Algorithms and Systems for Robot Videography from Human Specifications

Florian Shkurti

This thesis is submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

School of Computer Science
McGill University, Montreal

Abstract

In this thesis we view autonomous mobile robots as visual data collection platforms that work alongside human experts to help them record the type of footage they need. This viewpoint is useful in applications such as human-robot collaborative environmental monitoring, where for example, a diver needs to explore a marine environment assisted by an underwater robot; in visual search and inspection, where a farmer needs to get footage of crops that are at risk of disease; or in robot cinematography of athletic events. In all these cases, robots act as *videographers* on behalf, and in assistance, of humans who have a well-defined objective with respect to the type of visual data they want to collect. The two main questions examined in this thesis are: (a) how can we efficiently learn the objective function with respect to which a human expert records visual data, and (b) how can we enable robots to reliably visually navigate in 3D, alongside humans, in order to help them collect data that will be of use to their users? The main contributions of this thesis are twofold: for (a) we propose an active learning based approach for modeling human preferences over visual data, and for (b) we propose visual tracking systems that take into account the uncertainty of the behavior of the human learned via inverse reinforcement learning, as well as visual navigation approaches based on stereo cameras and inertial information. We show validation of these algorithms in the context of collaboration between humans and multiple robots for environmental monitoring in multiple large-scale robot field trials.

Abrégé

Dans cette thèse, nous considérons les robots mobiles autonomes comme des plateformes de collecte de données visuelles qui fonctionnent aux côtés d'experts humains pour les aider enregistrer le type de métrage dont ils ont besoin. Ce point de vue est utile dans des applications telles que la surveillance environnementale collaborative homme-robot, où par exemple, un plongeur doit explorer un environnement marin assisté par un robot sous-marin; dans la recherche visuelle et l'inspection, où un agriculteur doit obtenir images de cultures à risque de maladie; ou dans la cinématographie robotique d'événements sportifs. Dans tous ces cas, les robots agissent en tant que vidéographes pour le compte de et en assistance, des personnes ayant un objectif bien défini en ce qui concerne le type de données visuelles qu'elles souhaitent collecter. Les deux questions principales examinées dans cette thèse sont: (a) comment pouvons-nous apprendre efficacement la fonction objective à laquelle un expert humain enregistre donnèes visuelles, et (b) comment permettre aux robots de naviguer visuellement de manière fiable en 3D, aux côtés des humains, afin de les aider à collecter des données qui seront utiliser pour leurs utilisateurs? Les principales contributions de cette thèse sont doubles: pour (a) nous proposons une approche basée sur l'apprentissage actif pour modéliser les préférences humaines données visuelles, et pour (b) nous proposons des systèmes de suivi visuel qui tiennent compte de l'incertitude du comportement de l'homme appris par l'apprentissage par renforcement, ainsi que les approches de navigation visuelle basées sur des caméras stéréoscopiques et des informations inertielles. Nous montrons la validation de ces algorithmes dans le cadre d'une collaboration entre des humains et de multiples robots pour la surveillance de l'environnement dans le cadre d'essais multiples sur le terrain à grande échelle.

Acknowledgements

I want to express my gratitude towards my supervisor, Prof. Gregory Dudek. It is not an exaggeration to say that my life would have taken a completely different trajectory if it weren't for the opportunities that he gave me at different key points in my career, starting with admitting me to his lab. Aside from expectations of scientific excellence and true care, vision, creativity, and enthusiasm about the future of robotics, what sets Greg apart in terms of supervision style is that he constantly sees the people he works with not as they are, but as what they can be. As the pace of research becomes more fast-paced in A.I.-related fields, it is a privilege to have had a supervisor who allows his students the academic freedom to focus on important, foundational, long-term problems in the field. I also want to thank him for keeping field robotics active and healthy in Canada, and showing us glimpses of visions of what robots can do for us, despite the cost, and the large effort and time required on his part to organize these experiments. I consider it a highlight of my education to have spent time getting machines to work intelligently in the wild. Nothing beats having the ocean, or the forest, or the Utah desert, as your office for a few days, even if it means getting a few scratches, scrapes, and seeing fires(!) and unscheduled landings from robots in the meantime. I also want to thank his wife, Kris, for welcoming me and my labmates, while we invaded their family vacations on many occasions to do experiments.

