition accuracy while ensuring the autonomous vehicle can perform the commanded trajectory. Table 19 contains the experimental setup information. The trajectories are approximated by 12 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectory being adapted. In this experiment the total cost is defined in Equation 30. The key difference between this and the following experiment is that the simulated vehicle has a turn rate that is set to 100. Such a turn rate allows the vehicle to be more maneuverable. The combination of including the Act and Area task costs along



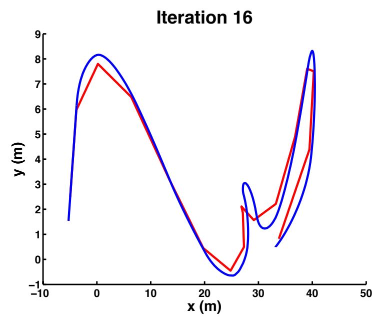
(a) Iteration 0.



(b) Iteration 115.



(c) Iteration 425.



(d) Worst DTW.

Figure 31: Experiment 2 Trajectory Adaptation. In (a) the initial trajectory is seen. The trajectories in red and blue are the commanded trajectories and actual vehicle trajectories, respectively. As the TAR Framework attempts to improve recognition, it is also keeping the trajectories ordered within the capability of the acting vehicle. The adapted trajectories for iteration 115 and 425 are seen in (b) and (c), respectively. The commanded trajectory in (d) has the worst Act (DTW) cost as it is too difficult for the vehicle to follow given it has a jagged edge in the middle.

with recognition accuracy in the total cost function should produce commanded trajectories that the vehicle can track with little error and be performed within a small operating area.

Results The evolution of the trajectory is seen in Figure 31. Table 20 contains the highlighted iteration costs seen starting at an initial recognition error cost of 1.979 and improving to 0.3802 at iteration 425. Figure 32 contains graphs of iterations vs various costs. It is fascinating to see that the Max and Min DTW cost, seen in Figures 32a and 32b, begin to level out between 0.4 and 0.25 between iterations 300 and 400. This indicates that the system has found a local minima which is supported by the iteration by total cost, seen in Figure 32d. The final set of trajectories to be

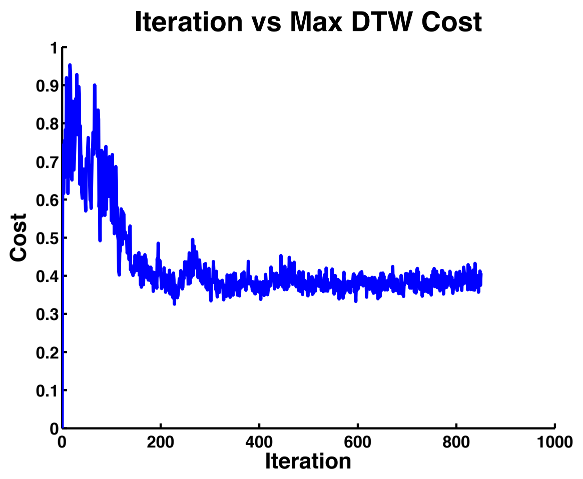
Table 20: Experiment 2 Iteration and Costs

Iteration	R	Act	Area	Total
0	1.979	0.091	0.000	2.071
115	0.504	0.284	0.035	0.824
425	0.380	0.252	0.000	0.632

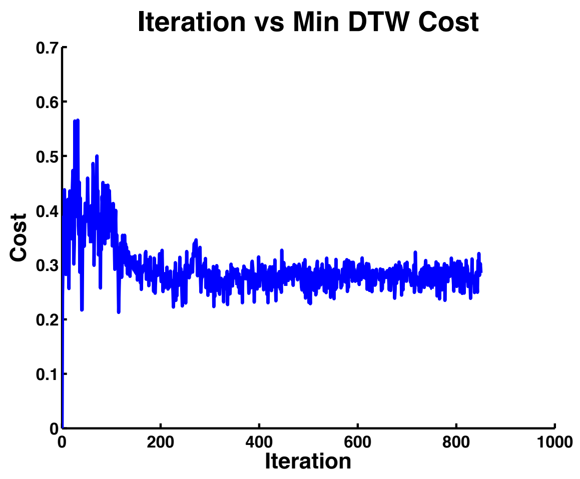
recognized at iteration 425 can be seen in Figure 33.

Discussion This is the most responsive vehicle that these experiments simulate in MOOS-IvP with a turn rate parameter of 100. Therefore, it is not surprising that the vehicle can follow a commanded trajectory with a sharp right turn, as seen in Figure 31c. However, there is at least some tracking error in Figure 31d with a sharp jagged bump in the middle. More extreme turns may be attempted by the TAR Framework if the operating area was even further reduced in size.

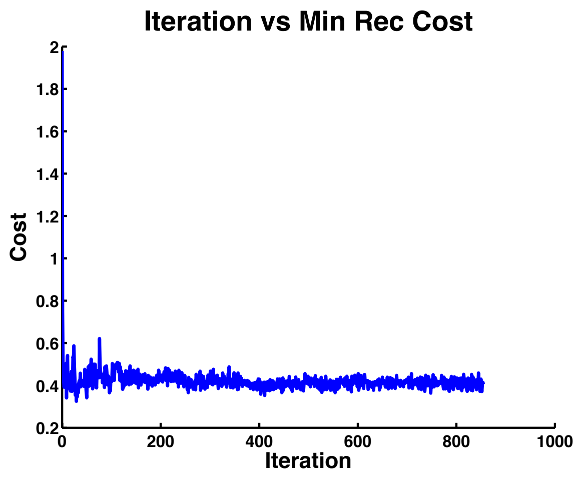
Conclusion This vehicle is responsive to the commanded trajectory. We expect to see worse tracking error in the following experiment as the turn rate will be reduced and therefore the simulated vehicle will be less able to follow sharp changes in the commanded trajectory.



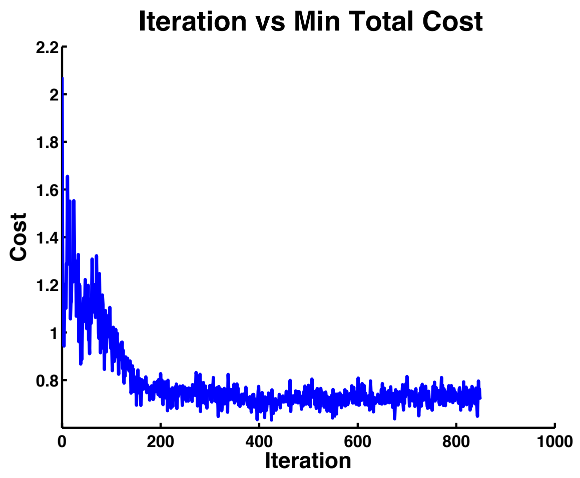
(a) Iteration vs Max DTW.



(b) Iteration vs Min DTW.

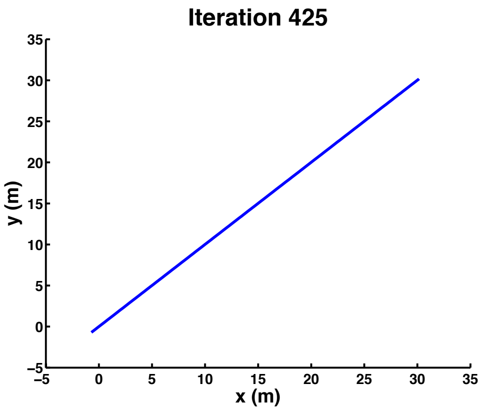


(c) Iteration vs Rec Score Min.

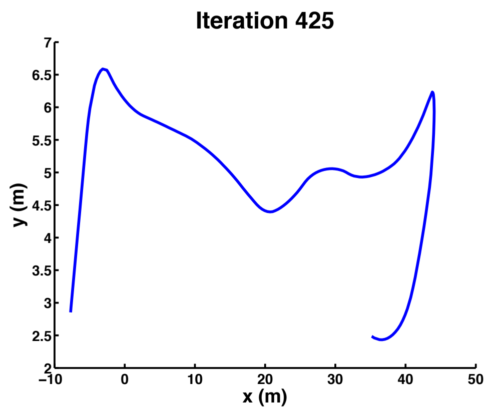


(d) Iteration vs Total Cost.

Figure 32: Experiment 2 Iterations vs Various Costs.

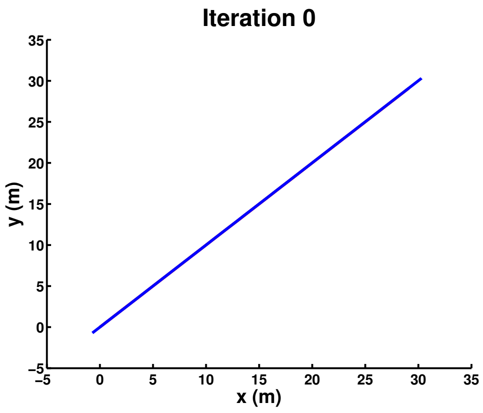


(a) Trajectory 1.

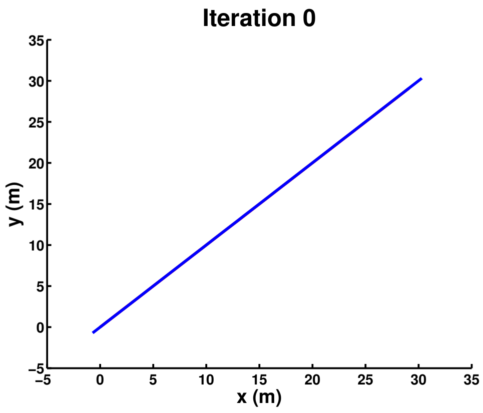


(b) Trajectory 2.

Figure 33: The final set of trajectories to be recognized at iteration 425 in experiment 2. Both trajectory one and two were initialized as straight lines. In order to reduce recognition error, the TAR Framework adapted trajectory two while maintaining vehicle dynamics and keeping within a specified area. Trajectory one and two at iteration 425 are seen in (a) and (b), respectively.



(a) Trajectory 1.



(b) Trajectory 2.

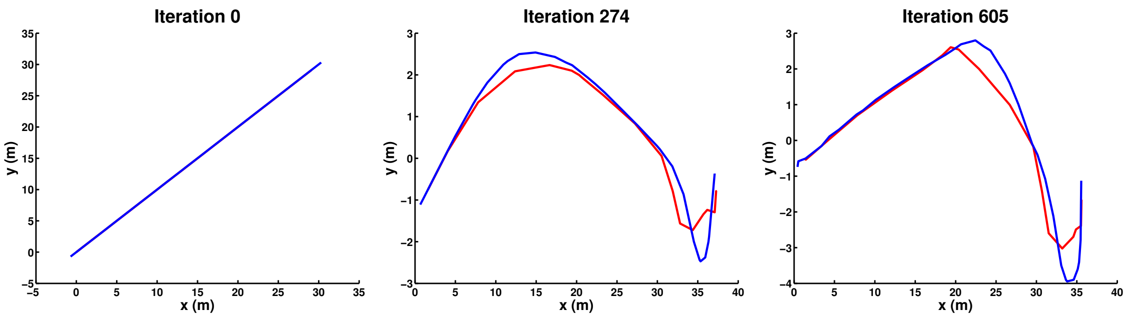
Figure 34: Initial set of trajectories to be recognized for experiment 3. Both trajectory one and two, seen in (a) and (b), have been initialized as straight lines to induce recognition confusion. The TAR Framework will perturb the second trajectory in order to reduce recognition error while maintaining vehicle dynamics and keeping within an area.

Experiment 3 Setup

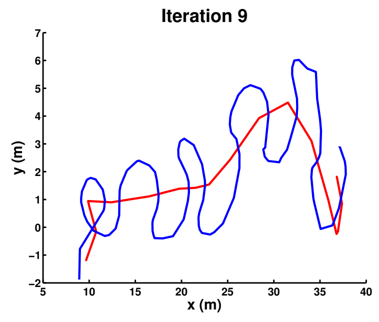
Table 21: Experiment 3 Setup

Task	Initial Trajectory	Perturb	Cost Functions
Straight Line	Straight Line	N	N/A
Adapted Line	Straight Line	Y	$R + Act + Area$
# RBFs Per Dim: 12		Initial Sigma: 2	

Similarly to the previous experiment, experiment 3 begins with two straight lines, as seen in Figure 34, and modifies one of them to improve the recognition accuracy while ensuring the autonomous vehicle can perform the commanded trajectory. The experimental setup is contained in Table 21. The trajectories are approximated by 12 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectory being adapted. The total cost is defined in Equation 30, which is the same cost function as the previous experiment. The key difference between this and the previous experiment is that the simulated vehicle has a turn rate that is set to 50. Such a turn rate reduces the maneuverability of the simulated vehicle.



(a) Iteration 0. (b) Iteration 274. (c) Iteration 605.



(d) Worst DTW.

Figure 35: Experiment 3 results. The trajectories in red and blue are the commanded trajectories and actual vehicle trajectories, respectively. As the system attempts to improve recognition, it is also keeping the trajectories ordered within the capability of the acting vehicle. The initial trajectory is seen in (a) where the commanded red trajectory is overlaid by the blue actual trajectory. The adapted trajectories are seen at iterations 274 and 605 in (b) and (c), respectively. The commanded waypoints in the red trajectory shown in (d) is too close for the vehicle to follow given its slow turn rate and thus produces a trajectory with many oscillations as the vehicle misses a waypoint due to the vehicle dynamics and circles around.

The combination of including the Act and Area task costs along with recognition accuracy in the total cost function should produce commanded trajectories that the vehicle can track with little error and be performed within a small operating area. As in the previous experiment, the TAR Framework starts with two straight line trajectories and only adapts the second trajectory.

Results The evolution of the trajectory can be seen in Figure 35. Various plots of iterations by different costs is seen in Figure 36. The modifiable trajectory begins as a straight line at iteration 0, seen in Figure 35a, and can be seen to be modified at iterations 274 and 605 in Figures 35b and 35c, respectively. The trajectory attempted

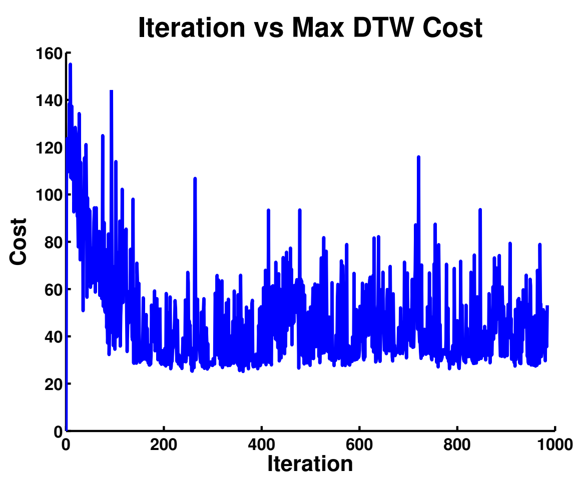
Table 22: Experiment 3 Iteration and Costs

Iteration	R	Act	Area	Total
0	1.627	0.000	0.000	1.627
274	1.026	0.000	0.000	1.026
605	0.434	0.074	0.000	0.509

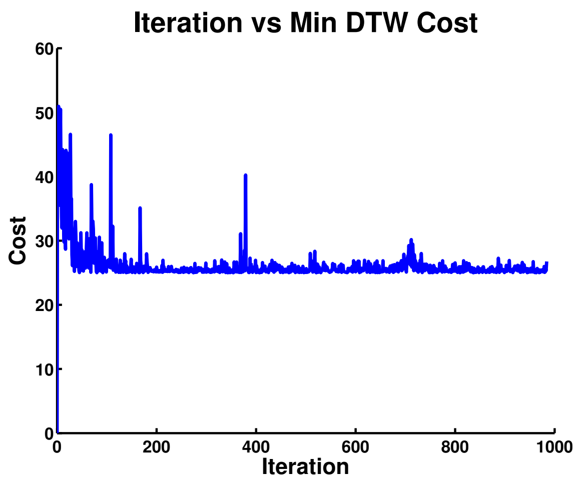
by the TAR framework with the worst DTW cost occurred on iteration 9, seen in Figure 35d. Table 22 contains the highlighted iterations and their respective costs. Overall, the total cost per highlighted iteration lowers starting at 1.627 at the initial iteration to 0.509 at iteration 605. The final set of trajectories to be recognized are seen in Figure 37.

Discussion As can be seen in Figure 35 the trajectories favored by the TAR framework for this low turn rate vehicle are smoother than the previous experiment. The commanded trajectory with the worst DTW cost has obvious tracking errors with oscillations in attempts to recover missed waypoints.

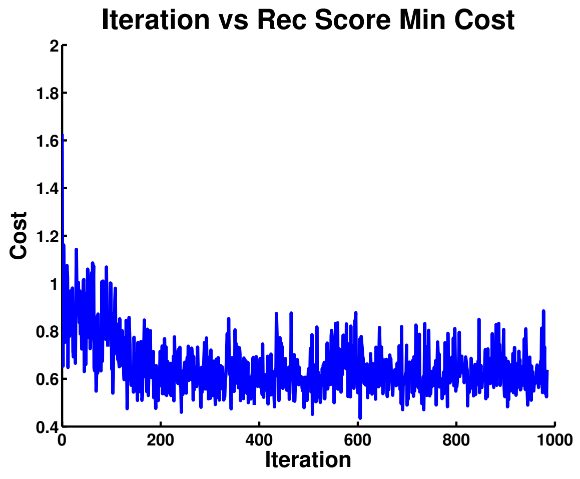
Conclusion The DTW cost is a useful measure for whether or not a simulated autonomous vehicle can follow a commanded trajectory. As can be seen by the different outcomes for the different simulated turn rate vehicles, the TAR framework favors trajectories that produce the lower overall cost. As with the vehicle with a 50 turn rate, it produces trajectories that are more amenable to a smoother slope. The 100 run rate simulated vehicle, on the other hand, produces a trajectory that has a sharp right hand turn.



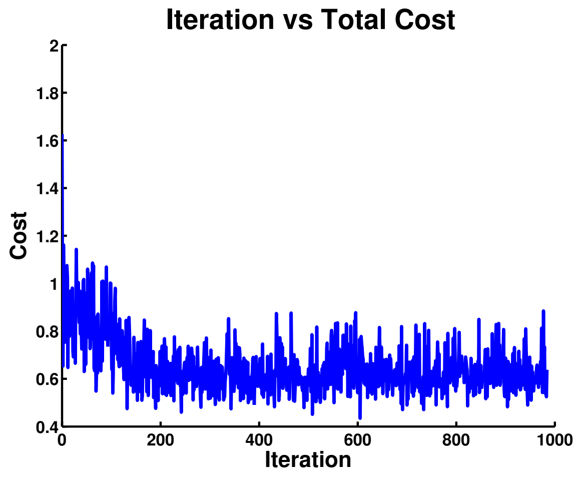
(a) Iteration vs Max DTW.



(b) Iteration vs Min DTW.

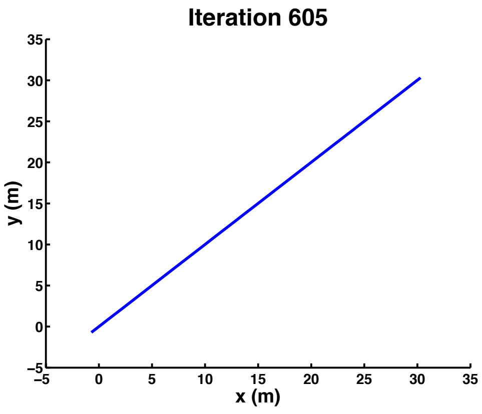


(c) Iteration vs Min Rec Error

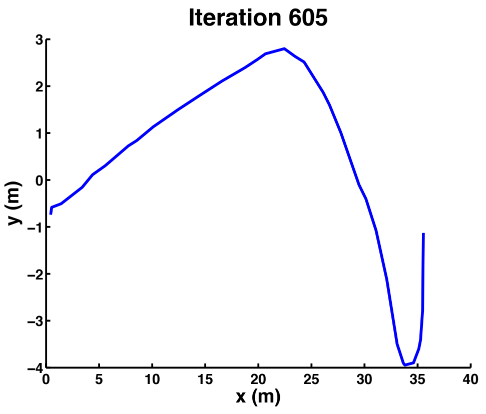


(d) Iteration vs Total Cost

Figure 36: Experiment 3 Iterations vs Various Costs.

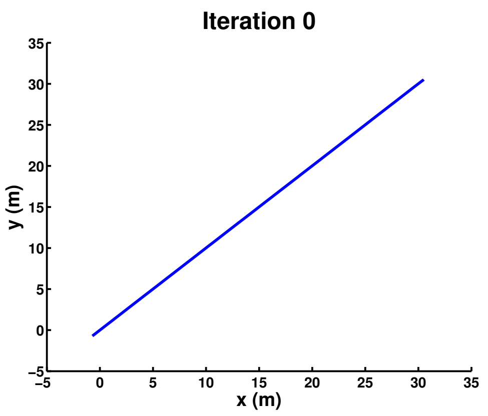


(a) Trajectory 1.

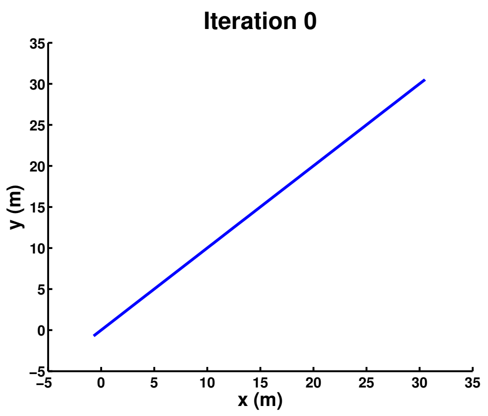


(b) Trajectory 2.

Figure 37: The final set of trajectories to be recognized at iteration 605 in experiment 3. Both trajectory one and two were initialized as straight lines. In order to reduce recognition error, the TAR Framework adapted trajectory two while maintaining vehicle dynamics and keeping within a specified area. Trajectory one and two at iteration 605 are seen in (a) and (b), respectively.



(a) Trajectory 1.



(b) Trajectory 2.

Figure 38: Initial set of trajectories to be recognized for experiment 4. Both trajectory one and two, seen in (a) and (b), have been initialized as straight lines to induce recognition confusion. The TAR Framework will perturb the second trajectory in order to reduce recognition error while maintaining vehicle dynamics, keeping within an area, and increasing area coverage.

6.8.5 Common Task Objectives

In the following experiments, various task or trajectory objectives are implemented in the form of cost functions. For example, the cost function for a search task is sensor coverage. The cost function for maintaining a circle loiter is the within radii cost. The following experiments are performed in order to determine that the formulations of the task objectives as cost functions is appropriate. The elements of the cost function from the previous experiments remain such as recognition error, actionability, and min area.

Experiment 4 Setup

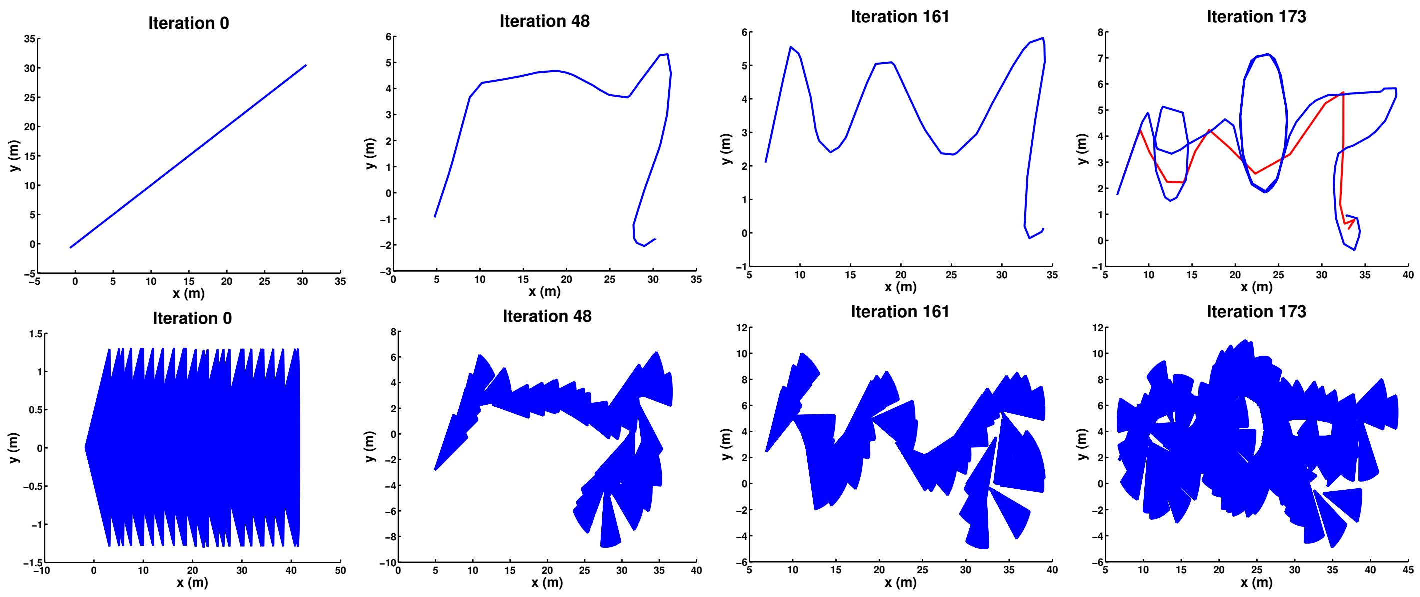
Table 23: Experiment 4 Setup

Task	Initial Trajectory	Perturb	Cost Functions
Straight Line	Straight Line	N	N/A
Search	Straight Line	Y	*Cover*
# RBFs Per Dim: 12		Initial Sigma: 2	

In experiment 4, the TAR framework begins with two straight lines, as seen in Figure 38, and adapts one of them for improved recognition and for improved sensor coverage. The experimental setup is contained in Table 23. While the list of cost functions for the adapted trajectory only displays $Cover$, it does include $R + Act + Area$ but is not displayed for brevity. The trajectories are approximated by 12 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectory being adapted. In this experiment the TAR framework is augmented with the inclusion of a new task cost: sensor coverage. The sensor of choice in this experiment is a simulated sonar. It is simulated as a simple triangle directed in the orientation of the autonomous vehicle. The sensor coverage objective, was previously described in Section 6.4, is defined as, $1 - \frac{covered\ pixels}{total\ pixels}$. The total cost is defined as,

$$C = R + Act + Area + Cover. \tag{32}$$

It is predicted that the inclusion of the sensor coverage objective will create trajectories that cover an operating area as thoroughly as possible while keeping recognition error low.



(a) Iteration 0. (b) Iteration 48. (c) Iteration 161. (d) Best Sonar.

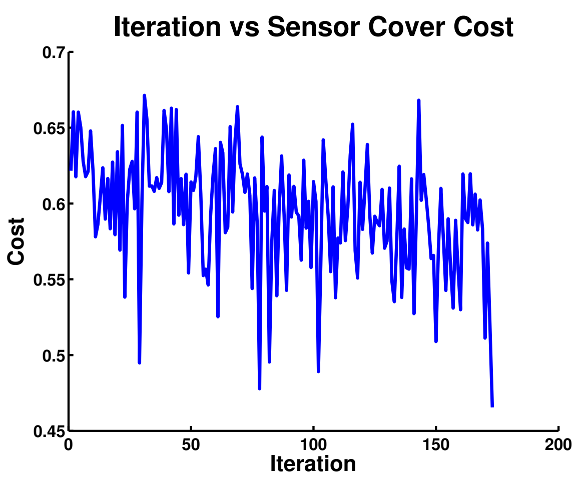
Figure 39: Experiment 4 results. The top row figures are the trajectories performed by the autonomous vehicle in blue and on the bottom row are their corresponding simulated sonar coverage. The TAR framework starts with two straight lines and is allowed to modify the second trajectory with the objectives: recognition error, within an area, and maximizing sensor coverage. The initial trajectory is seen in (a) with improvements in sensor coverage seen in (b) with a cost of 0.711 and (c) with a cost of 0.6512. An important thing to note is that in (d) the best sensor coverage is found at iteration 173 with a cost of 0.4655. However, it is also one of the worst trajectories at tracking the commanded waypoints as seen on the top of (d) with the red trajectory as the commanded one and the blue trajectory as the one executed by the autonomous vehicle.

Table 24: Experiment 4 Iteration and Coverage Cost

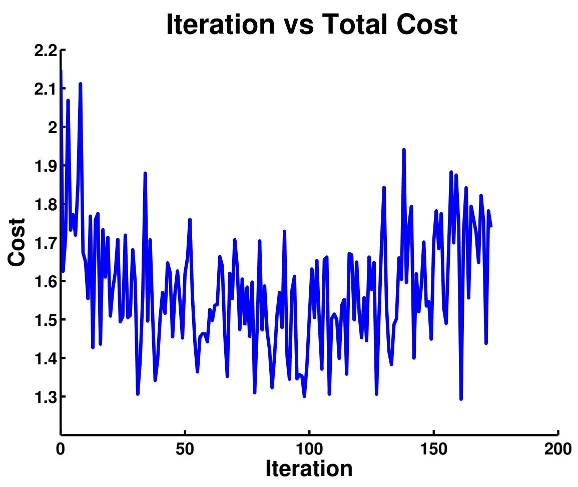
Iteration	Coverage Cost
48	0.711
161	0.651

Results The evolution of the trajectory is seen in Figure 39. The top row of Figure 39 displays the lowest cost trajectory per highlighted iteration while the bottom row displays each iteration's corresponding simulation of sensor coverage. Table 24 displays the coverage costs per highlighted iteration. Figure 40 displays iterations vs various costs. As this experiment begins with the trajectory as a straight line, Figure 39a displays the simulated sensor covering a straight line trajectory. As the trajectory is modified, its coverage of the area is improved and seen in Figure 39b. By Iteration 161, seen in Figure 39c, the modified trajectory has begun to look more like a search pattern seen performed by manned and unmanned vehicles. The final set of trajectories to be recognized are seen in Figure 41.

Discussion The inclusion of the sensor coverage objective is successful in creating a trajectory that covers an operating area. The fascinating discovery is that this task objective in particular will be the most successful at creating the desired task properties. However, it should be noted that this task objective will not always produce a trajectory that is as neat in its coverage. For example, in this result the coverage proceeds neatly from one end of the operating area to the other. In experiments presented further below, the coverage will be successful but may have loops in the trajectory. An important insight that continues from previous experiments is the importance of including the Actionable (Act) objective. As seen in Figure 39d, Iteration 173 produced a fantastic sensor coverage with a coverage cost as low as 0.4655. Although that particular trajectory covered the area well, it was due to tracking error produced by a combination of vehicle dynamics and autonomy with the desired trajectory in red and performed trajectory in blue. The Act cost thus

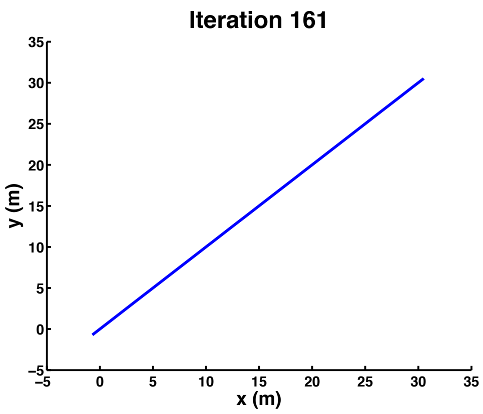


(a) Iteration vs Sensor Cover

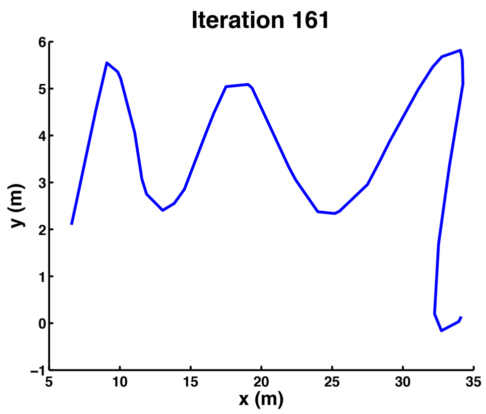


(b) Iteration vs Total Cost

Figure 40: Experiment 4 Iterations vs Various Costs



(a) Trajectory 1.

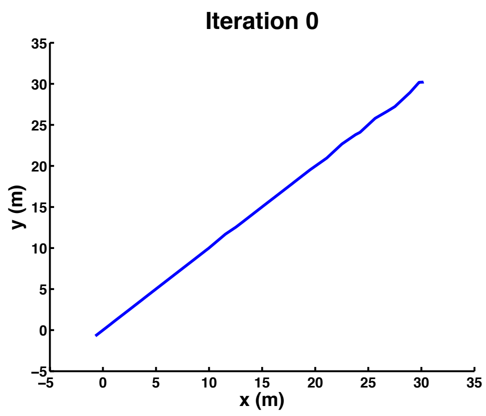


(b) Trajectory 2.

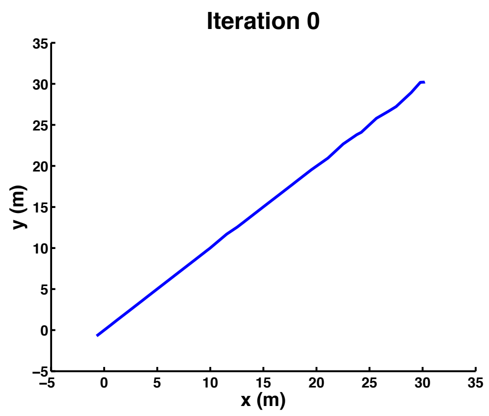
Figure 41: The final set of trajectories to be recognized at iteration 161 in experiment 4. Both trajectory one and two were initialized as straight lines. The TAR Framework adapted trajectory two in order to reduce recognition error and increase sensor coverage while maintaining vehicle dynamics and keeping within a specified area. Trajectory one and two at iteration 161 are seen in (a) and (b), respectively.

offset the improved sensor coverage cost and kept Iteration 173 out of contention for a viable trajectory.

Conclusion The sensor coverage objective is successful at producing trajectories that cover an operating area. The continued inclusion of the actionable (Act) objective is crucial to producing trajectories the autonomous vehicle can perform with little tracking error.



(a) Trajectory 1.



(b) Trajectory 2.

Figure 42: Initial set of trajectories to be recognized for experiment 5. Both trajectory one and two, seen in (a) and (b), have been initialized as straight lines to induce recognition confusion. With the goal of creating a circle loiter-like trajectory, the TAR Framework will perturb the second trajectory in order to reduce recognition error while maintaining vehicle dynamics, keeping within an area, keeping within radii, and minimizing the distance between the first and last points.

Experiment 5 Setup

Table 25: Experiment 5 Setup

Task	Initial Trajectory	Perturb	Cost Functions
Straight Line	Straight Line	N	N/A
Circle Loiter	Straight Line	Y	$Radii + DFirstLast$

# RBFs Per Dim: 12	Initial Sigma: 2

In experiment 5, the TAR framework begins with two straight lines, as seen in Figure 42, and adapts one of them for improved recognition and to produce a circle loiter-like shape. The experimental setup is contained in Table 25. While the list of cost functions for the adapted trajectory only displays $Raddi + DFirstLast$, it does include $R + Act + Area$ but is not displayed for brevity. The trajectories are approximated by 12 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectory being adapted. In order to produce trajectories that resemble or maintain a circle loiter, two new task objectives are implemented for

Table 26: Experiment 5 Iteration and New Task Costs

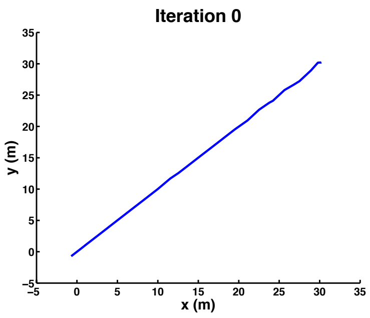
Iteration	0	56	60	72	78	237	661	696
Radii	0.892	0.818	0.634	0.618	0.764	0.830	0.512	0.575
DFirstLast	0.892	0.000	0.069	0.003	0.000	0.071	0.005	0.001

experiment five. Circle loiter is a common task performed by both manned and unmanned vehicles. In particular, autonomous underwater vehicles (AUVs) perform circle loiters for various mission states such as holding for commands, error state, and mission completion. The first new cost is within radius (Radii), which was described in Section 6.4, which is defined as, $1 - \frac{pts\ w/in\ radius}{total\ pts}$. This cost is designed to encourage the trajectory to be within a radius of the center. The second new cost is distance between first and last point (DFirstLast). DFirstLast is required and useful for any trajectory which is desired to be repeatable. The closer the first and last trajectory waypoints are from each other the sooner the autonomous vehicle can reproduce that trajectory. The cost function for this experiment is,

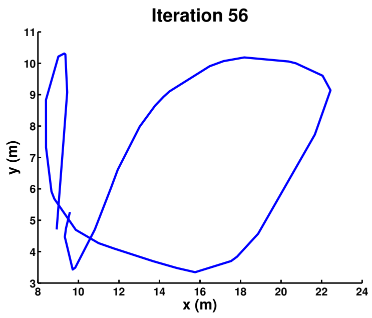
$$C = R + Act + Area + Radii + DFirstLast. \tag{33}$$

Results The evolution of the trajectory is seen in Figure 43. Figure 44 displays figures of Iteration vs various costs. Table 26 contains the within radius and distance between first and last points costs for each highlighted iteration. The highlighted iterations display the best, lowest, total cost along the optimization process. The objective of distance between start and stop points is obviously seen in the highlighted trajectories as the beginning and end of the trajectories are within a close proximity. The within radius objective is harder to visually observe in Figure 43 until the very final highlighted iterations of 661 and 696, seen in 43g and 43h, respectively. The overall improvement of Radii objective cost is clearly seen in Table 26 starting at 0.89 at iteration 0 and ending at 0.575 at iteration 696. The final set of trajectories for recognition is seen in Figure 45.

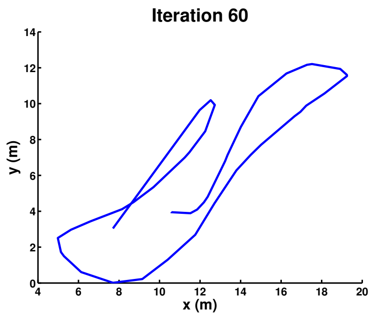
Discussion It is important to note that the within radius objective does not



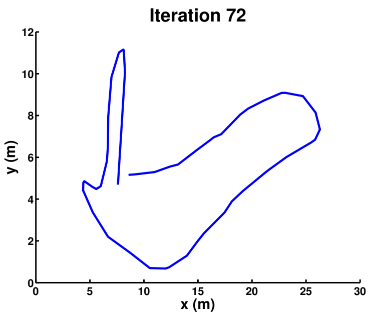
(a) Iteration 0.



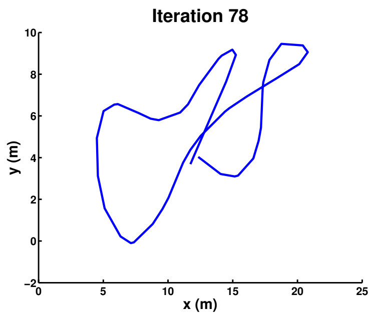
(b) Iteration 56.



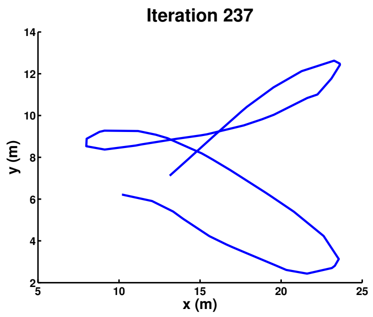
(c) Iteration 60.



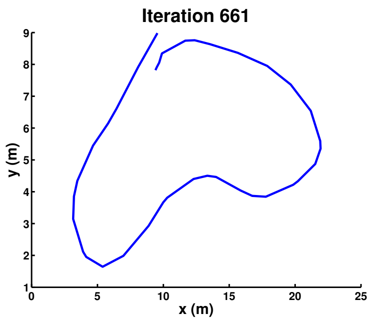
(d) Iteration 72.



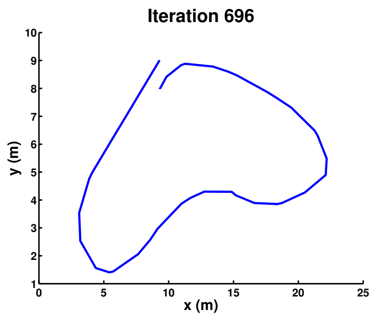
(e) Iteration 78.



(f) Iteration 237.

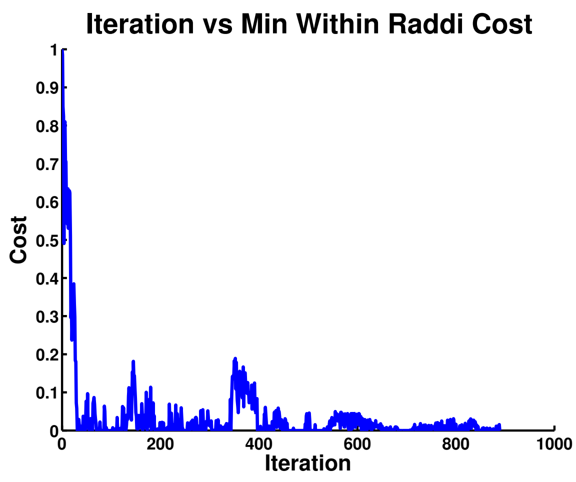


(g) Iteration 661.

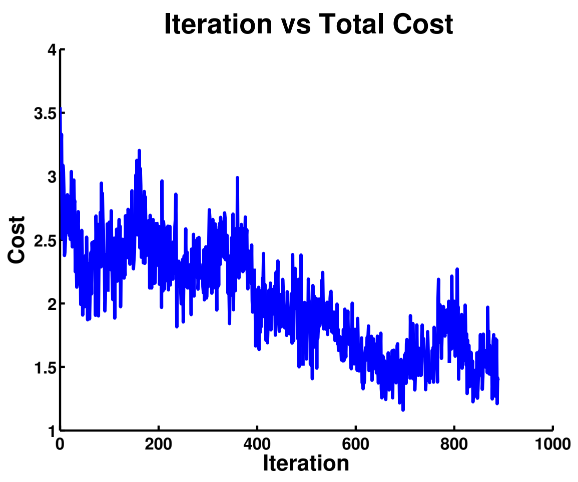


(h) Iteration 696.

Figure 43: Experiment 5 results. The TAR framework begins with two straight lines as seen in (a). Adaptation occurs which improves recognition accuracy and for trajectory 2, ensures the trajectory is within a specified set of radii. The second trajectory is seen iterating in (a), (b), (c), (d), (e) (f), (g), and (h).



(a) Iteration vs Min Within Radii Cost

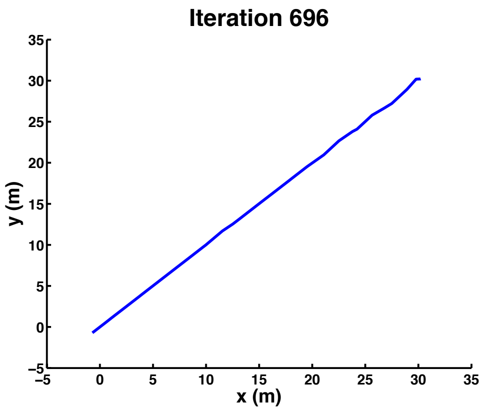


(b) Iteration vs Total Cost

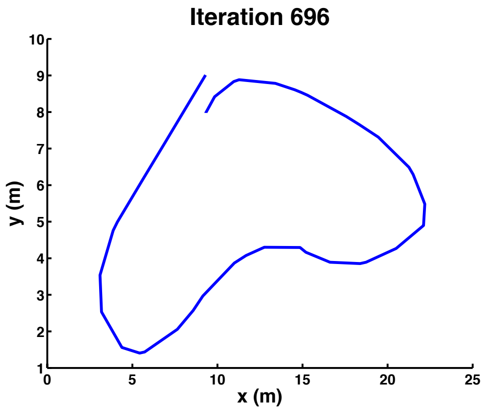
Figure 44: Experiment 5 Iteration vs Various Costs

require that the sequential points in the trajectory produce a strict circle. Only that the points individually are within a specified radius. It is interesting that starting from a straight line can produce a circle loiter. Although it is not a strict circle it does accomplish the desirable properties of a circle loiter for an autonomous vehicle such as maintain motion so as not to fall out of the sky or sink to the bottom while within a small task space. If the purpose of this circle loiter was to maintain an orthogonal sensor towards a target in the center of the loiter, then the objective cost would increase the cost of the circle loiter produced in iterations 661 and 696 because of the dip inwards seen at the bottom of those loiters.

Conclusion The combination of minimizing the distance between the first and last point (DFirstLast) and points within specified radii (Radii) can produce a circle loiter like trajectory starting from a straight line. The circle loiter produced is not a perfect circle due to the nature of the cost function chosen.



(a) Trajectory 1.



(b) Trajectory 2.

Figure 45: The final set of trajectories to be recognized at iteration 696 in experiment 5. Both trajectory one and two were initialized as straight lines. In order to reduce recognition error while creating a circle loiter-like trajectory, the TAR Framework adapted trajectory two while maintaining vehicle dynamics, keeping with a specified area, keeping within two radii, and minimizing the distance between starting and stoping locations. Trajectory one and two at iteration 696 are seen in (a) and (b), respectively.

6.8.6 Mark The Spot

Previous work, presented in Chapter 5, has demonstrated the use of autonomous vehicle trajectories to communicate pertinent information to a teammate during interrupted communication. In particular, we have demonstrated the use of a figure 8 loiter to indicate the location of an important object during autonomous team cooperation. The following experiments attempt to automatically create trajectories that concentrate the autonomous vehicle's trajectory over a specific location. Ideally, these trajectories would be figure 8 loiter-like yet that is not necessary. Such results would indicate that the designed cost function is appropriate for maintaining a trajectory with such a task objective.

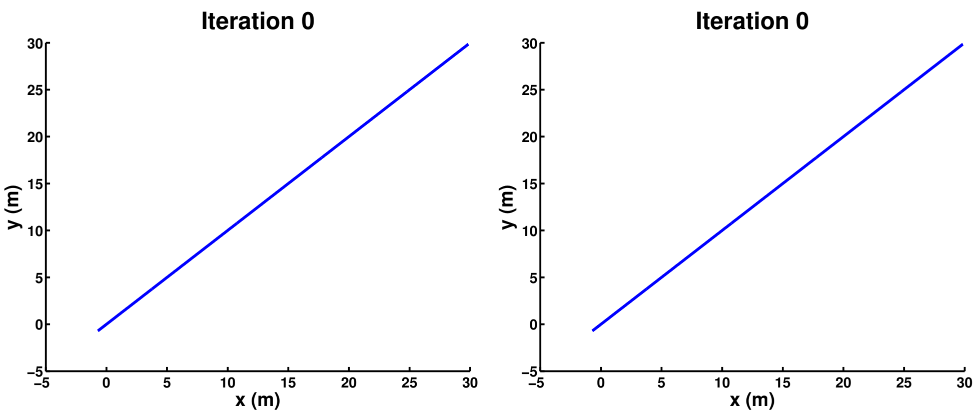
Experiment 6 Setup

Table 27: Experiment 6 Setup

Task	Initial Trajectory	Perturb	Cost Functions
Straight Line	Straight Line	N	N/A
Mark Spot	Straight Line	Y	$DFirstLast + MarkSpot$

# RBFs Per Dim: 12	Initial Sigma: 2

In experiment 6, the TAR framework begins with two straight lines, seen in Figure 46, and adapts one of them for improved recognition and to mark the spot of a location of interest. The experimental setup is contained in Table 27. While the list of cost functions for the adapted trajectory only displays $DFirstLast + MarkSpot$, it does include $R + Act + Area$ but is not displayed for brevity. The trajectories are approximated by 12 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectory being adapted. In order to produce a trajectory that can mark the spot of interest, experiment six introduces a new task objective called Mark the Spot (MarkSpot). MarkSpot, described in Section 6.4, encourages a trajectory to increase the points in a certain descritized location along a trajectory



(a) Trajectory 1. (b) Trajectory 2.

Figure 46: Initial set of trajectories to be recognized for experiment 6. Both trajectory one and two, seen in (a) and (b), have been initialized as straight lines to induce recognition confusion. The TAR Framework will perturb the second trajectory in order to reduce recognition error while maintaining vehicle dynamics, keeping within an area, minimizing the distance between the first and last points, while attempting to mark a spot of interest.