Speaking of labmates, with so many talented colleagues, where does one begin? I want to thank Anqi Xu, who has been a wonderful friend since I joined the lab and an inspiring labmate. I am a big fan of his gusto and joie de vivre. I also want to thank my colleagues, amazing friends, and roommates for many years, Juan Camilo Gamboa Higuera and Arnold Kalmbach, for being exceptionally creative and technical. Travis Manderson has been a great friend, collaborator, and a source of constant learning for me. I am amazed by the sheer breadth and depth of his knowledge at the intersection of hardware and software. Yogesh Girdhar has had a huge influence on which problems I view as important, and has always been extremely fun to work with. I missed his presence in the lab dearly after he graduated. Malika Meghjani has been my officemate, friend and collaborator on many papers and projects. I will never forget the adventures we had while trying to get our robots to work in the ocean. Sandeep Manjanna has been a source of serenity amid chaos,

and always a great friend and officemate. Johanna Hansen has been an amazing friend and colleague and a source of leadership, technical knowledge, and machine learning wisdom. I regret the fact that I never got the chance to work on a paper with Sandeep and Johanna. I also want to say thanks to Jimmy Li, whose focus and organization has been inspiring, and to Nikhil Kakodkar for being an amazing collaborator, and for raising the spirits of the lab in high pressure situations. Wei-Di Chang has been a fantastic and reliable collaborator. So has Peter Henderson. Also, Karim Koreitem, Andrew Holliday, Lucas Berry, Auguste Lalande, Yi Tian Xu, and Monika Patel who always made it a pleasure to go to the lab. Scott Fujimoto and Ed Smith have also been extremely inspiring, both in research discussions but also socially. I also want to acknowledge the "old guard": Philippe Giguere, Junaed Sattar, and Ioannis Rekleitis, who have supported me time and again over the years with career and research advice. I'm lucky to have met them.

I owe many overdue, but heartfelt, thanks to the engineers who worked tirelessly in the lab and made our research possible: Chris Prahacs, who designed and built Aqua, Bikram Dey, Nik Pateromichelakis and Ian Karp. Without them I would be stuck in simulation.

On the professors' end I want to thank Luc Devroye for his inimitable enthusiasm and contagious love for probability. I also want to thank Joelle Pineau, Doina Precup, and Mike Langer for their feedback and research discussions. I have learned a lot from them. David Meger has always been a voice of research wisdom, especially during brainstorming sessions. It has always been fun working with him on many projects and papers, and I'm going to miss having him around.

Many thanks also to Isabelle Lacroix, Jan binder, Nick Wilson, Laurena Deligny, Diti Anastasopoulos, Tricia Bernier, Sheryl Morrissey and Ann Jack, for doing heavy lifting on the admin side to support our work.

Last, but not least, I want to thank my parents, Tasim and Adriana and my sister, Vjosana, for their patience and encouragement, and for providing an environment where we could pursue self-indulgent activities, such as a Ph.D. degree. Through their hard work they made us lucky.

Contents

List of Figures

List of Tables

Dedicated to my mentors.

1 INTRODUCTION

The unifying research objective of this thesis is to enable robots to autonomously perform visual exploration in challenging, unknown, outdoor environments in a way that is informed by human preferences and specifications. The main application area that we aim to have an impact on is the robotic automation of environmental monitoring.

Field robotics provides an abundance of long video sequences collected from a large number of robot deployments in various outdoor scenes ranging from snowy forests, to deserts and underwater environments. The volume of this data is often overwhelming to the human who has to inspect it for useful insights, as robots are typically oblivious to the type of data that humans need. This problem is of particular importance in environmental monitoring, since scientists often need to closely examine the recorded data to make observations and measurements about the health of ecosystems. There are two complementary ways to address this problem: *active vision* and *video summarization*. In active vision the robot visits regions of the environment, and directs the camera towards parts of the scene that are most informative according to some objective. Video summarization on the other hand makes no assumptions about whether the data has been deliberatively collected or not, and presents the highlights of the video to the user.
Our approach in this thesis is to follow the active vision approach, in a way that the human's preferences are taken into account in the robot's visual attention model during exploration. In order to enable this type of informed visual exploration we asked the following two guiding questions:

> *1. How can users efficiently specify the type of visual data that they want their robots to record?*
> *2. How should robots help humans explore unknown natural environments according to that specification?*

For the first question we seek data-efficient learning methods that will enable users to quickly affect the visual attention process their robots use for exploratory behavior. For the second we focus on scenarios in which the robot videographer occasionally tracks the human using vision, while at the same time actively explores the environment, being robust to uncertainty due to intermittent observations.