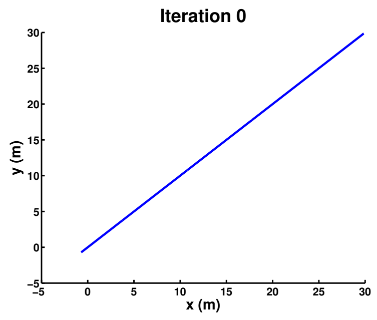
Table 28: Experiment 6 Iteration and Costs

Iteration	0	118	310
MarkSpot	0.113	0.050	0.048
DFirstLast	1.000	0.000	0.000

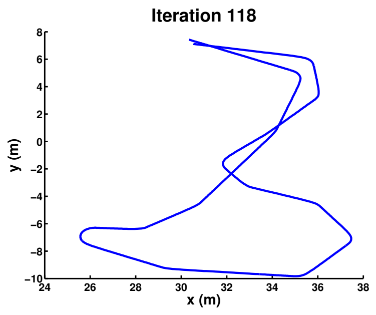
while minimizing the average points throughout the rest of the trajectory locations. It is predicted that such a task objective will produce trajectories with crossings such as in a figure 8 loiter. In this experiment the total cost is defined as,

$$C = R + Act + Area + DFirstLast + MarkSpot. \tag{34}$$

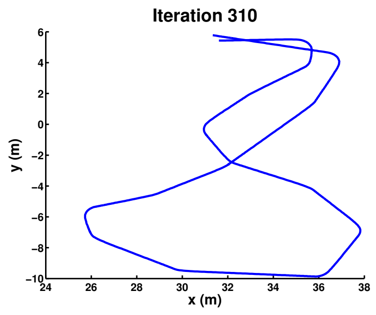
Results The evolution of the trajectory is seen in Figure 47. Figure 48 displays various figures of iteration vs various costs. Notice that the introduction of the MarkSpot cost results do form a trajectory similar to a figure 8 when starting from a straight line. However, the starting and ending locations, while close, are not oriented in such a way that repetition is easy. In fact, the starting and ending location orientations are in opposite directions. Table 28 contains the costs for MarkSpot and DFirstLast per highlighted iterations. The DFirstLast cost starts at a value



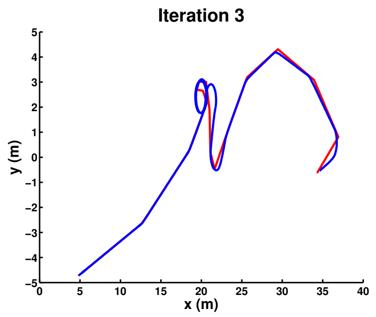
(a) Iteration 0.



(b) Iteration 118.

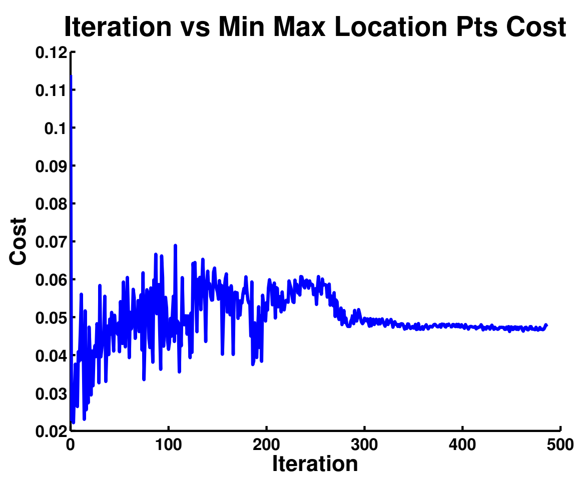


(c) Iteration 310.

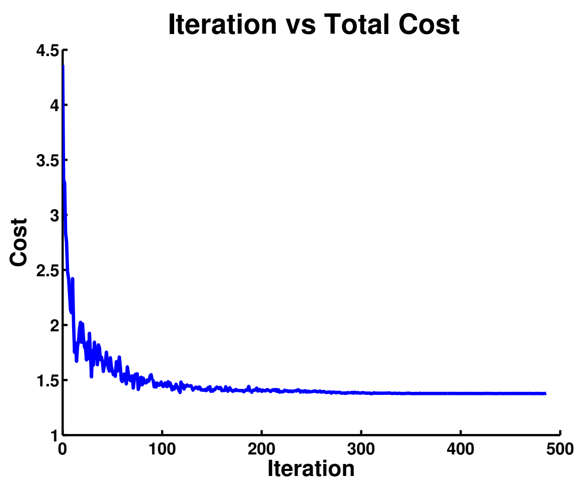


(d) Best Mark Spot.

Figure 47: Experiment 6 results. The TAR framework begins with two straight line trajectories, seen in (a). The framework improves recognition accuracy and adapts trajectory 2 ensuring the trajectory is maximizing the points in one location of the trajectory while lowering the average points in the rest of the trajectory. It is hoped this will produce a trajectory that will mark the spot of a location of interest. The second trajectory is seen evolving in (a), (b), and (c). The best MarkSpot score is achieved in (d) even though it is due to an tracking error.



(a) Iteration vs Mark Spot Cost

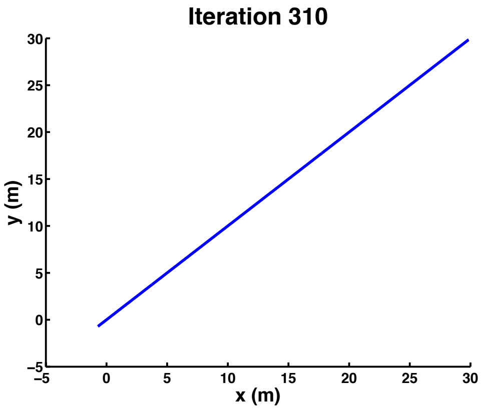


(b) Iteration vs Total Cost

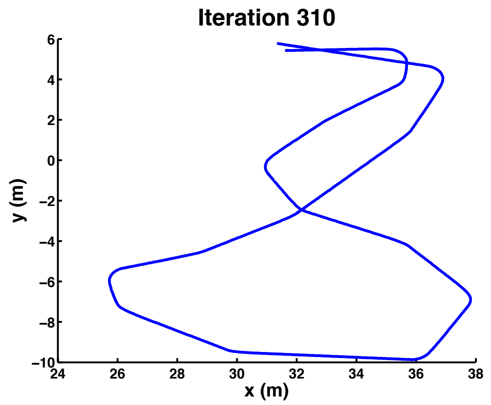
Figure 48: Experiment 6 Iterations vs Various Costs

of 1 and then proceeds to zero as the trajectories produced begin to loop so that the starting and ending locations are close. The MarkSpot cost is seen starting at 0.1139 for iteration 0 and reduced to 0.04806 at iteration 310. The lowest MarkSpot cost occurred during iteration 3 with a value of 0.02216 in which a loop occurred in a specific location driving the max points high and the average points elsewhere low as the rest of the trajectory is roughly straight line segments, seen in Figure 47d. However, the circle at the spot was a combination of vehicle dynamics and autonomy tracking error between the desired trajectory and the commanded one with an actionable (Act) cost of 1. The final recognition set of trajectories is seen in Figure 49.

Discussion The MarkSpot cost can produce a figure 8 loiter-like trajectory starting from a straight line. Although it must be noted that this objective cost, as it is implemented, can produce trajectories that have multiple crossings, as is seen in both Figure 47b and Figure 47c. This indicates that a team of cooperative agents will need more than just an intersection in a trajectory to mark the location of interest. Repeatedly, the results demonstrate the importance of including the actionable (Act) task objective, as seen in Figure 47d. As the commanded trajectory is in red and the



(a) Trajectory 1.

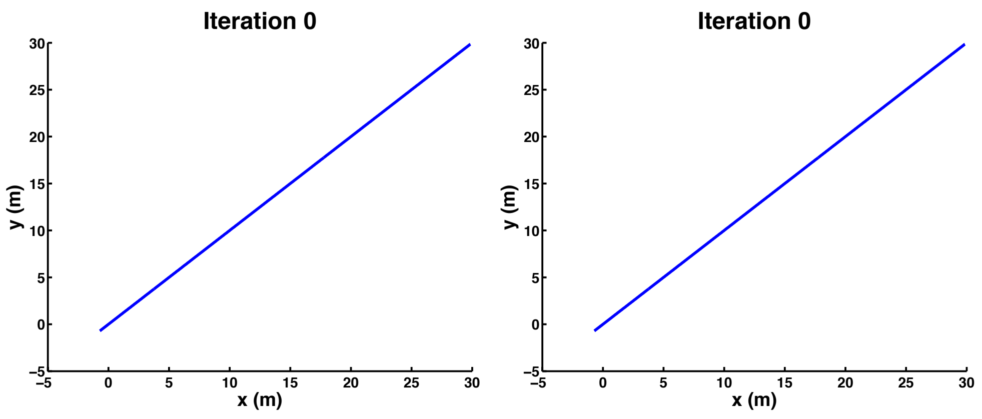


(b) Trajectory 2.

Figure 49: The final set of trajectories to be recognized at iteration 310 in experiment 6. Both trajectory one and two were initialized as straight lines. In order to create a trajectory that indicates a location of interest, the TAR Framework adapted trajectory two to reduce recognition error while maintaining vehicle dynamics, keeping within an area, minimizing the distance between first and last locations while attempting to mark the spot. Trajectory one and two at iteration 310 are seen in (a) and (b), respectively.

tracked trajectory is in blue indicate the trajectory maximizing the one location the best out of the possible trajectories was created out of tracking error. Subsequently, the Act objective cost was high and that trajectory was not included in the viable set by cost. While the DFirstLast cost ensured that the first and last point in the trajectory were close for ease of repetition, the orientation of the autonomous vehicle at those locations were opposite and therefore the vehicle would need to circle around to initiation a repetition. A new objective will need to be introduced so that the orientations are as similar as possible to ensure smooth and repeatable trajectories. Such an objective may also aid in reducing the last two intersections that occur towards the top of iteration 310, seen in Figure 47c.

Conclusion The MarkSpot cost can produce trajectories that are figure 8 loiter-like when starting from a straight line. A new task objective should be introduced to ensure that the starting and ending orientations of the autonomous vehicle are similar to ensure repetition without much correction.



(a) Trajectory 1. (b) Trajectory 2.

Figure 50: Initial set of trajectories to be recognized for experiment 7. Both trajectory one and two, seen in (a) and (b), have been initialized as straight lines to induce recognition confusion. The TAR Framework will perturb the second trajectory in order to reduce recognition error while maintaining vehicle dynamics, keeping within an area, minimizing the distance and orientation difference between the first and last locations, while attempting to mark a spot of interest.

Experiment 7 Setup

Table 29: Experiment 7 Setup

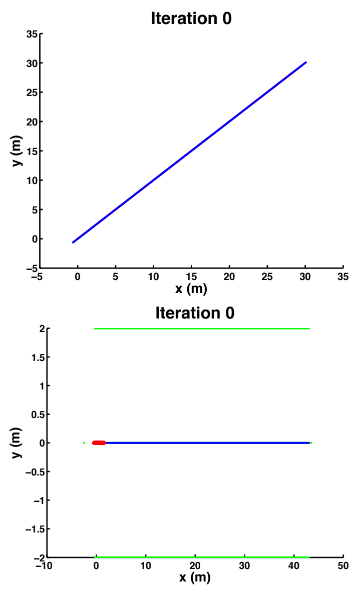
Task	Initial Trajectory	Perturb	Cost Functions
Straight Line	Straight Line	N	N/A
Mark Spot	Straight Line	Y	$DFirstLast + MarkSpot + MinTheta$

# RBFs Per Dim: 12	Initial Sigma: 2

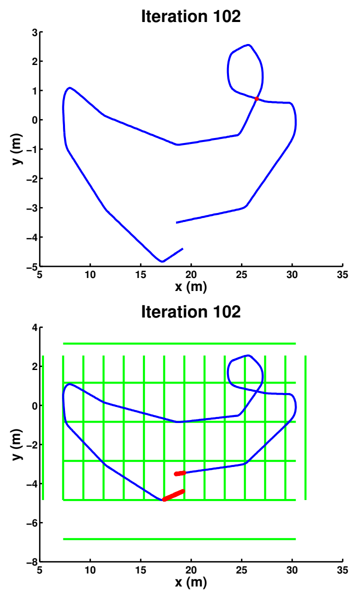
In experiment 7, the TAR framework begins with two straight lines, seen in Figure 50, and adapts one of them for improved recognition and to mark the spot of a location of interest. The experimental setup is contained in Table 29. While the list of cost functions for the adapted trajectory only displays $DFirstLast + MarkSpot + MinTheta$, it does include $R + Act + Area$ but is not displayed for brevity. The trajectories are approximated by 12 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectory being adapted. In order to correct the differing orientations of the initial and final locations of the autonomous vehicle from the previous experiment, a new task objective is introduced which minimizes the theta difference between the first and last orientations ($MinTheta$). The total cost is the same as the previous experiment except with the addition of $MinTheta$,

$$C = R + Act + Area + DFirstLast + MarkSpot + MinTheta. \tag{35}$$

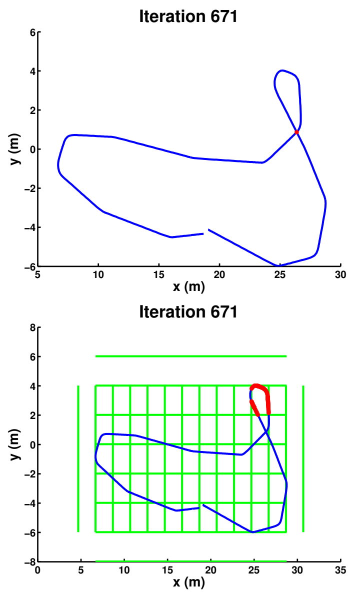
As previously, it is predicted that the MarkSpot cost will create a trajectory in which the autonomous vehicle visits a specific location more than the rest of the trajectory which will be figure 8 loiter-like. The inclusion of the task objective to minimize the distance between the start and end locations will allow for producible repetitions. The new task objective for this experiment, MinTheta, should encourage the TAR Framework to produce a trajectory in which the autonomous vehicle will begin and end a trajectory with the same orientation which will aid in trajectory repetitions.



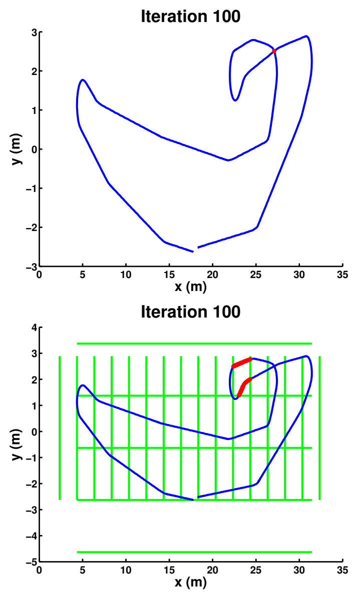
(a) Iteration 0.



(b) Iteration 102.



(c) Iteration 671.



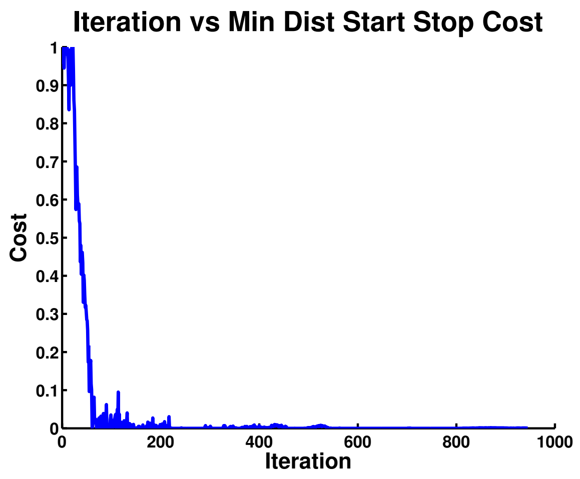
(d) Best Theta Start Stop.

Figure 51: Experiment 7 results. The TAR framework begins with two straight line trajectories. The framework adapts the second trajectory to improve recognition accuracy and ensures the trajectory is maximizing the points in one location of the trajectory while lowering the average points in the rest of the trajectory. It is hoped this will produce a trajectory that will mark the spot of a location of interest. The top row illustrates the followed trajectory while the bottom row illustrates the discretization in green and the cell with the max points in red which is used to calculate the MarkSpot cost. The second trajectory is seen iterating in (a), (b), and (c). The most similar starting and ending orientation is seen in (d).

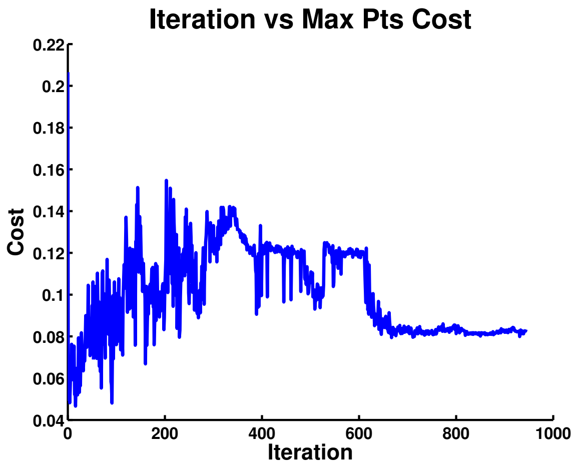
Results The evolution of the trajectory can be seen in Figure 51. The top row demonstrates the best trajectory produced by each highlighted iteration while the bottom row displays the results of the MarkSpot cost with the discretizations in green and the points of the cell with the max points are in red. The trajectories demonstrate that the inclusion of both the DFirstLast and the new MinTheta costs create trajectories with starting and ending points that are close together and have similar orientations, as was predicted. Figure 52 contains figures displaying iteration vs various costs. As seen in Figures 52b and 52c the costs of minimizing the distance between the starting and stopping locations (DFirstLast) and minimizing the difference in the starting and stoping orientations (MinTheta) quickly drop off to close to zero by iteration 300. The task objective which varies is the MarkSpot cost, as seen in Figure 52b. The MarkSpot objective begins to plateau around iteration 700. The final set of trajectories for recognition is seen in Figure 53.

Discussion It is evident that the inclusion of the MinTheta task objective does indeed create starting and stoping orientations that are similar. The inclusion of both DFirstLast and MinTheta has produced a trajectory that is easily repeatable without needing to travel much. The important thing to note is that it was predicted that the MarkSpot task objective would produce a figure 8 loiter-like trajectory. Which in this case it does resemble a figure 8 loiter. What it lacks is that the intersection is in the middle of the trajectory. Additionally, it was predicted that the location of the maximum points of the trajectory would be at an intersection. This is not the case as can be seen in Figure 51. For example, the maximum descritized locations in iteration 671, seen in Figure 51c is the top looping portion of the trajectory instead of the intersection.

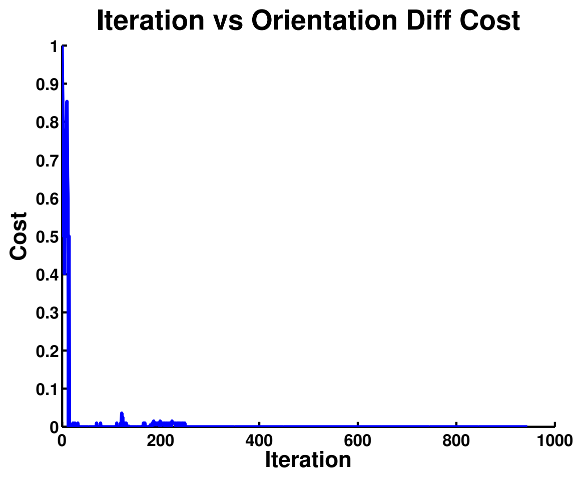
Conclusion The inclusion of MinTheta task objective does produce trajectories in which the orientation between the starting and ending position of the trajectory are similar.



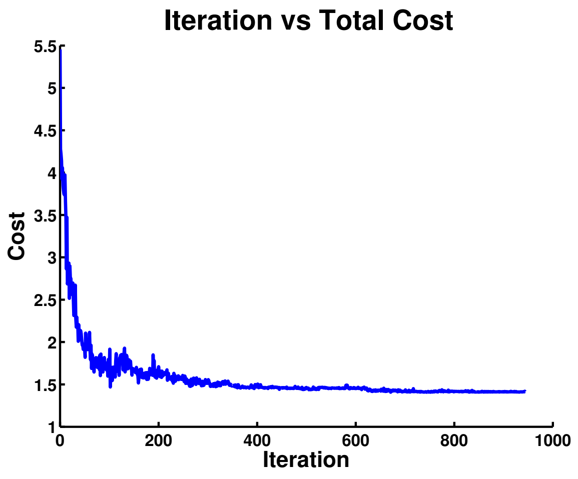
(a) Iteration vs Min Dist Start Stop Cost



(b) Iteration vs MarkSpot Cost

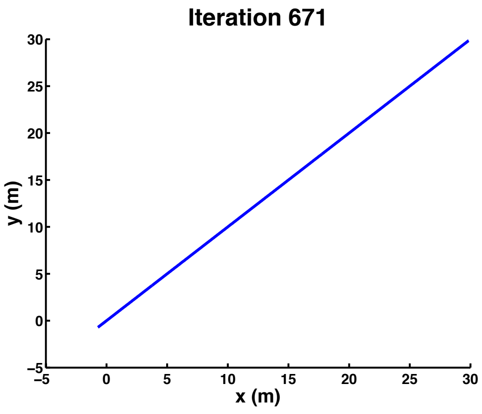


(c) Iteration vs MinTheta Cost

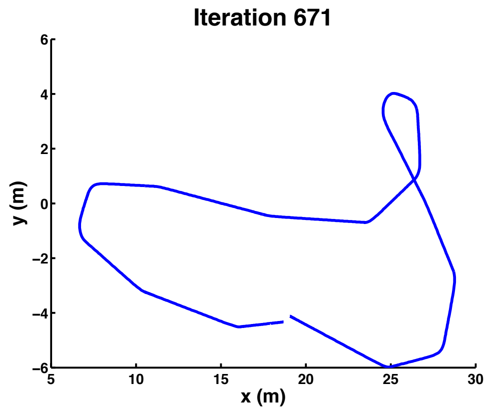


(d) Iteration vs Total Cost

Figure 52: Experiment 7 Iterations vs Various Costs

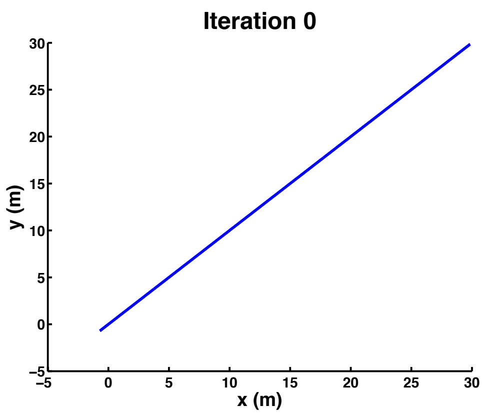


(a) Trajectory 1.

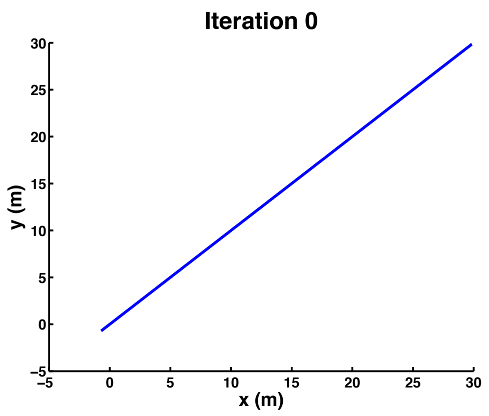


(b) Trajectory 2.

Figure 53: The final set of trajectories to be recognized at iteration 671 in experiment 7. Both trajectory one and two were initialized as straight lines. In order to create a trajectory that indicates a location of interest, the TAR Framework adapted trajectory two to reduce recognition error while maintaining vehicle dynamics, keeping within a specified area, minimizing the distance and orientation difference between the first and last location while attempting to mark the spot. Trajectory one and two at iteration 671 are seen in (a) and (b), respectively.



(a) Trajectory 1.



(b) Trajectory 2.

Figure 54: Initial set of trajectories to be recognized for experiment 8. Both trajectory one and two, seen in (a) and (b), have been initialized as straight lines to induce recognition confusion. The TAR Framework will perturb the second trajectory in order to reduce recognition error while maintaining vehicle dynamics, keeping within an area, minimizing the distance and orientation difference between the first and last locations, while attempting to mark a spot of interest.

Experiment 8 Setup

Table 30: Experiment 8 Setup

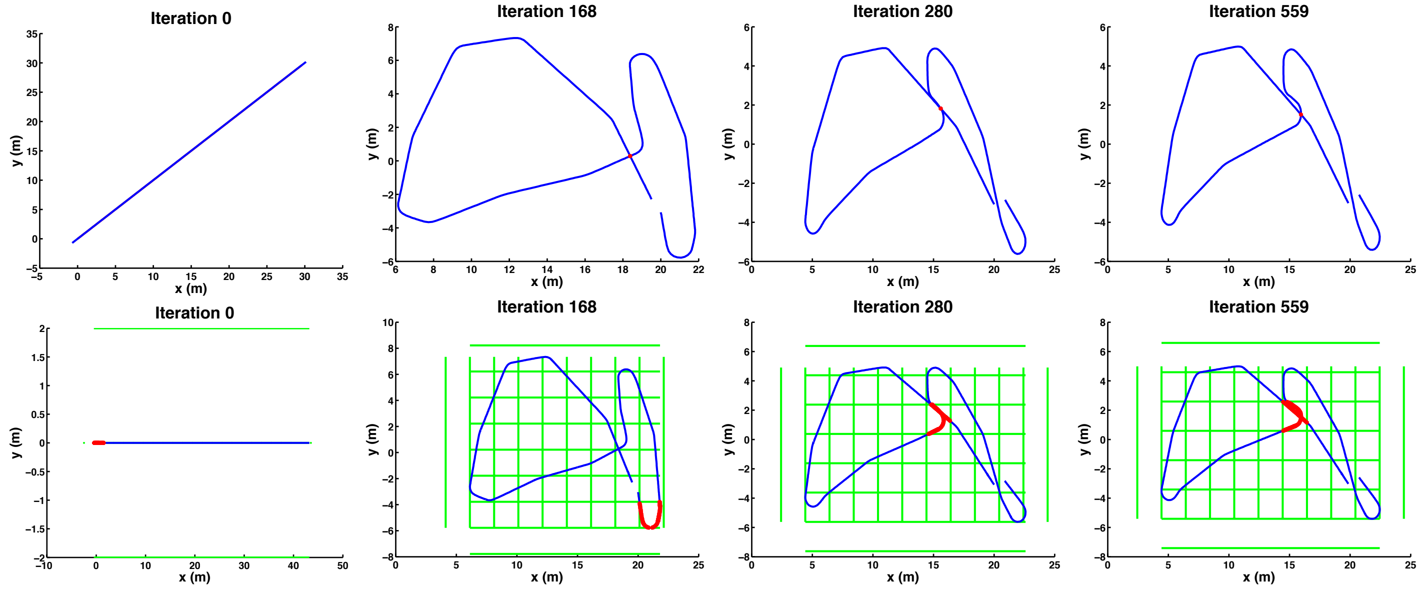
Task	Initial Trajectory	Perturb	Cost Functions
Straight Line	Straight Line	N	N/A
Mark Spot	Straight Line	Y	$DFirstLast + MarkSpot + MinTheta$
# RBFs Per Dim: 12		Initial Sigma: 2	

In experiment 8, the TAR framework begins with two straight lines, seen in Figure 54, and adapts one of them for improved recognition and to mark the spot of a location of interest. The experimental setup is contained in Table 30. While the list of cost functions for the adapted trajectory only displays $DFirstLast + MarkSpot + MinTheta$, it does include $R + Act + Area$ but is not displayed for brevity. The trajectories are approximated by 12 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectory being adapted. The cost for the

TAR Framework is the exact same as the previous experiment,

$$C = R + Act + Area + DFirstLast + MarkSpot + MinTheta. \tag{36}$$

This combination of task objectives is predicted to create a trajectory in which a location in the trajectory will be highlighted and that the start and ending location will be close with similar orientation.



(a) Iteration 0. (b) Iteration 168. (c) Iteration 280. (d) Iteration 559.

Figure 55: Experiment 8 results. The TAR framework starts with two straight lines and adapts the second trajectory ensuring the trajectory is maximizing the points in one location of the trajectory while lowering the average points in the rest of the trajectory. It is hoped this will produce a trajectory that will mark the spot of a location of interest. The second trajectory is seen iterating in (a), (b), (c), and (d). The top row of the figures are the trajectories produced by the autonomous vehicle while the bottom row are the corresponding descritizations by the MarkSpot cost in which the cell with the most trajectory locations are indicated in red.

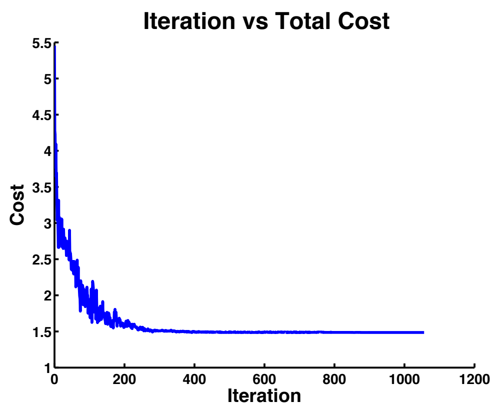
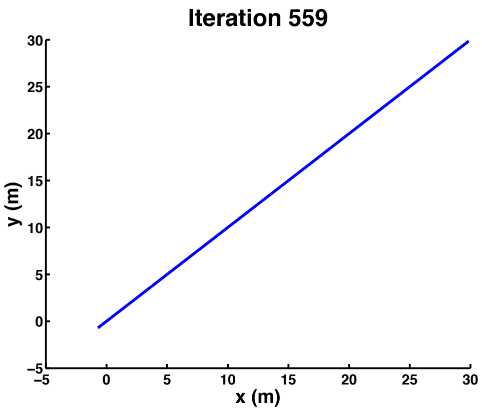


Figure 56: Experiment 8 Iteration vs Total Cost

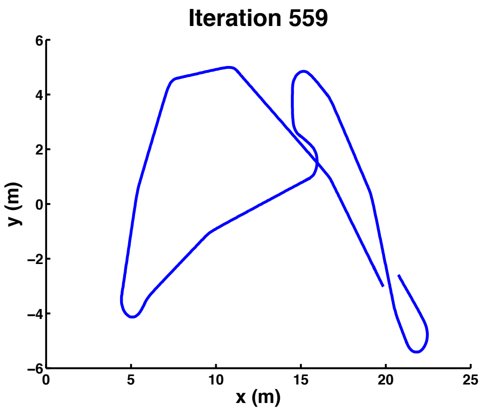
Results The evolution of the trajectory can be seen in Figure 55. The top row contains the best, minimum cost, trajectory produced at highlighted iterations while the bottom row contains the MarkSpot task objective with the discretization of the trajectory displayed in green and the cell with the maximum points displayed in red. In this trial of MarkSpot, a trajectory that is very similar to figure 8 loiter is produced by iteration 168, seen in Figure 55b. By iteration 280 the desired intersection in the trajectory becomes the location with the most points, as seen in Figure 55c. However, this iteration has introduced a half loop which is undesirable as there will be two intersections if the trajectory is repeated. Figure 56 displays the cost per iterations and it demonstrates that the total cost levels out around iteration 300. The final set of trajectories for recognition are seen in Figure 57.

Discussion The MarkSpot cost is able to create a figure 8 loiter-like trajectory. The inclusion of both MinTheta and DFirstLast costs does create the desired starting and stopping distance along with similar orientations which is conducive for repetitive trajectories. However, in this trial, the trajectories with lower cost starting at iteration 280 all the way to iteration 559 have introduced a half loop which will create a second trajectory intersection if repeated. This is an artifact of the TAR Framework not checking repetitions of a trajectory for unwanted features.

Conclusion Experiments 6, 7, and 8 have demonstrated the importance of the



(a) Trajectory 1.



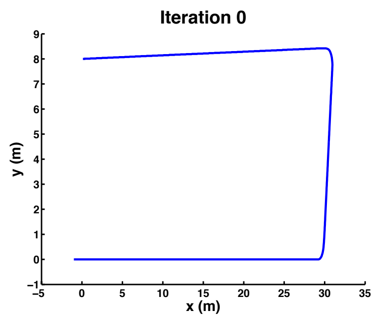
(b) Trajectory 2.

Figure 57: The final set of trajectories to be recognized at iteration 559 in experiment 8. Both trajectory one and two were initialized as straight lines. In order to create a trajectory that indicates a location of interest, the TAR Framework adapted trajectory two to reduce recognition error while maintaining vehicle dynamics, keeping within a specified area, minimizing the distance and orientation difference between the first and last location while attempting to mark the spot. Trajectory one and two at iteration 559 are seen in (a) and (b), respectively.

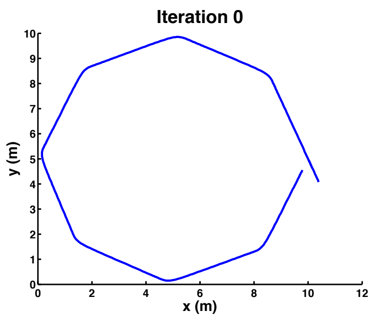
various task objectives. By itself, the MarkSpot cost can produce figure 8 loiter-like trajectories. However, the starting and ending location and orientations may not be conducive to repetitive trajectories in which it is desirable for the trajectory to easily repeat as soon as it terminates. Therefore, introducing the objectives DFirstLast and MinTheta encourage the TAR Framework to produce trajectories that have starting and ending locations with close proximity and similar orientations. It is important to note that the stochastic nature of the search algorithm does not guarantee a figure 8 loiter-like trajectory. A strategy is to run multiple copies of the optimization and choose the one that produces the best results.

6.8.7 Common AUV Mission: Many Circle Loiters

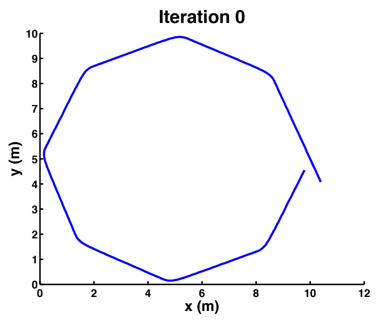
This section explores the ability of the TAR Framework to simultaneously optimize trajectories for recognition and task objectives in more complicated scenarios. These experiments involve a common mission encountered by autonomous underwater vehicles (AUVs) in which it performs a search mission with circle loiters performed while the vehicle is waiting for commands, in an error state, or upon mission completion.



(a) Trajectory 1.



(b) Trajectory 2.



(c) Trajectory 3.

Figure 58: Initial set of trajectories to be recognized for experiment 9. Trajectory one, seen in (a), has been initialized as a search leg. Trajectories two and three, seen in (b) and (c), have been initialized as circle loiters to induce recognition confusion and reflect a common scenario where a circle loiter is performed for multiple AUV states. The TAR Framework will perturb the third trajectory in order to reduce recognition error but keeping a circle loiter-like trajectory by maintaining vehicle dynamics, keeping within an area, minimizing the distance between the first and last locations, while staying within a set of radii.

Experiment 9 Setup

Table 31: Experiment 9 Setup

Task	Initial Trajectory	Perturb	Cost Functions
Search	Search	N	N/A
Circle Loiter 1	Circle Loiter	N	N/A
Circle Loiter 2	Circle Loiter	Y	$DFirstLast + w(Radii)$

# RBFs Per Dim: 6	Initial Sigma: 2

In this experiment an autonomous vehicle performs a search mission in which it performs circle loiter during states of error and mission completion. In this setup, the TAR framework only modifies the second circle loiter trajectory with the total cost defined as,

$$C = R + Act + Area + DFirstLast + w(Radii). \tag{37}$$

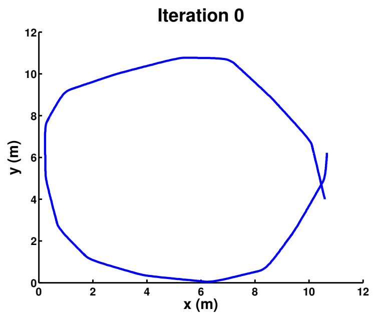
Previously attempted circle loiter adaptations failed to maintain a circle loiter shape. In order to prioritize maintaining a circle loiter-like trajectory a priority weighting function was applied to the Radii cost, which is indicated with $w(Radii)$. It is predicted that a combination of a priority weighted Radii cost will favor maintaining

Table 32: Experiment 9 Iteration and Costs

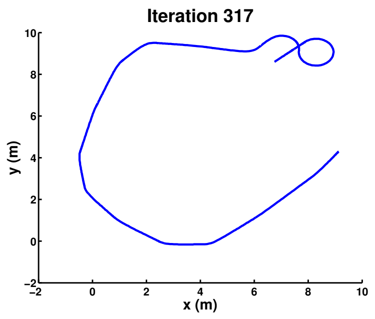
Iteration	0	317	345
Rec	2.598	1.811	1.811
Radii	0.000	0.001	0.001
Total Cost	4.656	3.961	3.96

a circle loiter-like trajectory and that recognition error would help perturb the second circle loiter within. In experiment 9, the TAR framework begins with a search trajectory and two circle loiter trajectories, seen in Figure 58, and adapts only one of them for improved recognition and to ensure it maintains a circle loiter-like trajectory. The experimental setup is contained in Table 31. While the list of cost functions for the adapted trajectory only displays $DFirstLast + w(Radii)$, it does include $R + Act + Area$ but is not displayed for brevity. The trajectories are approximated by 6 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectory being adapted.

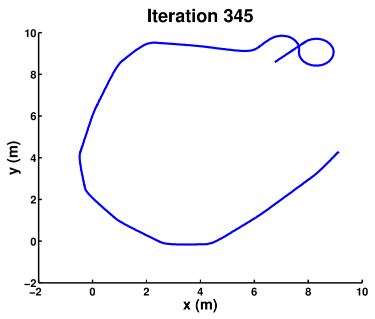
Results The evolution of the second circle loiter is seen in Figure 59. The corresponding iterations and task objective costs are seen in Table 32. The initial circle loiter is seen in Figure 59a and the perturbations are seen in Figure 59b and Figure 59c for iterations 317 and 345, respectively. Figure 61 displays various costs per iterations. An important task objective cost to observe are the recognition error which starts at 2.598 at the start and reduces to 1.811 by iteration 317. The corresponding confusion matrices per highlighted iterations are seen in Table 33. Although recognition accuracy improves from the initial iteration, the recognition accuracy does not rise above 80% for any one of the labels. As the desired property is that the perturbed trajectory maintain a circle loiter-like pattern the next important task objective cost is the priority weighted Radii which begins at 0 and ends at 0.0007741 at iteration 345. The worst Radii objective cost was produced in iteration 1, seen in Figure 59e, with a weighted value of 7,741. The final set of trajectories for recognition are seen



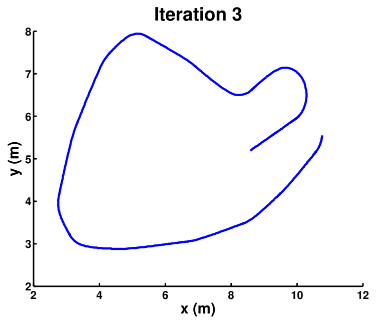
(a) Iteration 0.



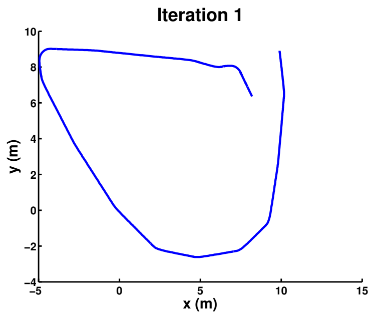
(b) Iteration 317.



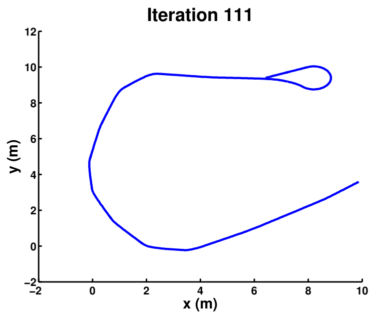
(c) Iteration 345.



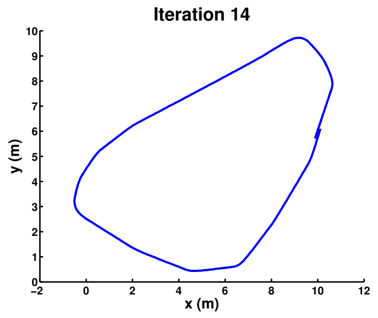
(d) Best Rec Cost.



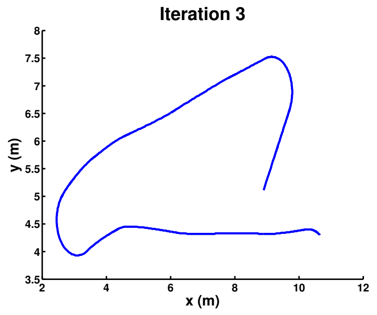
(e) Worst Radii Cost.



(f) Best Radii Cost.

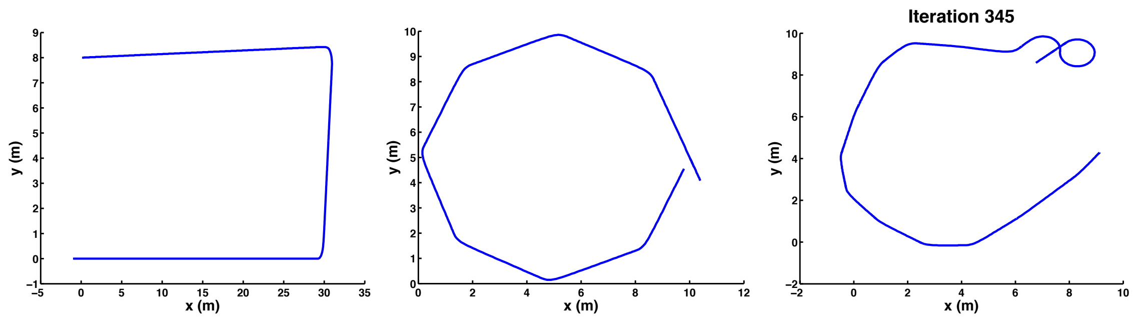


(g) Best Dist Start Stop.



(h) Best Area Score.

Figure 59: Experiment 9 results. The TAR framework starts with a search pattern and two identical circle loiters. The framework adapts the second circle loiter while improving recognition accuracy and ensuring the trajectory is within a specified set of radii. The second circle loiter is seen iterating in (a), (b), and (c). The best recognition error is seen in (d). The worst and best Raddi cost are seen in (e) and (f), respectively. The best starting and stopping distance is seen in (g). The variation with the lowest area cost is seen in (h).



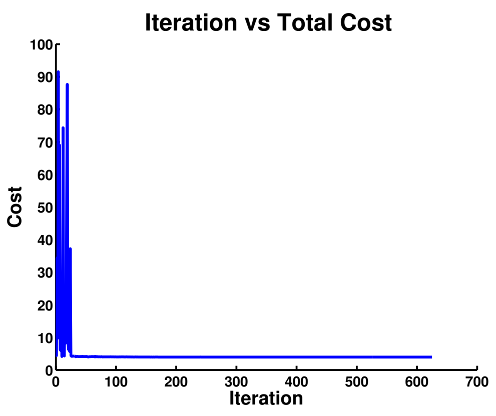
(a) Trajectory 1. (b) Trajectory 2. (c) Trajectory 3.

Figure 60: The final set of trajectories to be recognized at iteration 345 in experiment 9. Trajectory one was a static search leg, as seen in (a), throughout the experiment. Trajectories two and three were initialized as circle loiters with trajectory two remaining static, as seen in (b), while trajectory three was perturbed by the TAR Framework, seen in (c). Trajectory three was perturbed to reduce recognition error while maintaining a circle loiter-like trajectory by maintaining vehicle dynamics, keeping within a specified area, minimizing the distance between the first and last locations, while staying with a set of radii.

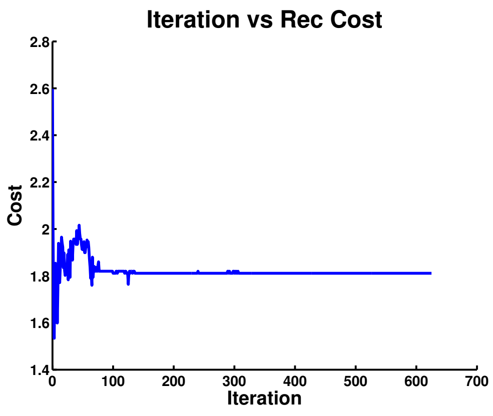
in Figure 60.

Discussion The use of a priority weighted Radii cost is successful in maintaining a circle loiter-like trajectory when optimizing among a more complicated scenario of one search pattern and two circle loiters. The recognition score is improved as desired yet still preferring a circle loiter-like trajectory. Note that the task objectives of DFirstLast or MinTheta are only clamped between 0 and 1. By not using priority weighting functions for DFirstLast or MinTheta, it allows the TAR Framework to flex within a circle loiter-like radius but not require the first and last position to be close nor have similar orientations. Unfortunately, the overall recognition accuracy for any one of the labels does not rise above 80%.

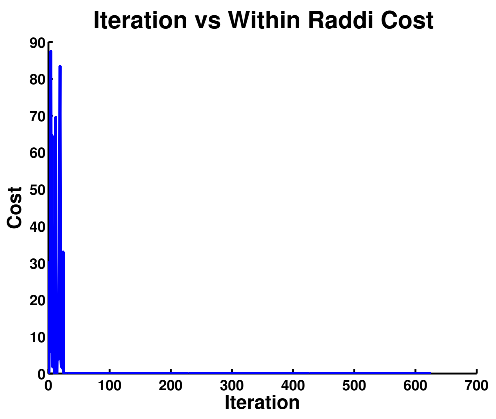
Conclusion Using a priority weighted Radii task objective did keep the perturbed trajectory within a circle loiter-like radius. Recognition accuracy did improve yet non of the labels improved beyond 80%.



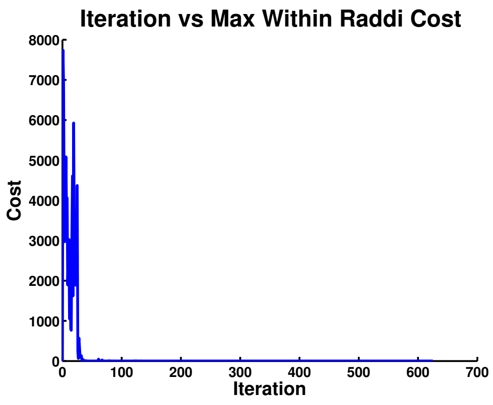
(a) Iteration vs Total Cost



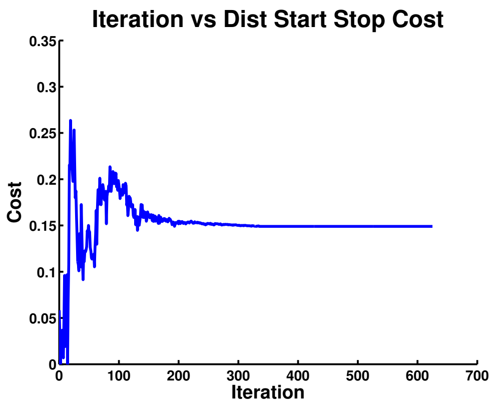
(b) Iteration vs Rec Cost



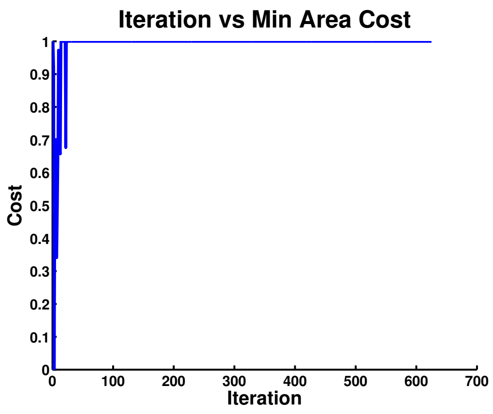
(c) Iteration vs Within Radii Cost



(d) Iteration vs Max Within Radii Cost



(e) Iteration vs Min Dist Start Stop Cost



(f) Iteration vs Area Cost

Figure 61: Experiment 9 Iterations vs Various Costs

Table 33: Experiment 9 Confusion Matrices

		Label			
		NoLabel	Search	Circle Loiter 1	Circle Loiter 2
Actual	*Transition*	23.00	33.30	57.50	6.90
	Search	0.00	74.80	11.70	13.60
	Circle Loiter 1	0.00	65.10	11.60	23.30
	Circle Loiter 2	0.00	0.00	48.50	51.50

(a) Iteration 0

		Label			
		NoLabel	Search	Circle Loiter 1	Circle Loiter 2
Actual	*Transition*	3.49	11.60	20.90	64.00
	Search	0.00	75.70	3.88	20.40
	Circle Loiter 1	0.00	0.00	62.80	37.20
	Circle Loiter 2	0.00	0.00	23.10	76.90

(b) Iteration 317

		Label			
		NoLabel	Search	Circle Loiter 1	Circle Loiter 2
Actual	*Transition*	3.49	11.60	20.90	64.00
	Search	0.00	75.70	3.88	20.40
	Circle Loiter 1	0.00	0.00	62.80	37.20
	Circle Loiter 2	0.00	0.00	23.10	76.90

(c) Iteration 345

Experiment 10 Setup

Table 34: Experiment 10 Setup

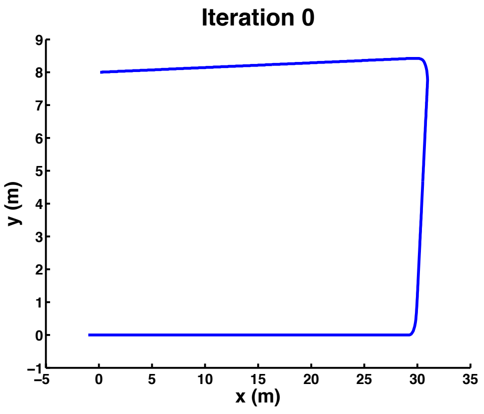
Task	Initial Trajectory	Perturb	Cost Functions
Search	Search	N	N/A
Circle Loiter 1	Circle Loiter	N	N/A
Circle Loiter 2	Circle Loiter	Y	$w(DFirstLast) + w(Radii)+$ $w(MinTheta)$
Circle Loiter 3	Circle Loiter	Y	$w(DFirstLast) + w(Radii)+$ $w(MinTheta)$

# RBFs Per Dim: 6	Initial Sigma: 2

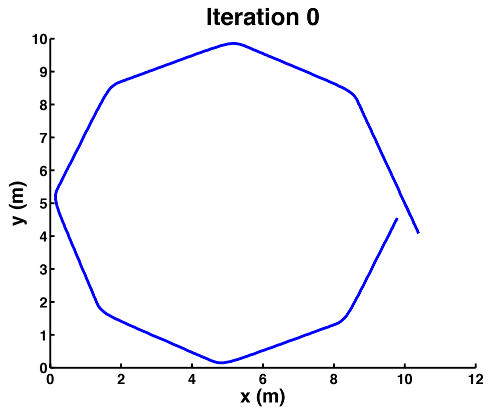
The autonomous vehicle performing a search scenario is made more challenging by including three circle loiter modes: loiter for commands, error state, and mission completion for a total of four trajectory labels. The TAR Framework modifies two of the circle loiters with both tasks having the same objectives of *Radii*, $DFirstLast$, $MinTheta$, and *Area*. The total cost function is defined as,

$$C = R + Act_3 + Area_3 + w(DFirstLast_3) + w(Radii_3) + w(MinTheta_3)+ \\ Act_4 + Area_4 + w(DFirstLast_4) + w(Radii_4) + w(MinTheta_4). \quad (38)$$

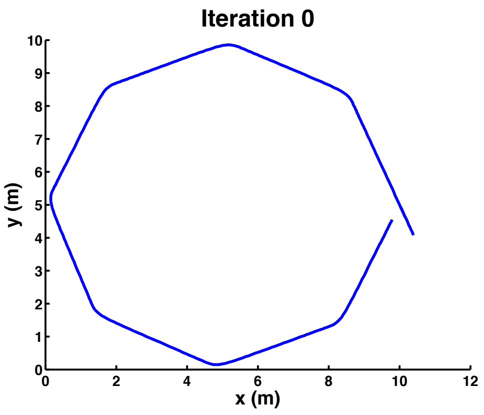
The subscript for each cost corresponds to the id number of the trajectory. For example, $Raddi_3$ corresponds to the *Raddi* cost for the third trajectory which in this experiment is Circle Loiter 2. In order to ensure that the two trajectories maintain the most semblance to circle loiter, a priority weighting function of $w(x) = 15,000x^6$ is applied to the *Radii*, $DFirstLast$, and $MinTheta$ costs individually. This should restrict the trajectories from opening up from their circle, such as was seen in the previous experiment. In experiment 10, the TAR framework begins with a search trajectory and three circle loiter trajectories, seen in Figure 62, and adapts two of them for improved recognition and to ensure they maintain a circle loiter-like trajectory. The experimental setup is contained in Table 34. While the list of cost functions for the adapted trajectories only displays $DFirstLast + Radii + MinTheta$, they do



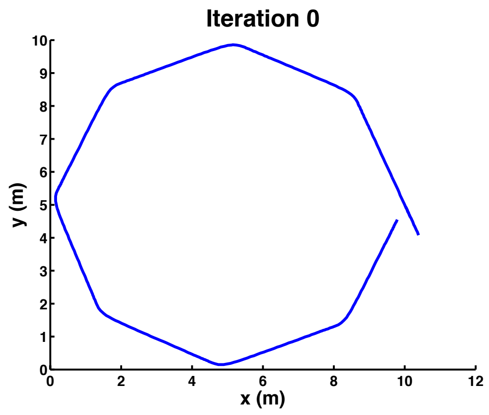
(a) Trajectory 1.



(b) Trajectory 2.

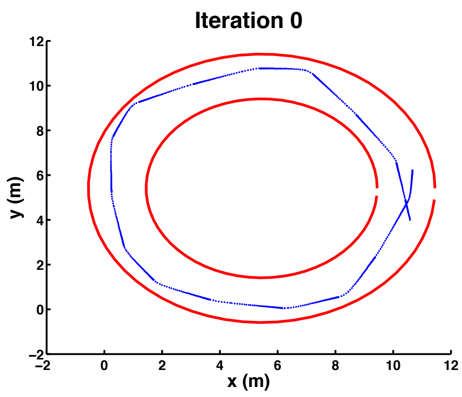


(c) Trajectory 3.

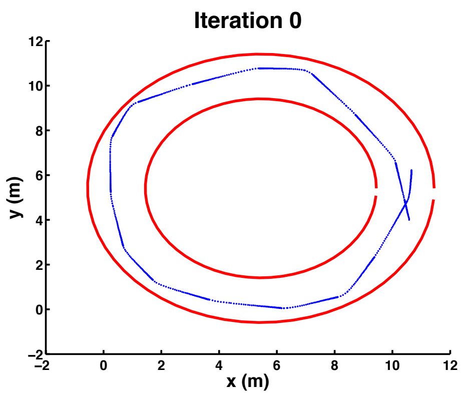


(d) Trajectory 4.

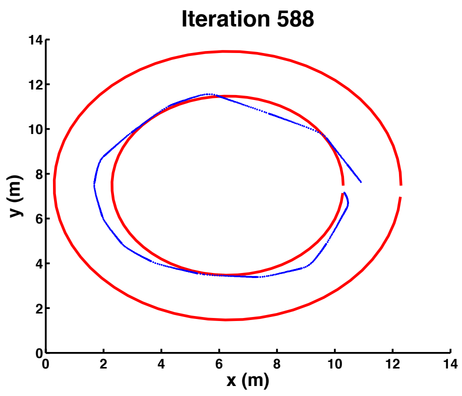
Figure 62: Initial set of trajectories to be recognized for experiment 10. Trajectory one, seen in (a), has been initialized as a search leg. Trajectories two, three, and four, seen in (b), (c), and (d), respectively, have been initialized as circle loiters to induce recognition confusion and reflect a common scenario where a circle loiter is performed for multiple AUV states. The TAR Framework will perturb the third and fourth trajectories in order to reduce recognition error while maintaining circle loiter-like trajectories by maintaining vehicle dynamics, keeping within an area, minimizing the distance and orientation difference between the first and last locations, while staying within a set of radii.



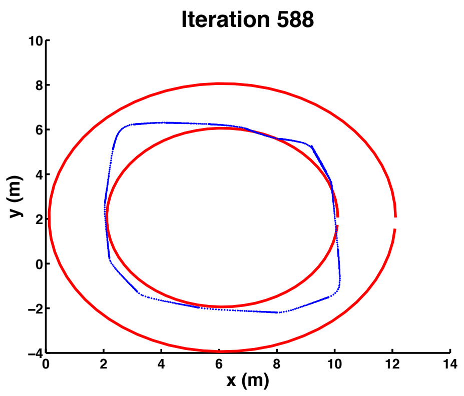
(a) CircleLoiter2 Iteration 0.



(b) CircleLoiter3 Iteration 0.



(c) CircleLoiter2 Iteration 588.



(d) CircleLoiter3 Iteration 588.

Figure 63: Experiment 10 results. The TAR framework starts with a search pattern and three identical circle loiters. The framework improves recognition accuracy while ensuring that circle loiters two and three maintain circle loiter-like trajectories. Circle loiter two can be seen adapting in (a) and (c). Circle loiter three can be seen adapting in (b) and (d).

include $R + Act + Area$ but they are not displayed for brevity. The trajectories are approximated by 6 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectories being adapted.

Results The evolution of the two circle loiters are seen in Figure 63. Circle loiter two is seen in Figures 63a and 63c for iterations 0 and 588, respectively. Circle loiter three is seen in Figures 63b and 63d for iterations 0 and 588, respectively. Qualitatively it can be seen that both modified circle loiters have starting and ending positions that are close together and have similar orientations. Circle loiter two appears to favor the inner specified radius, seen in Figure 63c. Circle loiter three appears to have slight bulges at the upper left and lower right corners, seen in Figure 63d. Figure 64 displays the total cost per iteration of the experiment. Table 35 contains the confusion matrices for iteration 0 and 588. As can be seen, the recognition accuracy for the search trajectory remains at 74.8%. Unfortunately, the recognition accuracy for the circle loiter one remains below 12% starting at 11.6% at the beginning and even becoming worse at iteration 588 with an accuracy of 6.98%. Interestingly, the circle loiter two improves in recognition accuracy starting at 51.5% to 69.6% by iteration 588. The trajectory with the best recognition accuracy improvement is circle loiter three. It begins at 0% recognition accuracy and improves to 85.7%. The final set of trajectories for recognition is seen in Figure 65.

Discussion The TAR Framework is capable of increasing the recognition accuracy in a complicated search mission with three types of circle loiters by modifying two circle loiters. Introducing priority weighting functions for the Radii, DFirstLast, and MinTheta does keep the trajectories produced by the TAR Framework to resemble circle loiters when the initial trajectories start as circle loiters. While the recognition accuracy is improved, only one of the trajectory labels improves above 80%. The first circle loiter continues to have a recognition accuracy below 15% regardless of improvement. The second and third circle loiters do improve their recognition accuracy. A possible way to improve the recognition accuracy of the first circle loiter is to additionally include it for adaptation by the TAR Framework. Ultimately though, it is a complicated scenario which pushes recognition accuracy when the three circle loiters are desired to maintain similar shapes to their initial configuraiton.

Conclusion The TAR Framework can improve the recognition accuracy of trajectories performed in a complicated search mission with three possible circle loiters. Using priority weighting functions on the Radii, DFirstLast, and MinTheta task objectives does keep the trajectories being modified close to a circle loiter-like shape.

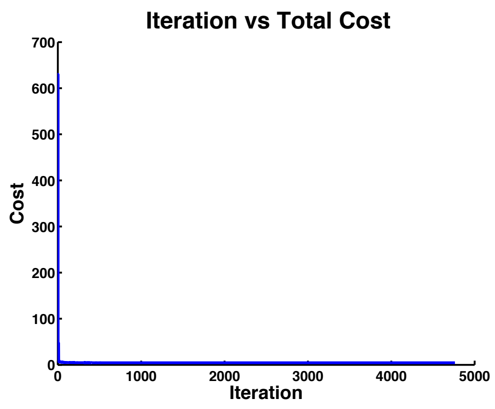


Figure 64: Experiment 10 Iteration vs Total Cost

Unfortunately, the recognition accuracy never improves beyond 12% for the first circle loiter. This may indicate a limit in this implementation of the TAR Framework or it may indicate that including the first circle loiter as a trajectory that can be adapted is necessary to have overall improvement.

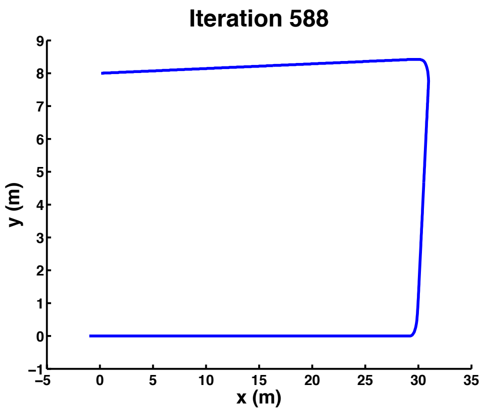
Table 35: Experiment 10 Confusion Matrices

		Label				
		NoLabel	Search	Circle Loiter 1	Circle Loiter 2	Circle Loiter 3
Actual	*Transition*	1.89	36.80	55.70	5.66	0.00
	Search	0.00	74.80	11.70	13.60	0.00
	Circle Loiter 1	0.00	65.10	11.60	23.30	0.00
	Circle Loiter 2	0.00	48.50	0.00	51.50	0.00
	Circle Loiter 3	0.00	0.00	72.70	27.30	0.00

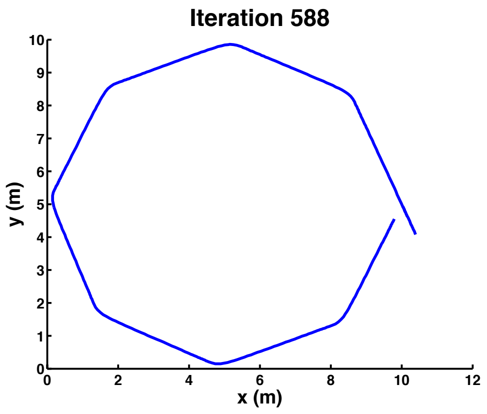
(a) Iteration 0

		Label				
		NoLabel	Search	Circle Loiter 1	Circle Loiter 2	Circle Loiter 3
Actual	*Transition*	3.81	7.62	4.76	67.60	16.20
	Search	0.00	74.80	0.00	10.70	14.60
	Circle Loiter 1	0.00	0.00	6.98	58.10	34.90
	Circle Loiter 2	0.00	0.00	8.70	69.60	21.70
	Circle Loiter 3	0.00	0.00	0.00	14.30	85.70

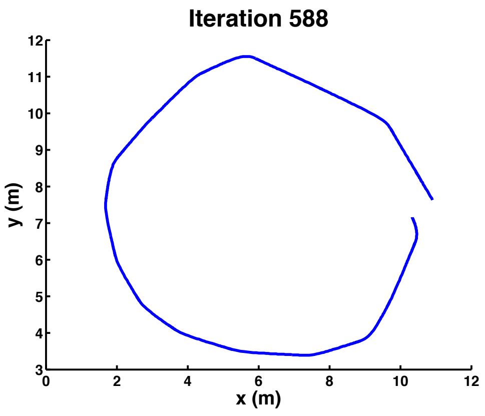
(b) Iteration 588



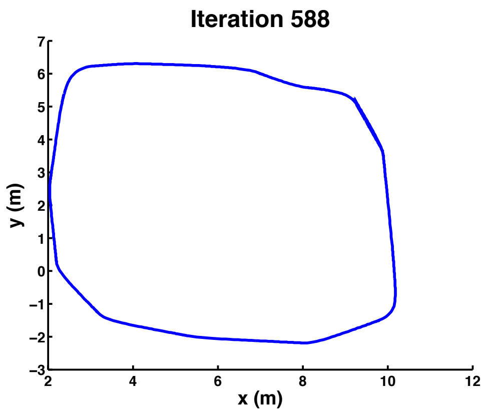
(a) Trajectory 1.



(b) Trajectory 2.



(c) Trajectory 3.



(d) Trajectory 4.

Figure 65: The final set of trajectories to be recognized at iteration 588 in experiment 10. Trajectory one was a static search leg, as seen in (a), throughout the experiment. Trajectories two, three, and four were initialized as circle loiters with trajectory two remaining static, as seen in (b). Trajectories three and four were perturbed by the TAR Framework, seen in (c) and (d), respectively, to reduce recognition error while attempting to maintain circle loiter-like trajectories. This was achieved by maintaining vehicle dynamics, keeping within a specified area, minimizing the distance and orientation difference between the first and last locations, while staying within a set of radii.

Experiment 11 Setup

Table 36: Experiment 11 Setup

Task	Initial Trajectory	Perturb	Cost Functions
Search	Search	N	N/A
Circle Loiter 1	Circle Loiter	N	N/A
Circle Loiter 2	Circle Loiter	Y	$w(DFirstLast) + w(Radii) + w(MinTheta)$
Circle Loiter 3	Circle Loiter	Y	$w(DFirstLast) + w(Radii) + w(MinTheta)$

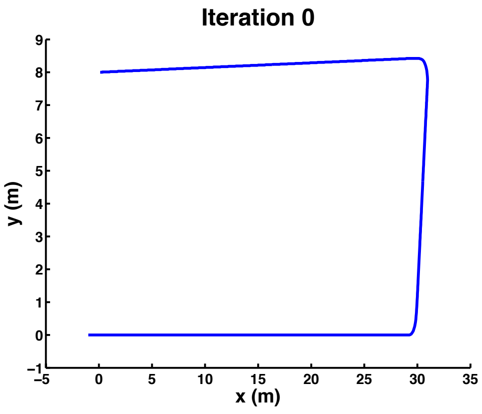
# RBFs Per Dim: 6	Initial Sigma: 2

This experiment is an exact replication of the previous one. In experiment 11, the TAR framework begins with a search trajectory and three circle loiter trajectories, as seen in Figure 66, and adapts two of them for improved recognition and to ensure they maintain a circle loiter-like trajectory. The experimental setup is contained in Table 36. While the list of cost functions for the adapted trajectories only displays $w(DFirstLast) + w(Radii) + w(MinTheta)$, they do include $R + Act + Area$ but they are not displayed for brevity. The total cost function is defined as,

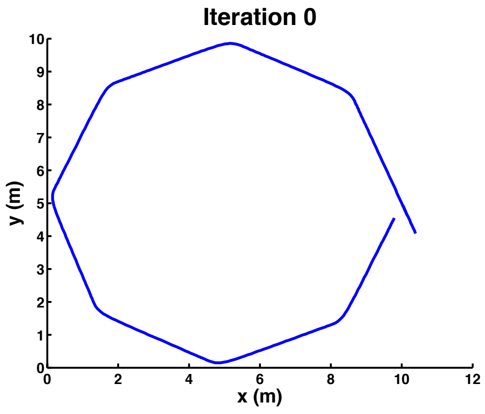
$$C = R + Act_3 + Area_3 + w(DFirstLast_3) + w(Radii_3) + w(MinTheta_3) + \\ Act_4 + Area_4 + w(DFirstLast_4) + w(Radii_4) + w(MinTheta_4). \quad (39)$$

The subscript for each cost corresponds to the id number of the trajectory. For example, $Raddi_3$ corresponds to the $Raddi$ cost for the third trajectory which in this experiment is Circle Loiter 2. A priority weighting function of $w(x) = 15,000x^6$ is applied to the $Radii$, $DFirstLast$, and $MinTheta$ costs individually. The trajectories are approximated by 6 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectories being adapted.

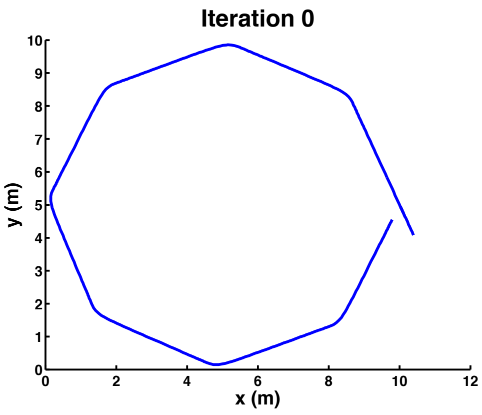
Results The evolution of the two circle loiters can be seen Figure 67. Circle loiter two is seen in Figures 67a and 67c for the initial and iteration 1000, respectively.



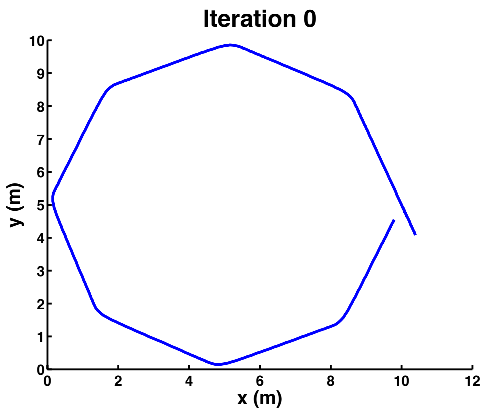
(a) Trajectory 1.



(b) Trajectory 2.

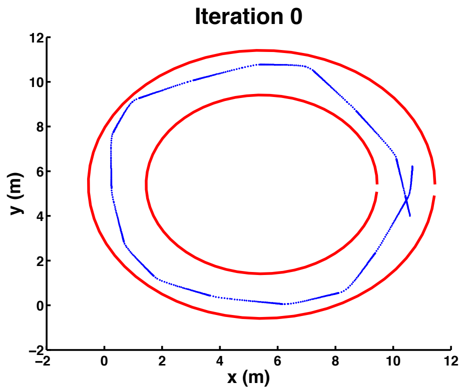


(c) Trajectory 3.

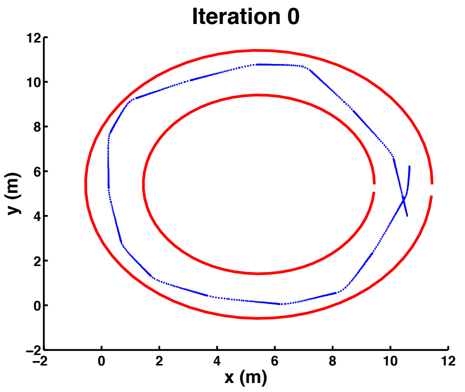


(d) Trajectory 4.

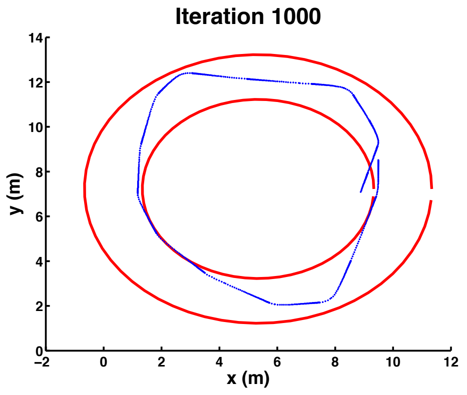
Figure 66: Initial set of trajectories to be recognized for experiment 11. Trajectory one, seen in (a), has been initialized as a search leg. Trajectories two, three, and four, seen in (b), (c), and (d), respectively, have been initialized as circle loiters to induce recognition confusion and reflect a common scenario where a circle loiter is performed for multiple AUV states. The TAR Framework will perturb the third and fourth trajectories in order to reduce recognition error while still maintaining circle loiter-like trajectories by maintaining vehicle dynamics, keeping within an area, minimizing the distance and orientation difference between the first and last locations, while staying within a set of radii.



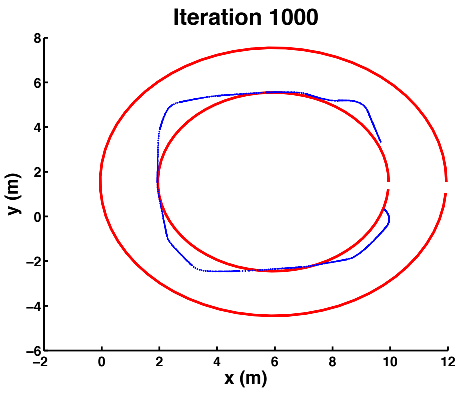
(a) CircleLoiter2 Iteration 0.



(b) CircleLoiter3 Iteration 0.



(c) CircleLoiter2 Iteration 1000.



(d) CircleLoiter3 Iteration 1000.

Figure 67: Experiment 11 results. The TAR framework begins with a search pattern and three circle loiters. The framework improves recognition accuracy while ensuring circle loiters two and three maintain their circle loiter-like trajectories. Circle loiter two can be seen evolving in (a) and (c). Circle loiter three can be seen evolving in (b) and (d).

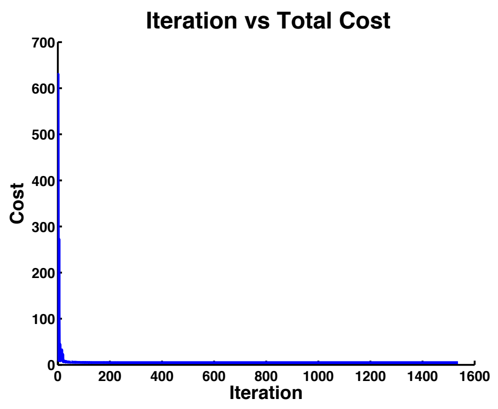


Figure 68: Experiment 11 Iteration vs Total Cost

By iteration 1000, the circle loiter two has a bulge at the bottom of the trajectory and a larger one on the top. Circle loiter three is seen in Figures 67b and 67d. By iteration 1000, the circle loiter three tends to hug the inner radius with 4 slight bulges evenly spaced around the trajectory. Figure 68 displays the minimum total cost per each iteration. It can be seen that the initial trajectories have total costs above 600 but quickly reduce onwards. Table 37 contains the confusion matrices for the initial configuration and iteration 1000. The label accuracy for Search increases from 74.8% to 78.6%. In this replicant, the circle loiter one does see a slight improvement in recognition accuracy starting at 11.6% and ending in 18.6%. The recognition accuracy of circle loiter two does improve from 51.5% to 80%. The most dramatic improvement in recognition accuracy occurs with circle loiter three starting at 0% accuracy and increasing to 75% accuracy. The final set of trajectories for recognition are seen in Figure 69.

Discussion

As a replicant of the previous experiment, the slight differences in the results demonstrate the stochastic nature of this implementation of the TAR Framework. As previously, the priority weighted Radii, DFirstLast, and MinTheta costs ensure that the TAR Framework produces trajectories that still resemble circle loiter-like. An important issue arises as to whether the proper balance of priority weighting functions are utilized. For example, if recognition accuracy improvement was the most important desired outcome then a priority weighting function should be applied to it rather than to the other three circle loiter objectives of Radii, DFirstLast, and MinTheta. This combination would most likely result in trajectories that produce high recognition accuracy yet no longer resemble circle loiters. It is possible that maybe a combination of weighting functions for all the task objectives would keep the trajectories circle loiter-like and obtain higher recognition accuracy.

Conclusion The second replicant having similar results, although slightly better, in recognition accuracy improvement supports that this scenario may be pushing the limits of this implementation of the TAR Framework. Results may be improved by including the first circle loiter in the group of trajectories to be modified or by applying different priority weighting functions to all of the costs.

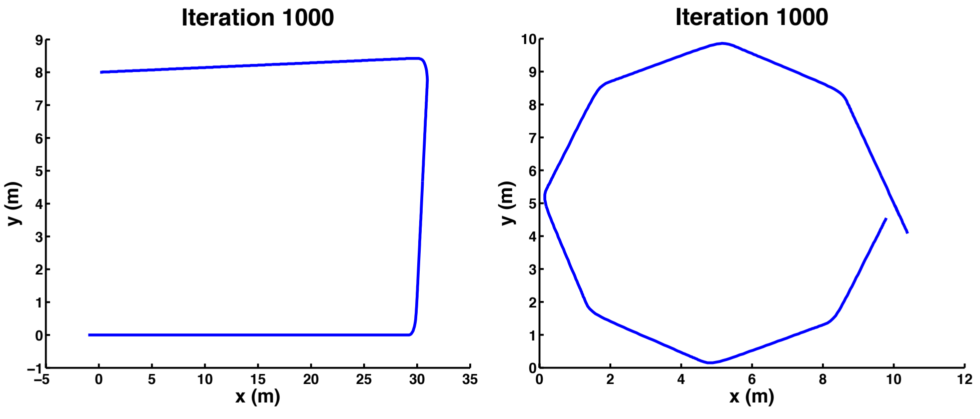
Table 37: Experiment 11 Confusion Matrices

		Label				
		NoLabel	Search	Circle Loiter 1	Circle Loiter 2	Circle Loiter 3
Actual	*Transition*	1.89	36.80	55.70	5.66	0.00
	Search	0.00	74.80	11.70	13.60	0.00
	Circle Loiter 1	0.00	65.10	11.60	23.30	0.00
	Circle Loiter 2	0.00	48.50	0.00	51.50	0.00
	Circle Loiter 3	0.00	0.00	72.70	27.30	0.00

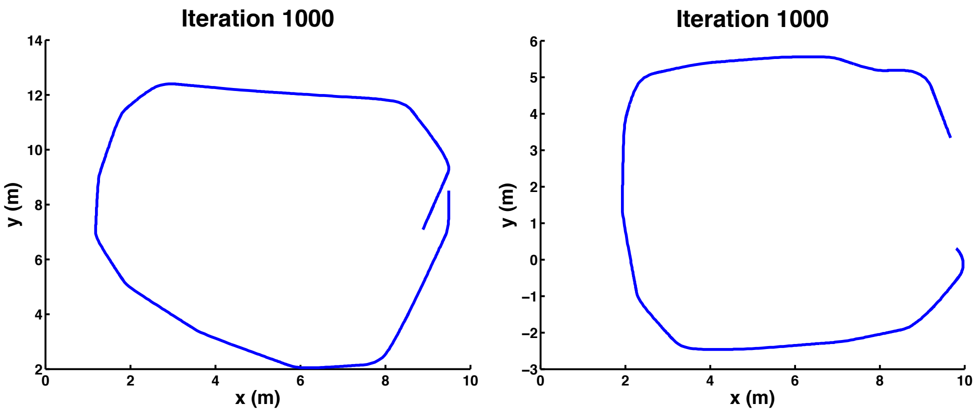
(a) Iteration 0

		Label				
		NoLabel	Search	Circle Loiter 1	Circle Loiter 2	Circle Loiter 3
Actual	*Transition*	3.77	7.55	0	7.55	81.10
	Search	0.00	78.60	0.00	6.80	14.60
	Circle Loiter 1	0.00	0.00	18.60	37.20	44.20
	Circle Loiter 2	0.00	0.00	0.00	80.00	20.00
	Circle Loiter 3	0.00	0.00	0.00	25.00	75.00

(b) Iteration 1000



(a) Trajectory 1. (b) Trajectory 2.



(c) Trajectory 3. (d) Trajectory 4.

Figure 69: The final set of trajectories to be recognized at iteration 1,000 in experiment 11. Trajectory one was a static search leg, as seen in (a), throughout the experiment. Trajectories two, three, and four were initialized as circle loiters with trajectory two remaining static, as seen in (b). Trajectories three and four were perturbed by the TAR Framework, seen in (c) and (d), respectively, to reduce recognition error while attempting to maintain circle loiter-like trajectories. This was achieved by maintaining vehicle dynamics, keeping within a specified area, minimizing the distance and orientation difference between the first and last locations, while staying within a set of radii.

6.8.8 Evolve Common Tasks: Simultaneously

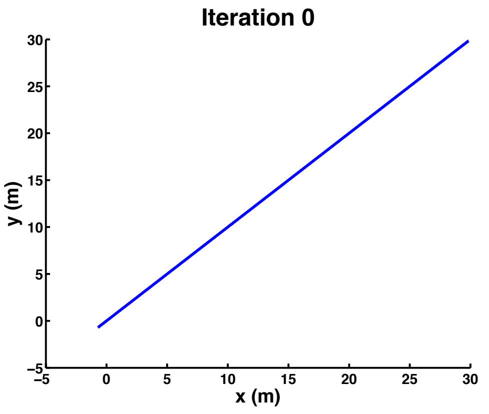
This section presents experiments in which the TAR Framework attempts to simultaneously adapt trajectories that perform four common manned and unmanned vehicle tasks while improving their recognition accuracy. The four common tasks are straight line, search, circle loiter, and figure 8 loiter. The first experiment begins with the trajectories as straight lines and they simultaneously adapt into their respective tasks. The second experiment introduces the concept of individually evolving trajectories for their given tasks separately and then combining them all to continue adapting simultaneously for recognition accuracy improvement. The third experiment starts with each modifiable trajectory as circles and then simultaneously evolving them to their respective tasks.

Experiment 12 Setup

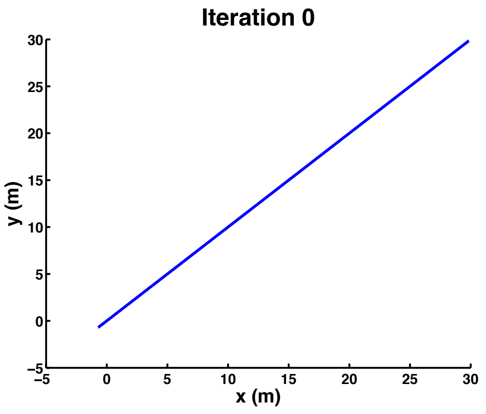
Table 38: Experiment 12 Setup

Task	Initial Trajectory	Perturb	Cost Functions
Straight Line	Straight Line	N	N/A
Search	Straight Line	Y	$Cover$
Circle Loiter	Straight Line	Y	$DFirstLast + Radii + MinTheta$
Mark Spot	Straight Line	Y	$DFirstLast + MarkSpot + MinTheta$
# RBFs Per Dim: 6		Initial Sigma: 5	

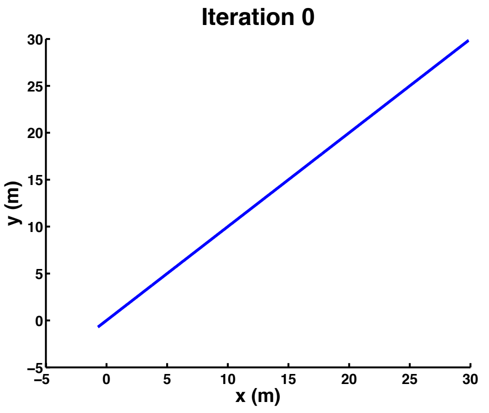
In experiment 12, the TAR Framework simultaneously evolves four common autonomous vehicle tasks starting from four straight lines, seen in Figure 70. Table 38 contains the experimental setup. The trajectories are approximated by 6 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectories being adapted. The first trajectory is GoToWaypoint which does not get modified while the other three which are modified are Search, Circle Loiter, and Mark the Spot. All three trajectories share the basic task objectives of *Recognition*, *Act*, and



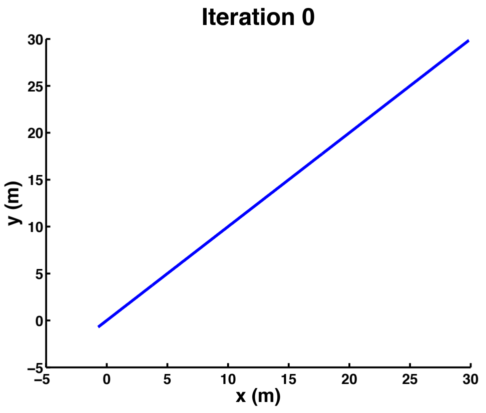
(a) Trajectory 1.



(b) Trajectory 2.



(c) Trajectory 3.



(d) Trajectory 4.

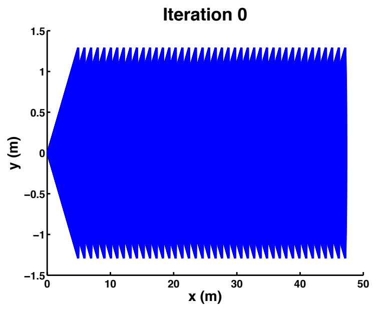
Figure 70: Initial set of trajectories to be recognized for experiment 12. All four trajectories have been initialized as straight lines, as seen in (a), (b), (c), and (d). The TAR Framework will perturb the second, third, and fourth trajectories in order to create trajectories that search, maintain circle loiter-like shape, and mark a spot, respectively. The objectives for all the trajectories is to reduce recognition error while maintaining vehicle dynamics and keeping the trajectories within a specified area. The final objective for trajectory three is maximize search coverage. Because trajectories three and four are designed for repetition, they both include the objectives to minimize the distance and orientation difference between the first and last locations. The final objectives for trajectories three and four are stay within two radii and mark the spot, respectively.

Area, which are omitted from the table for brevity. The search pattern trajectory has the additional task objective of sensor coverage. The circle loiter trajectory has the additional task objectives of DFirstLast, MinTheta, and Radii. The Mark the Spot has the additional task objectives of DFirstLast, MinTheta, and MarkSpot. The task objectives are all clamped between zero and 1 meaning there are no priority weighting functions associated with them. For brevity, the task objectives that are shared in common will have indices to them. The total cost function is defined as,

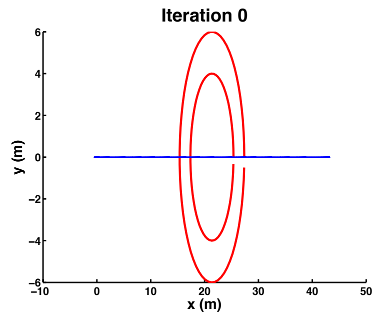
$$\begin{multline} C = R + Act_2 + Area_2 + Cover_2 + Act_3 + Area_3 + DFirstLast_3 + \\ Radii_3 + MinTheta_3 + Act_4 + Area_4 + DFirstLast_4 + \\ MarkSpot_4 + MinTheta_4. \quad (40) \end{multline}$$

The subscript for each cost corresponds to the id number of the trajectory. For example, $Raddi_3$ corresponds to the *Raddi* cost for the third trajectory which in this experiment is Circle Loiter.

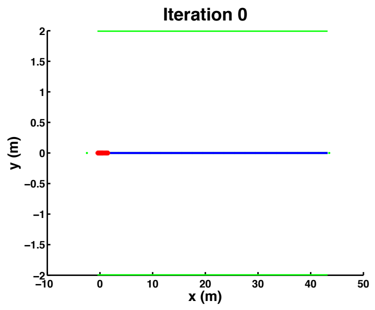
Results The evolution of the trajectories is seen in Figure 71. Qualitatively, the second trajectory is tasked with sensor coverage of an area and progressively improves its area coverage starting evolving from iteration 0, to iteration 178, and ending at iteration 2658, seen Figures 71a, 71d, and 71g, respectively. The third trajectory is tasked with creating a circle loiter-like trajectory and can be seen evolving from iteration 0, 178, and 2658 in Figures 71b, 71d, and71h, respectively. Qualitatively, the third trajectory does not evolve into a circle loiter-like trajectory. However, it does improve the cost of the Radii task objective by ensuring that the points of the trajectory are within the specified radii. The fourth trajectory is tasked with marking the spot or figure 8 loiter and is seen evolving at iterations 0, 178, and 2658 in Figures 71c, 71f, and 71i, respectively. Qualitatively, trajectory four does not appear to be figure 8 loiter-like nor is it visually apparent that any one of the intersections marks the spot of interest. Table 39 contains the confusion matrices for the various



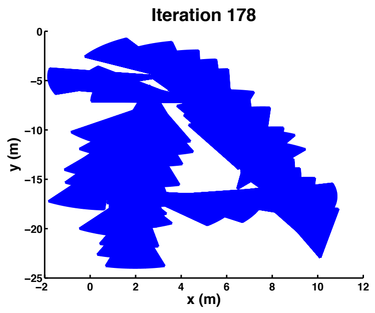
(a) Search Iteration 0.



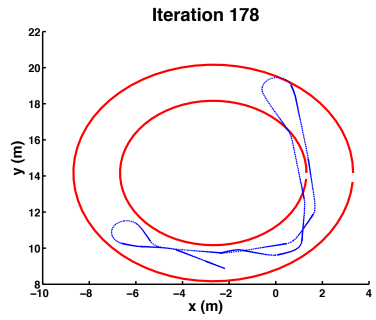
(b) CircleLoiter Iteration 0.



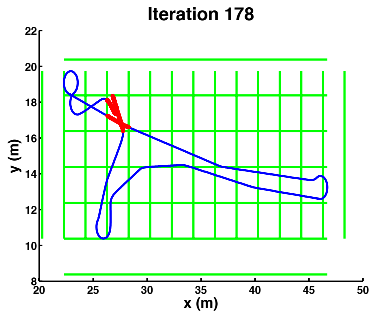
(c) MarkSpot Iteration 0.



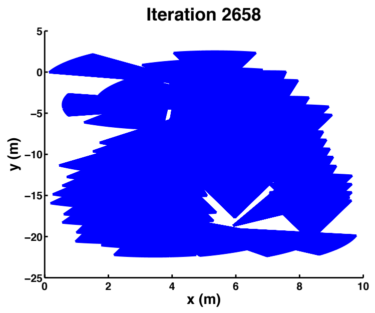
(d) Search Iteration 178.



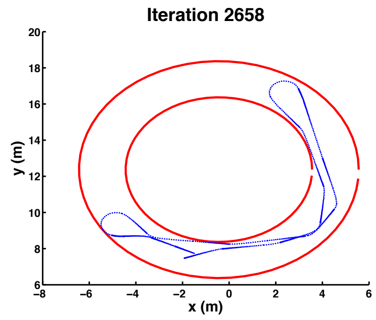
(e) CircleLoiter Iteration 178.



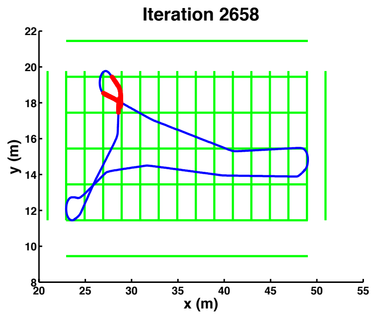
(f) MarkSpot Iteration 178.



(g) Search Iteration 2658.



(h) CircleLoiter Iteration 2658.



(i) MarkSpot Iteration 2658.

Figure 71: Experiment 12 results. The TAR framework begins with four straight line trajectories. The framework improves recognition accuracy while ensuring the task objectives of search, circle loiter, and mark the spot. Trajectory 2 is tasked with sensor coverage and is seen iterating in (a), (d), and (g). Trajectory 3 is tasked with maintaining a circle loiter-like trajectory and is seen iterating in (b), (e), and (h). Trajectory 4 is tasked with marking the spot through and is seen iterating in (c), (f), and (i).

highlighted iterations starting with the initial setup through iterations 178 and 2658. The initial recognition accuracy for all four trajectories is low with GoToWaypoint, Search, Circle Loiter, and Mark the Spot having recognition accuracies of 47.5%, 39.3%, 0%, and 0%, respectively. By iteration 178, the recognition accuracy for trajectory 1, GoToWaypoint, decreases from 47.5% to 20.6%. Trajectory 2, search, also decreased from 39.3% in the initial setup to 12.5%. Trajectory 3, circle loiter, improves from 0% to 32.1%. Trajectory 4, mark the spot, makes a dramatic jump from 0% accuracy to 82.8%. By iteration 2658 the first trajectory reduces its recognition accuracy further to 14.5%. Trajectory 2, search, increases in recognition accuracy to 24.2%. The third trajectory, circle loiter, reduces its recognition accuracy to 22.6%. The fourth trajectory continues to increase its recognition accuracy reaching 84.2%. The final set of trajectories for recognition are seen in Figure 72.

Discussion Unfortunately, starting from straight lines and adapting all four trajectories simultaneously to achieve respective task objectives while also improving recognition accuracy does not seem to qualitatively or quantitatively produce satisfiable results. Visually, the only modified trajectory that seems to accomplish its task is trajectory 2 which is supposed to search an area. The other two modified trajectories did not adapt into circle or figure 8 loiter-like trajectories. Additionally, from a recognition accuracy perspective the results were very low. The only way the recognition accuracy results would be acceptable is if the only required task to recognize well is trajectory 4 at an 80% accuracy rate then it would suffice. Otherwise, the final confusion matrix for iteration 2658 seen in Table 39c demonstrates that the other three trajectories had recognition accuracies below 30%. These results raise the question of whether or not priority weighting functions should have been applied to the different task objectives. In this case, because recognition accuracy was so low a priority weighting function applied to R may improve the results.

Conclusion Attempting to adapt four common tasks simultaneously starting from straight lines to their respective task objectives while also improving recognition accuracy does not produce the desired results. The only task objective that worked well was sensor coverage. The recognition accuracy was below 30% for all but the fourth trajectory which achieved an 84.2% recognition accuracy.

Table 39: Experiment 12 Confusion Matrices

		Label				
		NoLabel	GoToWayPoint	Search	Circle Loiter	Mark Spot
Actual	*Transition*	0.51	98.00	1.53	0.00	0.00
	GoToWayPoint	0.00	47.50	52.50	0.00	0.00
	Search	0.00	60.70	39.30	0.00	0.00
	Circle Loiter	0.00	38.10	61.90	0.00	0.00
	Mark Spot	0.00	70.00	30.00	0.00	0.00

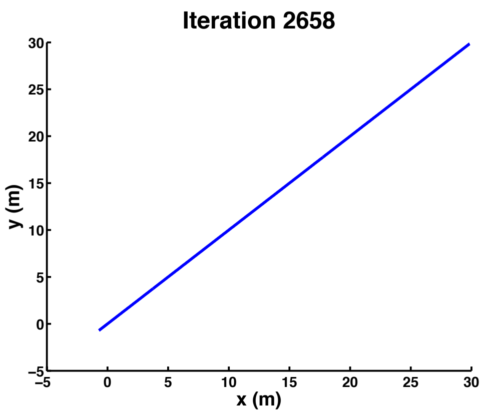
(a) Iteration 0

		Label				
		NoLabel	GoToWayPoint	Search	Circle Loiter	Mark Spot
Actual	*Transition*	4.14	17.20	1.78	51.50	25.40
	GoToWayPoint	3.17	20.60	1.59	1.59	73.00
	Search	0.00	5.68	12.50	12.50	69.30
	Circle Loiter	0.00	25.00	0.00	32.10	42.90
	Mark Spot	0.00	13.80	0.00	3.45	82.80

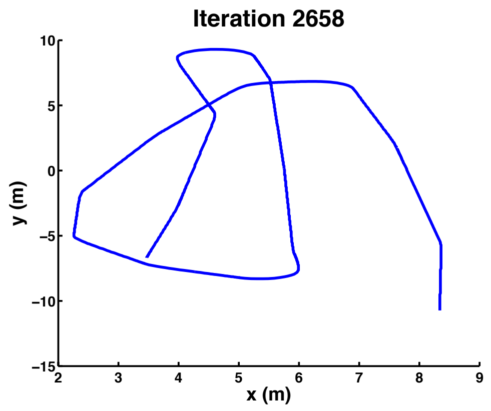
(b) Iteration 178

		Label				
		NoLabel	GoToWayPoint	Search	Circle Loiter	Mark Spot
Actual	*Transition*	4.29	17.20	2.45	55.20	20.90
	GoToWayPoint	8.06	14.50	0.00	3.23	74.20
	Search	0.00	0.00	24.20	14.70	61.10
	Circle Loiter	0.00	16.10	0.00	22.60	61.30
	Mark Spot	0.00	10.50	0.00	5.26	84.20

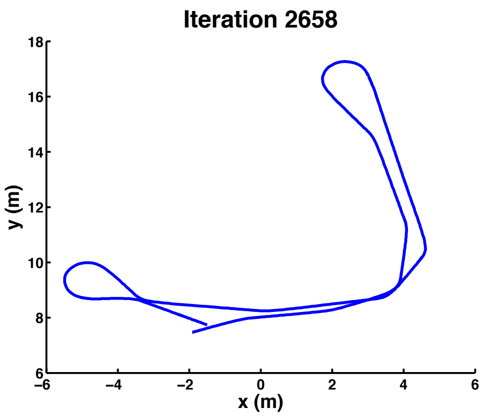
(c) Iteration 2658



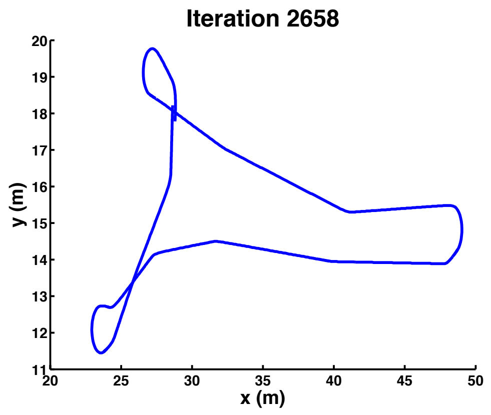
(a) Trajectory 1.



(b) Trajectory 2.



(c) Trajectory 3.



(d) Trajectory 4.

Figure 72: The final set of trajectories to be recognized at iteration 2658 in experiment 12. All four trajectories were initialized as straight lines with trajectory one remaining static, as seen in (a). Trajectory two was tasked with search, seen in (b), trajectory three was tasked with maintaining a circle loiter-like shape, seen in (d), and trajectory four was tasked with marking a spot, seen in (c). The TAR Framework perturbed all of the trajectories with the objectives of reducing recognition error while maintaining vehicle dynamics and performing trajectories within a specified area. Trajectory two's final objective was to maximize sensor coverage. Because trajectory three and four are meant to be repetitive, they both included the objectives of minimizing the distance and orientation difference between the start and end locations. The final objectives of trajectories three and four were stay within two radii and mark the spot, respectively.

Experiment 13

Table 40: Experiment 13 Setup

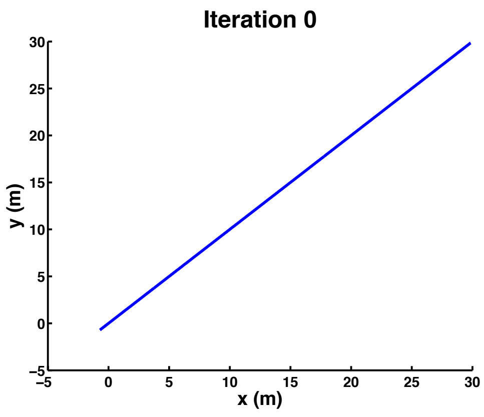
Task	Initial Trajectory	Perturb	Cost Functions
Straight Line	Straight Line	N	N/A
Search	PO Search	Y	$Cover$
Circle Loiter	PO Circle Loiter	Y	$DFirstLast + Radii + MinTheta$
Mark Spot	PO Mark Spot	Y	$DFirstLast + MarkSpot + MinTheta$

# RBFs Per Dim: 12	Initial Sigma: 2

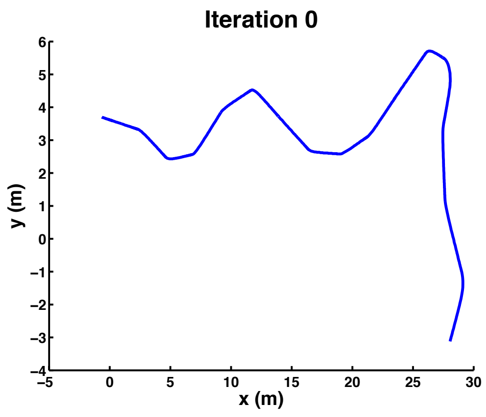
Table 40 contains the experimental setup. In this experiment the TAR Framework uses previously individually optimized trajectories as an initial point for simultaneous evolution, seen in Figure 73. Previously optimized trajectories are indicated with the initials PO in Table 40. The individual tasks were optimized for search, circle loiter, and mark the spot by themselves and then combined into one large optimization. The trajectories are approximated by 12 radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectories being adapted. All three trajectories share the basic task objectives of Recognition, Act, and Area, which are omitted from the table for brevity. The total cost is identical to the previous experiment and defined as,

$$\begin{aligned}C = R + Act_2 + Area_2 + Cover_2 + Act_3 + Area_3 + DFirstLast_3 + \\ Radii_3 + MinTheta_3 + Act_4 + Area_4 + DFirstLast_4 + \\ MarkSpot_4 + MinTheta_4. \quad (41)\end{aligned}$$

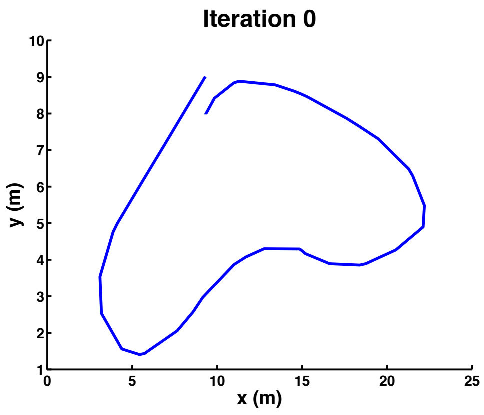
The subscript for each cost corresponds to the id number of the trajectory. For example, $Raddi_3$ corresponds to the $Raddi$ cost for the third trajectory which in this experiment is Circle Loiter.



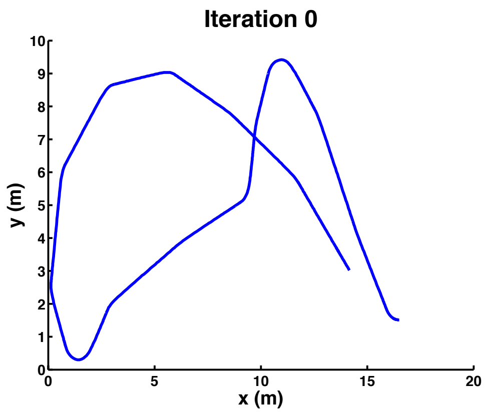
(a) Trajectory 1.



(b) Trajectory 2.

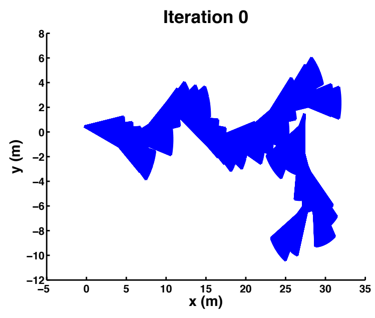


(c) Trajectory 3.

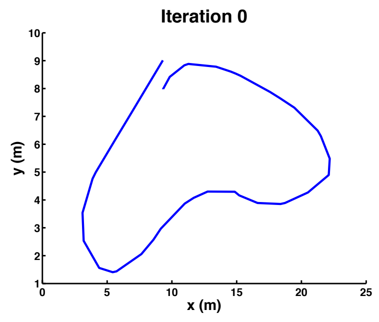


(d) Trajectory 4.

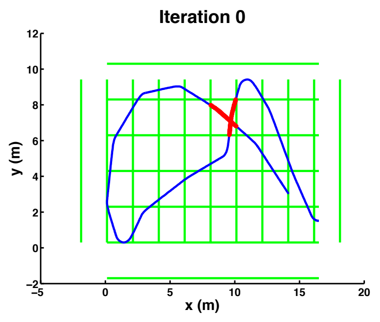
Figure 73: Initial set of trajectories to be recognized for experiment 13. Trajectory one was initialized as a straight line, seen in (a), and will remain static throughout the experiment. Trajectories two, three, and four are initialized from previously optimized trajectories for search, circle loiter, and mark the spot, respectively and can be seen in (b), (c), and (d). The TAR Framework will continue to perturb trajectories two, three, and four in order to improve recognition accuracy while maintaining search, circle loiter, and mark the spot, respectively. The common objectives for all three perturbed trajectories are to lower recognition error while maintaining vehicle dynamics and performed within a specified area. Trajectory two's final objective is to maximize sensor coverage area. Because trajectories three and four are repetitive, they both had the objectives of minimizing the distance and orientation difference between the first and last location. The final objectives for trajectory three and four were stay within two radii and mark the spot, respectively.



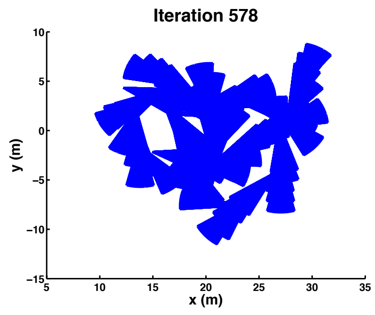
(a) Search Iteration 0.



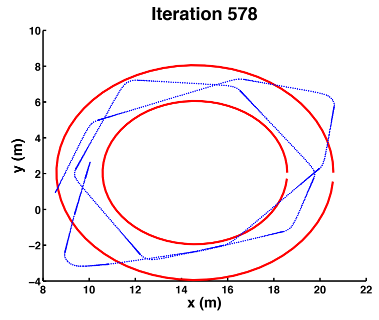
(b) Circle Loiter Iteration 0.



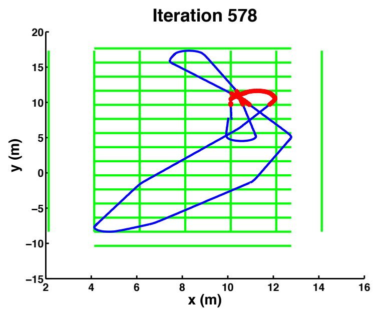
(c) Mark Spot Iteration 0.



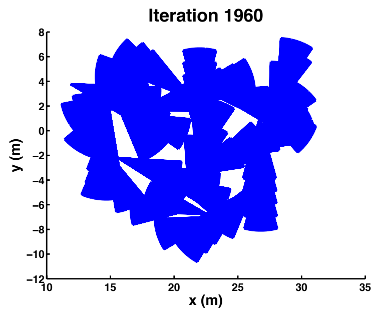
(d) Search Iteration 578.



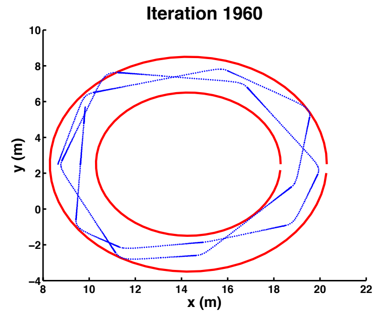
(e) Circle Loiter Iteration 578.



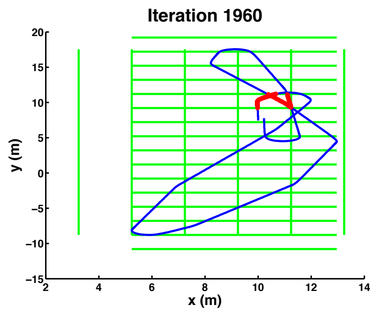
(f) Mark Spot Iteration 578.



(g) Search Iteration 1960.



(h) Circle Loiter Iteration 1960.



(i) Mark Spot Iteration 1960.

Figure 74: Experiment 13 results. The TAR framework begins with trajectories that were previously individually optimized for search, circle loiter, and mark the spot. The framework improves recognition accuracy while simultaneously ensuring the task objectives of search, circle loiter, and mark the spot. Search is seen adapting in (a), (d), and (g). Circle loiter is seen adapting in (b), (e), and (h). Mark the spot is seen adapting in (c), (f), and (i).

Results The adaptation of the trajectories is seen in Figure 74. The individually optimized trajectories all started as straight lines. Trajectory 2 was previously optimized for sensor coverage to the trajectory seen in Figure 74a. The third trajectory was previously optimized to be circle loiter-like to the trajectory seen in Figure 74b. The fourth trajectory was previously optimized to be mark the spot or figure 8 loiter-like to the trajectory seen in Figure 74c. Trajectory 2 is seen improving its area coverage in iterations 578 and 1960, in Figures 74d and 74g, respectively. Trajectory three begins as a circle loiter-like trajectory at iteration 0, seen in Figure 16, and proceeds to overlap itself within the circle Radii as seen in Figures 74e and 74h at iterations 578 and 1960, respectively. The fourth trajectory begins the simultaneous adaptation process as a figure 8 loiter-like trajectory in the initial setup, seen in Figure 74c, and then proceeds to introduce more intersections than the original as seen in Figures 74f and 74i corresponding to iterations 578 and 1960, respectively. Figure 75 displays the minimum total cost per iteration. Table 41 contains the confusion matrices for the initial setup, iteration 578, and iteration 1960. It is important to note that the recognition accuracy for each trajectory label is above 47%. By iteration 578, seen in Table 16, all the trajectories have a recognition accuracy label above 80% with two trajectories having 100% accuracy. The final recognition accuracy at iteration 1960 is 86.9%, 100%, 94%, and 100% for GoToWaypoint, Search, Circle Loiter, and Mark Spot, respectively. The final set of trajectories for recognition is seen in Figure 76.

Discussion Starting from previously optimized trajectory per respective task does produce desired results. In particular, the recognition accuracy at iteration 1960 is above 85% for all four trajectory labels. This is more impressive because circle loiter has a 94% accuracy and both search and mark the spot have 100% accuracy. In addition, qualitatively the search and circle loiter trajectories have continued to evolve into the desired trajectories. Only the fourth trajectory which had as its

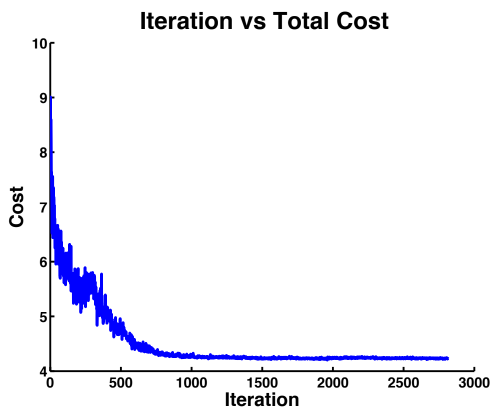
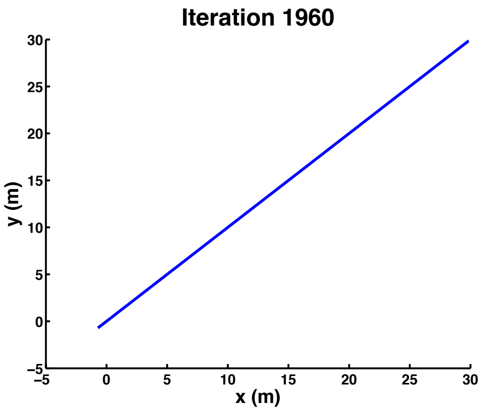
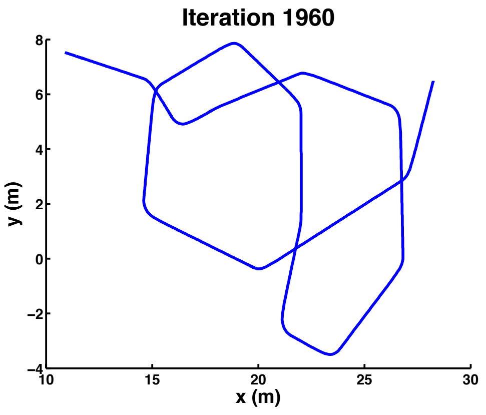


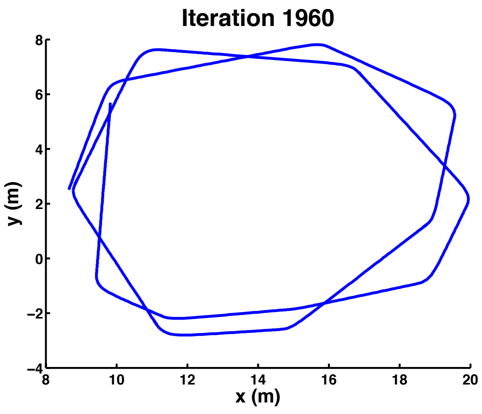
Figure 75: Experiment 13 Iteration vs Total Cost

task objective MarkSpot to create mark the spot or a figure 8 loiter-like trajectory strayed from the desired shape. However, the trajectory is able to concentrate many intersections within a four by four meter space which could be used to mark a spot of interest.

Conclusion Individually optimizing trajectories so that they accomplish their respective tasks and then combining them to simultaneously improve recognition accuracy while still accomplishing their respective tasks is successful. Recognition accuracy improves from and initial accuracy of 52$\dot{5}$%, 60.7%, 78.6% and 47.7% to 86.9%, 100%, 94%, and 100% for GoToWaypoint, Search, Circle Loiter, and Mark The Spot, respectively. Additionally, the trajectories improve their overall recognition accuracy while still maintaining their task objectives.


(a) Trajectory 1.


(b) Trajectory 2.


(c) Trajectory 3.

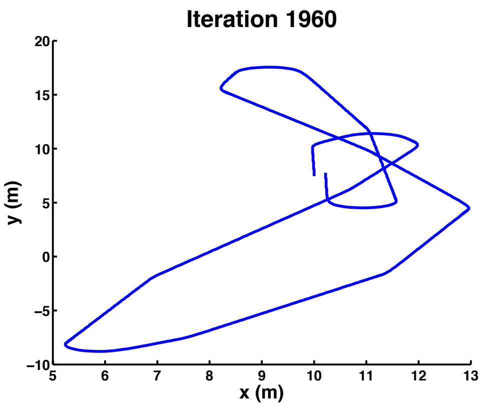

(d) Trajectory 4.

Figure 76: The final set of trajectories to be recognized at iteration 1960 in experiment 13. Trajectory one was initialized as a straight line, seen in (a), and remained static throughout the experiment. Trajectories two, three, and four were initialized from previously optimized search, circle loiter, and mark the spot experiments. The TAR Framework improved recognition accuracy while maintaining search, circle loiter, and mark the spot for trajectories two, three, and four, respectively, which can be seen at iteration 1960 in (b), (c), and (d).

Table 41: Experiment 13 Confusion Matrices

		Label				
		NoLabel	GoToWaypoint	Search	Circle Loiter	Mark Spot
Actual	*Transition*	21.60	19.90	18.70	17.00	22.80
	GoToWaypoint	0.00	52.50	44.30	0.00	3.28
	Search	0.00	7.14	60.70	26.80	5.36
	Circle Loiter	0.00	14.30	7.14	78.60	0.00
	Mark Spot	6.82	20.50	6.82	18.20	47.70

(a) Iteration 0

		Label				
		NoLabel	GoToWaypoint	Search	Circle Loiter	Mark Spot
Actual	*Transition*	62.00	32.30	1.90	1.90	1.90
	GoToWaypoint	13.10	86.90	0.00	0.00	0.00
	Search	0.00	0.00	100.00	0.00	0.00
	Circle Loiter	0.00	6.15	0.00	93.80	0.00
	Mark Spot	0.00	0.00	0.00	0.00	100.00

(b) Iteration 578

		Label				
		NoLabel	GoToWaypoint	Search	Circle Loiter	Mark Spot
Actual	*Transition*	62.20	32.70	2.56	1.28	1.28
	GoToWaypoint	13.10	86.90	0.00	0.00	0.00
	Search	0.00	0.00	100.00	0.00	0.00
	Circle Loiter	0.00	5.97	0.00	94.00	0.00
	Mark Spot	0.00	0.00	0.00	0.00	100.00

(c) Iteration 1960

Experiment 14 Setup

Table 42: Experiment 14 Setup

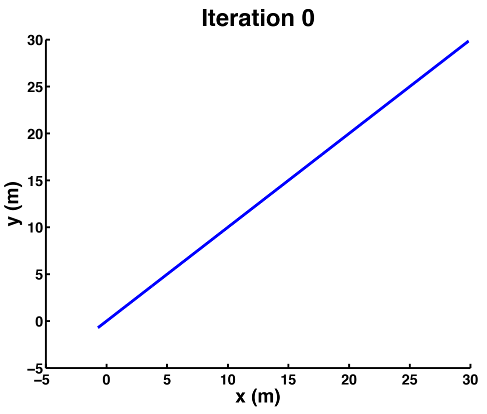
Task	Initial Trajectory	Perturb	Cost Functions
Straight Line	Straight Line	N	N/A
Search	Circle Loiter	Y	$Cover$
Circle Loiter	Circle Loiter	Y	$DFirstLast + Radii + MinTheta$
Mark Spot	Circle Loiter	Y	$DFirstLast + MarkSpot + MinTheta$
# RBFs Per Dim: 6		Initial Sigma: 2	

In experiment 14 the initial trajectories are one straight line and three identical circle loiters, seen in Figure 77. Table 42 contains the experimental setup. The trajectories are approximated by six radial basis functions (RBFs) per dimension. The initial mean for the CMA-ES optimization algorithm is a concatenated vector of the RBF weights corresponding to the trajectories being adapted. All three trajectories share the basic task objectives of Recognition, Act, and Area, which are omitted from the table for brevity. In this experiment four common tasks are optimized for: GoToWaypoint, Search, Circle loiter, and Mark the Spot. There are no priority weighting functions applied to the task objectives, meaning they are all clamped between 0 and 1. All four trajectories are simultaneously adapted but the three modifiable trajectories costs remain the same as the previous experiment,

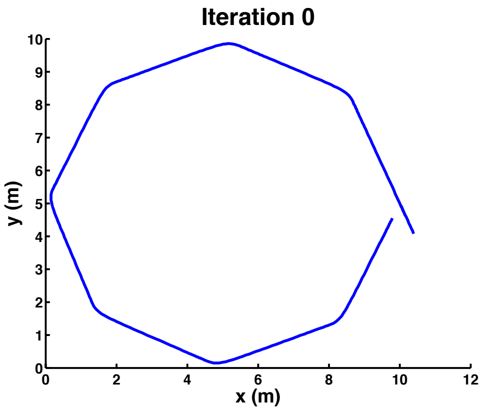
$$C = R + Act_2 + Area_2 + Cover_2 + Act_3 + Area_3 + DFirstLast_3 + Radii_3 + MinTheta_3 + Act_4 + Area_4 + DFirstLast_4 + MarkSpot_4 + MinTheta_4. \quad (42)$$

The subscript for each cost corresponds to the id number of the trajectory. For example, $Raddi_3$ corresponds to the $Raddi$ cost for the third trajectory which in this experiment is Circle Loiter.

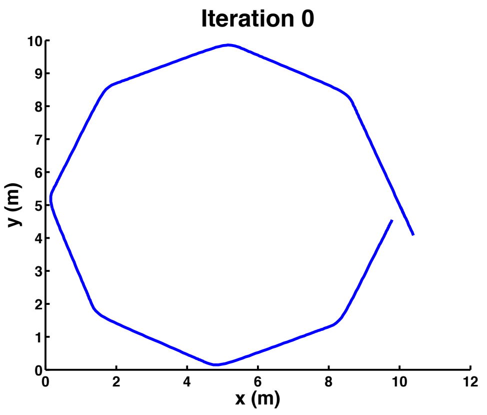
Results The evolution of the trajectories can be seen in Figure 78. The second



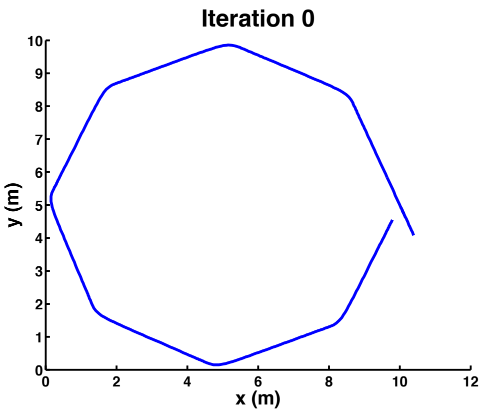
(a) Trajectory 1.



(b) Trajectory 2.

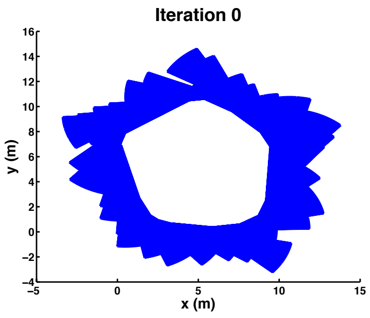


(c) Trajectory 3.

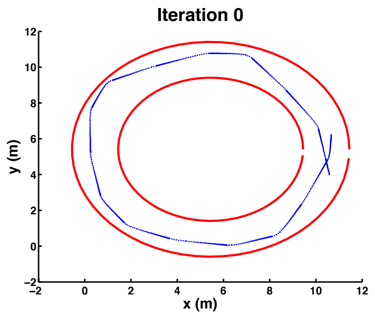


(d) Trajectory 4.

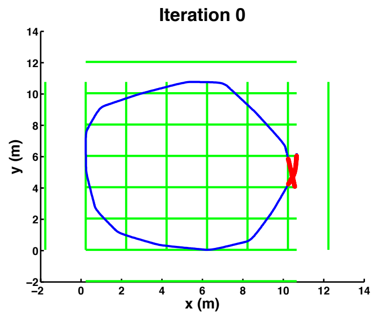
Figure 77: Initial set of trajectories to be recognized for experiment 14. Trajectory one was initialized as a straight line, seen in (a), and will remain static throughout the experiment. Trajectories two, three and four are initialized as circle loiters and can be seen in (b), (c), and (d). The TAR Framework will continue to perturb trajectories two, three, and four in order to improve recognition accuracy while maintaining search, circle loiter, and mark the spot, respectively. The common objectives for all the trajectories are minimize recognition error while maintaining vehicle dynamics and performing them within a specified area. The final objective for trajectory two is maximizing the search coverage. Because trajectory three and four were repetitive, they both included the objectives for minimizing the distance and orientation difference between the start and stop locations. The final objectives for trajectories three and four were stay within two radii and mark the spot, respectively.



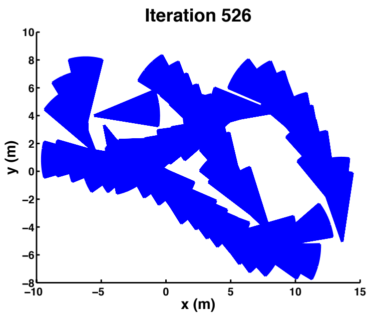
(a) Search Iteration 0.



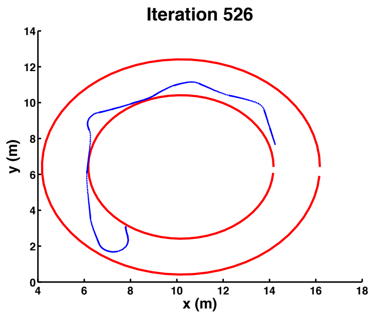
(b) Circle Loiter Iteration 0.



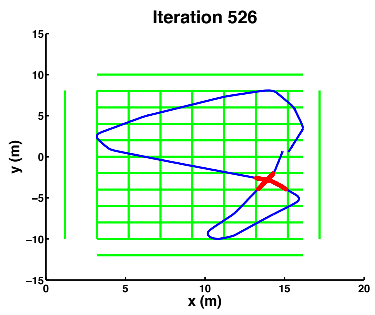
(c) Mark Spot Iteration 0.



(d) Search Iteration 526.



(e) Circle Loiter Iteration 526.



(f) Mark Spot Iteration 526.

Figure 78: Experiment 14 results. The TAR framework begins with one straight line and three circle trajectories. The framework improves recognition accuracy while ensuring task objectives of search, circle loiter, and mark the spot. Trajectory 2, seen in (a) and (d), is ensuring sensor coverage. Trajectory 3, seen in (b) and (e), is ensuring a circle loiter-like trajectory. Trajectory 4, seen in (c) and (f), is marking the spot.

trajectory is tasked with sensor coverage and can be seen as an initial circle loiter in Figure 78a and its final trajectory in Figure 78d at iteration 526. The third trajectory is tasked with maintaining a circle loiter and is seen in its initial state in Figure 78b and final state in Figure 78e at iteration 526. The fourth trajectory is tasked with marking the spot of interest and is initialized as a circle loiter, seen in Figure 78c, and accomplishes a mark the spot/figure 8 loiter at iteration 526, seen in Figure 78f. Figure 79 displays the minimum total cost per iteration. Table 43 contains the confusion matrices for the initial trajectory setup and for the final at iteration 1000. The initial recognition accuracy begins at 64.5%, 78.4%, 0%, and 0% for GoToWaypoint, Search, Circle Loiter, and Mark the spot, respectively. At the final iteration, each trajectory label has increased to a final accuracy of 83.3%, 81.5%, 72.2%, and 86.5%, respectively. The final set of trajectories for recognition can be seen in Figure 80.

Discussion Simultaneously evolving trajectories for four common tasks in which the initial modifiable trajectories begin as circle loiters can improve each task objective and recognition accuracy. It is interesting to note that in this simultaneous adaptation that the circle loiter breaks into a partial circle loiter, seen in Figure 78e. Although visually it is no longer circle loiter-like it still satisfies the Radii task objective as the points of the trajectory are within the two specified radii. The second trajectory does improve its sensor coverage task yet not as well as in previous experiments. The MarkSpot task objective does adapt from a circle into a visually appealing mark the spot of figure 8 loiter-like trajectory.

Conclusion Initializing modifiable trajectories as circle loiters can produce trajectories that accomplish the tasks of Search, Circle Loiter, and Mark the Spot while improving recognition accuracy.

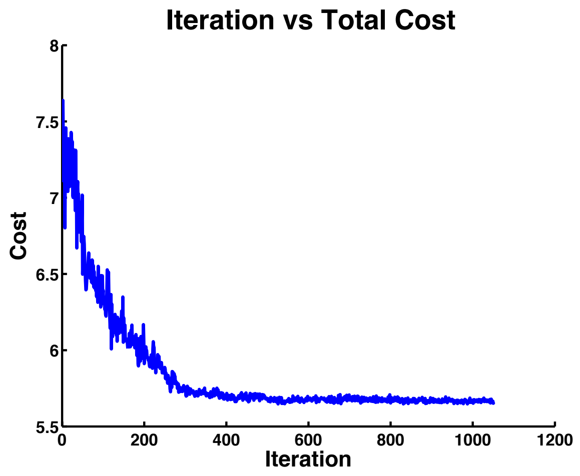


Figure 79: Experiment 14 Iteration vs Total Cost

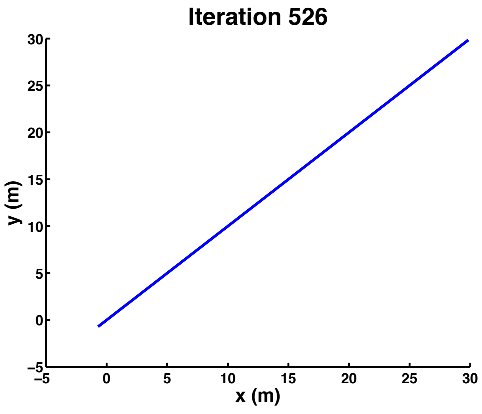
Table 43: Experiment 14 Confusion Matrices

		Label				
		NoLabel	GoToWaypoint	Search	Circle Loiter	Mark Spot
Actual	*Transition*	0.00	22.80	77.20	0.00	0.00
	GoToWaypoint	0.00	64.50	35.50	0.00	0.00
	Search	0.00	21.60	78.40	0.00	0.00
	Circle Loiter	0.00	38.70	61.30	0.00	0.00
	Mark Spot	0.00	34.40	65.60	0.00	0.00

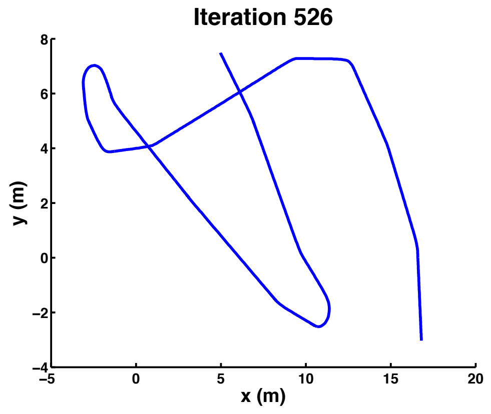
(a) Iteration 0

		Label				
		NoLabel	GoToWaypoint	Search	Circle Loiter	Mark Spot
Actual	*Transition*	3.89	28.30	10.60	50.60	6.67
	GoToWaypoint	0.00	83.30	0.00	16.70	0.00
	Search	0.00	3.70	81.50	14.80	0.00
	Circle Loiter	0.00	27.80	0.00	72.20	0.00
	Mark Spot	0.00	13.50	0.00	0.00	86.50

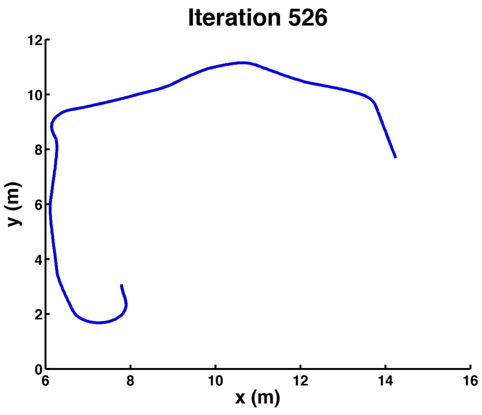
(b) Iteration 1000



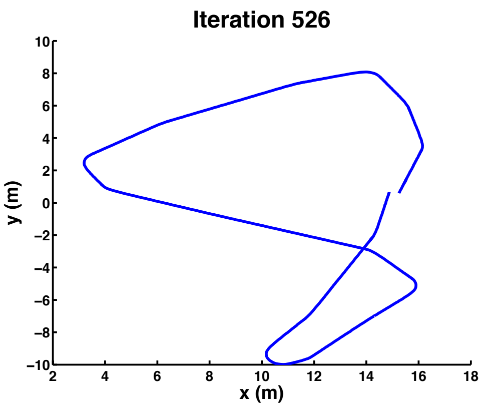
(a) Trajectory 1.



(b) Trajectory 2.



(c) Trajectory 3.



(d) Trajectory 4.

Figure 80: The final set of trajectories to be recognized at iteration 526 in experiment 14. Trajectory one was initialized as a straight line, seen in (a), and remained static throughout the experiment. Trajectories two, three, and four were initialized as circle loiters. The TAR Framework improved recognition accuracy while maintaining search, circle loiter, and mark the spot for trajectories two, three, and four, respectively, which can be seen at iteration 526 in (b), (c), and (d).

6.9 Conclusion

These experiments have demonstrated the feasibility of the TAR framework in generating and modifying trajectories that an autonomous vehicle can track with minimal error, accomplishes task objectives, and improves recognition accuracy. The first experiment demonstrated that in a simple scenario of comparing two trajectories the TAR framework can improve recognition accuracy. Although it improved recognition accuracy, the actual commanded trajectory had tracking errors by the vehicle from either the autonomy or vehicle dynamics or a combination of the two. Therefore, the following experiments included an objective for actionability (Act) as a measure for error in tracking. Adding such an objective encourages the TAR Framework to favor trajectories that the autonomous vehicle can follow with little error. Subsequent experiments demonstrated that given simple common tasks that TAR Framework can improve recognition accuracy while adhering to tasks objectives such as sensor coverage and maintaining a circle loiter. A more difficult task is autonomously generating a mark the spot or figure 8 loiter-like trajectory. In order to produce trajectories that could be leveraged for cooperative tasks the three objectives MarkSpot, DFirstLast, and MinTheta were introduced. Theses experiments demonstrated the stochastic nature of the search algorithm CMA-ES. CMA-ES can favor a local minima that produce trajectories that do not have the desired trajectory properties. More difficult scenarios were tackled such as that commonly encountered by AUVs in a search mission. During the search mission, an AUV will use multiple instances of circle loiter for different purposes such as wait for commands, error state, and mission completion. These experiments demonstrated the need to use priority weighting functions for the different task objectives in order to maintain circle loiter-like trajectories. Additionally, the difficulty of the scenario increased as more and more similar trajectories were included for recognition and modification, potentially demonstrating the limit of this instantiation of the TAR framework. The final set of experiments demonstrated the

TAR framework's ability to generate or modify four common tasks simultaneously ensuring task objectives were met along with improving recognition accuracy. These experiments demonstrated the efficacy of starting in different portions of the state space of trajectories: straight lines, circle loiters, or previously optimized trajectories.

CHAPTER VII

CONCLUSION

The primary contribution of this dissertation is the adaptation or modification of trajectories performed by autonomous vehicles so that trajectory recognition accuracy increases while still maintaining task objectives. The insight to adapt trajectories for improved recognition was motivated by experiments in recognizing the common trajectories performed by autonomous vehicles in Chapter 4. That foundational work demonstrated trajectory recognition of common autonomous vehicle tasks using various trajectory recognition methods. Even though various methods were employed and parameters tuned for improved recognition accuracy, there were situations in which the recognition accuracy was below an acceptable level for deployment. Experiments in Chapter 5 introduced the multi-robot cooperation strategy of using trajectory-based signaling similar to that of the honeybee "waggle" dance. The *InfinityPattern* trajectory was used to indicate a location of interest to a teammate during periods of intermittent or denied communications. While that foundational work explored using trajectory-based signaling for autonomous vehicle cooperation, there were issues with the manually created trajectories to indicate the location of interest. In particular, the manually created trajectories were often not viable for the autonomous vehicle to track without error. If the manually created trajectory had little tracking error, it often did not improve the trajectory recognition accuracy to a desired level. This motivated the insight of exploring a method to automatically adapt trajectories such that their combined recognition accuracy improved, were actionable by the autonomous vehicles, and yet still performed the desired tasks. The introduction of the Trajectory Adaptation for Recognition (TAR) framework in Chapter 6 directly addresses this adaptation process.

The experiments in Chapter 6 demonstrated TAR's ability to improve trajectory recognition while still enabling the trajectories to perform task objectives. Cost functions were designed to address the common autonomous vehicle tasks such as search pattern and circle loiter, which were originally recognized in Chapter 4. These cost functions ensured that the trajectories still performed their desired tasks. In order to address using a trajectory for trajectory-based signaling, originally studied in Chapter 5, a Mark the Spot cost function was introduced. An important facet of the TAR framework is the inclusion of whether an autonomous vehicle can follow a commanded trajectory. This is important because the TAR framework may adapt a trajectory for an improvement in recognition accuracy but if the autonomous vehicle can not follow the trajectory with little error then it is not viable. Dynamic time warping (DTW) was used to measure if commanded trajectories were actionable by the simulated autonomous vehicles. Once the cost functions for maintaining task objectives were verified, the TAR framework applied to more complicated scenarios. The first complicated scenario was a search mission often performed by autonomous underwater vehicles in which identical circle loiter patterns are employed for one of many states such as holding patter, error state, and mission completion. The experiments demonstrated that the TAR framework could modify the circle loiters for better distinguishability with some limitations. The final set of experiments demonstrated that the TAR framework can simultaneously adapt trajectories to fulfill common autonomous vehicle tasks such as search, circle loiter, and mark the spot while improving recognition accuracy.

The TAR framework is an initial attempt at jointly adapting autonomous vehicle trajectories for improved recognition accuracy while maintaining task objectives. There are several areas of research to pursue. The choice was made to have the recognition and trajectory modification representation be the same with the use of radial basis functions. Alternative basis functions may be appropriate for different tasks such as Fourier basis or Chebyshev polynomials. This is especially the case when trajectories have repeating global properties such as the search task. Alternatively, a disconnect can be made between the representation for recognition and modification. For example, a probabilistic graphical model can be employed for recognition such as a CRF while still using basis functions to approximate and modify the trajectories. The exploration of the possible trajectories was performed with the evolutionary algorithm CMA-ES so that cost functions for the task objectives could be as general as possible. An area of exploration is applying gradient-based optimization methods and corresponding cost functions. A natural extension of these experiments is the inclusion of sensor simulation for the observing autonomous vehicle. Inclusion of a sensor will increase the complexity of the optimization process. For example, the location that acting vehicle performs its trajectories in relation to the observer must be accounted for as there may be increased error with distance or angle. The resulting trajectories may improve recognition accuracy and inform the team strategy that the observer must be within certain bounds around the actor. While the TAR framework is an initial attempt, experiments demonstrated its ability to improve recognition accuracy while maintaining task objectives. Such a framework paves the way for enabling autonomous teammates to automatically adapt their motions so that they are more easily recognizable by each other while still accomplishing their intended tasks.

REFERENCES

[1] Akgün, B., Tunaöglu, D., and Sahin, E., "Action recognition through an action generation mechanism," in *International Conference on Epigenetic Robotics (EPIROB)*, 2010.

[2] Arkin, R. C., *Behavior-Based Robotics*, ch. 9. Cambridge, Mass., MIT Press, 1998.

[3] Baillieul, J., "Reliable and efficient communication through a controlled dynamical system," in *Information Sciences and Systems (CISS), 2012 46th Annual Conference on*, pp. 1–4, IEEE, 2012.

[4] Balch, T., Dellaert, F., Feldman, A., Guillory, A., Isbell, C., Khan, Z., Stein, A., and Wilde, H., "How multirobot systems research will accelerate our understanding of social animal behavior," *Proceedings of the IEEE*, vol. 94, pp. 1445–1463, 2006.

[5] Baxter, R., Lane, D., and Petillot, Y., "Behaviour recognition for spatially unconstrained unmanned vehicles," in *Proceedings of the IJCAI*, vol. 9, pp. 1–8, 2009.

[6] Baxter, R. H., Lane, D. M., and Petillot, Y., "Recognising agent behaviour during variable length activities," in *Proceedings of The 19th European Conference on Artificial Intelligence*, 2010.

[7] Benjamin, M. R., "MOOS-IvP Autonomy Tools Users Manual," Tech. Rep. MIT-CSAIL-TR-2010-039, MIT Computer Science and Artificial Intelligence Lab, August 2010.

[8] Benjamin, M., Schmidt, H., Newman, P., and Leonard, J., "Nested autonomy for unmanned marine vehicles with MOOS-IvP," *Journal of Field Robotics*, vol. 27, no. 6, pp. 834–875, 2010.

[9] Cao, Y., Fukunaga, A., and Kahng, A., "Cooperative mobile robotics: Antecedents and directions," *Autonomous robots*, vol. 4, no. 1, pp. 7–27, 1997.

[10] Chairman, "STANAG 4586 (edition 3) standard interfaces of UAV control system (UCS) for NATO UAV interoperability," *Standardization Agreement*, p. 509, 2012.

[11] Demiris, Y., "Prediction of intent in robotics and multi-agent systems," *Cognitive processing*, vol. 8, no. 3, pp. 151–158, 2007.

[12] Dias, M. and Stentz, A., "A free market architecture for distributed control of a multirobot system," in *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pp. 115–122, July 2000.

[13] Dragan, A., Lee, K., and Srinivasa, S., "Legibility and predictability of robot motion," in *Human-Robot Interaction*, March 2013.

[14] Dudek, G., Jenkin, M., Milios, E., and Wilkes, D., "A taxonomy for multi-agent robotics," *Autonomous Robots*, vol. 3, no. 4, pp. 375–397, 1996.

[15] Dzielski, J., DeLorme, M., Sedunov, A., Sammut, P., and Tsionskiy, M., "Guidance of an unmanned underwater vehicle using a passive acoustic threat detection system," *IEEE Waterside Security Conference (WSS)*, 2010.

[16] Feldman, A., and Balch, T., "Representing honey bee behavior for recognition using human trainable models," *Adaptive Behavior*, vol. 12, pp. 241–250, Dec. 2004.

[17] Gallese, V. and Goldman, A., "Mirror neurons and the simulation theory of mind-reading," *Trends in cognitive sciences*, vol. 2, no. 12, pp. 493–501, 1998.

[18] Gerkey, B. and Mataric, M., "Sold!: Auction methods for multirobot coordination," *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 5, pp. 758–768, 2002.

[19] Guard, U. C., *U.S. Coast Guard Addendum to the United States National Search and Rescue Supplement to the International Aeronautical and Maritime Search and Rescue Manual.* U.S. Department of Homeland Security. Washington, DC, 2000.

[20] Han, K. and Veloso, M., "Automated robot behavior recognition," in *Robotics Research-International Symposium*, vol. 9, pp. 249–256, 2000.

[21] Hansen, N., "The CMA evolution strategy: a comparing review," in *Towards a new evolutionary computation*, pp. 75–102, Springer, 2006.

[22] Hazan, A., Davesne, F., Vigneron, V., and Maaref, H., "Topological characterization of mobile robot behavior," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 4157–4162, IEEE, 2007.

[23] Hill, R. R., Carl, R. G., and Champagne, L. E., "Using agent-based simulation to empirically examine search theory using a historical case study.," *Journal Of Simulation*, vol. 1, no. 1, pp. 29 – 38, 2006.

[24] Inamura, T., Toshima, I., Tanie, H., and Nakamura, Y., "Embodied symbol emergence based on mimesis theory," *The International Journal of Robotics Research*, vol. 23, no. 4-5, pp. 363–377, 2004.

[25] Irish, J. and Lillycrop, J., "Scanning laser mapping of the coastal zone: the shoals system," *ISPRS Journal of Photogrammetry and Remote Sensing*, 1999.

[26] Jones, A. and Andersson, S., "A motion-based communication system," in *American Control Conference (ACC), 2013*, pp. 365–370, IEEE, 2013.

[27] Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S., "STOMP: Stochastic trajectory optimization for motion planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4569–4574, IEEE, 2011.

[28] Kanno, T., Nakata, K., and Furuta, K., "A method for team intention inference," *International Journal of Human-Computer Studies*, vol. 58, no. 4, pp. 393–413, 2003.

[29] Kohlsdorf, D., Starner, T., and Ashbrook, D., "Magic 2.0: A web tool for false positive prediction and prevention for gesture recognition systems," in *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pp. 1–6, IEEE, 2011.

[30] Kuniyoshi, Y., "Behavior matching by observation for multi-robot cooperation," in *Robotics Research-International Symposium*, vol. 7, pp. 343–352, MIT PRESS, 1996.

[31] Lafferty, J., McCallum, A., and Pereira, F., "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *International Conference on Machine Learning*, 2001.

[32] (N85), N. E. W. D., "Mine warfare platforms, programs and systems," *Naval expeditionary warfare vision 2010*, 2010.

[33] Nag, R., Wong, K., and Fallside, F., "Script recognition using hidden Markov models," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP*, vol. 11, pp. 2071 – 2074, April 1986.

[34] Navy, U., *NAVY Search and Rescue Tactical Information Document (SAR TACAID)*, vol. NWP 3-22.5-SAR-TAC NAVAIR A1-SARBA-TAC-000. NAVY WARFARE DEVELOPMENT COMMAND, 1997.

[35] Novitzky, M., "Improvement of multi-auv cooperation through teammate verification," in *Workshop PAMR at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[36] Novitzky, M., Balch, T., and Weiss, L., "Optimizing robot behavior for improved recognition by teammates," *Distributed autonomous robotic systems (DARS)*, 2014.

[37] Novitzky, M. and Balch, T. R., "Towards trajectory adaptation for recognition (TAR)," *ICRA 2015: Workshop on Persistent Autonomy for Aquatic Robotics (PAAR)*, 2015.

[38] Novitzky, M., Benjamin, M. R., Collins, T. R., Balch, T., West, M. E., and Weiss, L., "Behavior-based communication for cooperative mine clearing with an ASV," *Distributed autonomous robotic systems (DARS)*, 2014.

[39] Novitzky, M., Pippin, C., Balch, T. R., Collins, T. R., and West, M. E., "Behavior recognition of an AUV using a forward-looking sonar," in *Workshop on Marine Robotics at the Robotics Science and Systems (RSS). Los Angeles, CA*, 2011.

[40] Novitzky, M., Pippin, C., Collins, T. R., Balch, T., and West, M. E., "Behavior recognition of autonomous underwater vehicles using forward-looking sonar," *Distributed autonomous robotic systems (DARS)*, 2012.

[41] Novitzky, M., Pippin, C., Collins, T. R., Balch, T. R., and West, M. E., "Bio-inspired multi-robot communication through behavior recognition," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2012.

[42] Novitzky, M., Pippin, C., Collins, T. R., Balch, T. R., and West, M. E., "Conditional random fields for behavior recognition of autonomous underwater vehicles," in *Distributed autonomous robotic systems (DARS)*, pp. 409–421, Springer, 2012.

[43] Novitzky, M., Pippin, C., Collins, T. R., Balch, T. R., and West, M. E., "AUV behavior recognition using behavior histograms, HMMs, and CRFs," *Robotica*, vol. 32, pp. b1–b4, 3 2014.

[44] Oh, S., Rehg, J., Balch, T., and Dellaert, F., "Learning and inference in parametric switching linear dynamic systems," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, pp. 1161–1168, IEEE, 2005.

[45] Organization, I. M., *International Aeronautical and Maritime Search and Rescue Manual (IAMSAR).* International Maritime Organization. London, United Kingdom, 2013.

[46] Parker, L. E., "The effect of action recognition and robot awareness in cooperative robotic teams," in *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 1, pp. 212–219, IEEE, 1995.

[47] Plamondon, R. and Srihari, S. N., "Online and off-line handwriting recognition: a comprehensive survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 1, pp. 63–84, 2000.

[48] Rabiner, L., "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[49] Raghunathan, D. and Baillieul, J., "Relative motion of robots as a means for signaling," *Proceedings of the Intl Conf. on Intelligent Automation and Robotics, San Francisco, USA*, pp. 22–24, 2008.

[50] Raghunathan, D. and Baillieul, J., "Exploiting information content in relative motion," in *American Control Conference, 2009. ACC'09.*, pp. 2166–2171, IEEE, 2009.

[51] Raghunathan, D. and Baillieul, J., "Motion based communication channels between mobile robots-a novel paradigm for low bandwidth information exchange," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 702–708, IEEE, 2009.

[52] Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S., "CHOMP: Gradient optimization techniques for efficient motion planning," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 489–494, IEEE, 2009.

[53] Rizzolatti, G., Fadiga, L., Gallese, V., and Fogassi, L., "Premotor cortex and the recognition of motor actions," *Cognitive brain research*, vol. 3, no. 2, pp. 131–141, 1996.

[54] Sariel, S., Balch, T., and Erdogan, N., "Naval mine countermeasure missions," *Robotics & Automation Magazine, IEEE*, vol. 15, no. 1, pp. 45–52, 2008.

[55] Schaal, S., "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive Motion of Animals and Machines*, pp. 261–280, Springer, 2006.

[56] Schulman, J., Ho, J., Lee, A., Awwal, I., Bradlow, H., and Abbeel, P., "Finding locally optimal, collision-free trajectories with sequential convex optimization.," in *Robotics: Science and Systems*, vol. 9, pp. 1–10, Citeseer, 2013.

[57] Search, N. and Committee, R., *United States National Search and Rescue Supplement to the International Aeronautical and Maritime Search and Rescue Manual.* National Search and Rescue Committee. Washington, DC, 2000.

[58] Sotzing, C. and Lane, D., "Improving the coordination efficiency of limited-communication multi-autonomus underwater vehicle operations using a multiagent architecture," *Journal of Field Robotics*, vol. 27, no. 4, pp. 412–429, 2010.

[59] Stone, P. and Veloso, M., "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork," *Artificial Intelligence*, vol. 110, no. 2, pp. 241–273, 1999.

[60] Sukthankar, G., *Activity recognition for agent teams.* PhD thesis, CMU, 2007.

[61] Tambe, M. and Rosenbloom, P., "RESC: An approach for real-time, dynamic agent tracking," in *International Joint Conference on Artificial Intelligence*, vol. 14, pp. 103–111, 1995.

[62] Theodoridis, S. and Koutroumbas, K., *Pattern Recognition, Fourth Edition.* Academic Press, 4th ed., 2008.

[63] Tulldahl, M. and Pettersson, M., "Lidar for shallow underwater target detection," *Proceedings of the SPIE, Electro-Optical Remote Sensing, Detection, and Photonic Technologies and Their Applications*, 2007.

[64] Vail, D. and Veloso, M., "Feature selection for activity recognition in multi-robot domains," in *Proceedings of AAAI*, 2008.

[65] Vail, D., Veloso, M., and Lafferty, J., "Conditional random fields for activity recognition," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pp. 1–8, ACM, 2007.

[66] Von Frisch, K., "The dance language and orientation of bees.," 1967.

[67] West, M., Novitzky, M., Varnell, J., Melim, A., Sequin, E., Toler, T., Collins, T., and Bogle, J., "Design and development of the Yellowfin UUV for homogenous collaborative missions," *Association for Unmanned Vehicle Systems International*, 2010.

[68] Williams, D. L., "Loitering behavior of autonomous underwater vehicles," Master's thesis, Naval PostGraduate School, Dudley Knox Library Naval postgraduate School 411 Dyer Road 1 University Circle Monterey, California USA 93943, 6 2002.