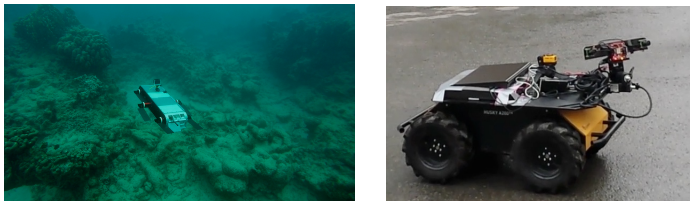


FIGURE 1.1: Typical active vision and informed exploration scenarios in robotics: (a) underwater robot tasked with monitoring the health of corals (b) ground robot controlling a pan-tilt-zoom camera while moving.

1.1 Visual Attention Models for Robotics via Human Specifications

The problem of formulating computational models of visual attention in robotics has a rich history, which has been summarized in [Frintrop et al., 2010]. A broader treatment, looking at modeling the stages of visual attention in the brain is given in [Bruce et al., 2015, Tsotsos, 2011, Zhaoping, 2014, Bruce and Tsotsos, 2008]. This thesis is only broadly related to that literature insofar as the long-term goal is to find how to best incorporate top-down cues for visual attention from human input [Tsotsos, 2011]. More concretely, however, we look at modeling saliency maps, a special case of visual attention in robotics, without investigating issues related to neuroscience or psychophysics. We also establish connections between learned saliency maps, and reward functions based on images, which so far has mostly been examined in the context of inverse reinforcement learning and simple low-dimensional scenarios [Sadigh et al., 2017, Ramachandran and Amir, 2007].

1.2 Collaborative Human-Robot Visual Exploration

Collaborative exploration [Doherty et al., 2018, Burgard et al., 2000] between humans and robots in outdoor environments provides a number of advantages over robot-only or human-only exploration. The major difference that humans can make is that they have better judgment over the parts of the environment worth paying attendion to. Robots on the other hand can have mobility advantages such as speed, or in the case of flying and underwater vehicles, the capability to view the scene from multiple altitudes.

If robots are able to explore around the human's vicinity, and understand the user's preferences and intent, they will be more effective at exploration than if they ignore such preferences. Enabling a robot to remain in the vicinity of a human,

using visual observations, is a problem with rich history. That said, allowing for intermittent visual contact is typically beyond the consideration of existing research. Pursuit-evasion games [Vidal et al., 2002], for example, where this aspect of the problem is typically dealt with, mostly comprise impractical methods that make a large number of unreasonable assumptions about idealized environments and optimality guarantees.

In this thesis we examine visual tracking and pursuit methods that are robust to intermittent visual contact. These methods make predictions about the short-term and long-term behavior of humans based on patterns of navigation from historical observations. We also look at the problem of visual state estimation and navigation in 3D space, which conceptually underlies the navigation process in scenarios of collaborative human-robot exploration.

1.3 Contributions

This thesis includes the following contributions:

- **Active learning of informed visual attention models:** an active supervised learning approach that minimizes the model uncertainty of a distribution of pixel-level saliency maps. This method actively selects the most informative images to present to the expert user for pixel-level annotation of their regions of interest as captured in the image. This will be presented in Chapter 4.

- **Collaborative human-robot visual exploration:** we present robot field trials involving cooperation between aerial, sea surface, and underwater vehicles exploring environments in a semi-autonomous fashion, according to the specifications of a human. These types of multi-robot systems working in concert with humans are unique in the robotics literature and have paved the way for novel environmental monitoring applications. Relevant publications for this topic include [Shkurti et al., 2012] and [Shkurti et al., 2017], which are presented in more detail in Chapters 3 and 6, respectively.

- **Predictive pursuit in belief space using known behavior models:** a method that combines visual tracking, prediction, planning and control so as to be able to follow a known target with intermittent observations. Unlike many existing trackers, it is able to deal with the case where visual contact with the subject is lost, by making informed predictions about its subject's behavior, which is learned through inverse reinforcement learning. This is one of the first works to enable practical pursuit at scale in realistic environments. The

main publication for this method is [Shkurti et al., 2018] and it is discussed in Chapter 5 of this thesis.

- **Topological constraints in predictive pursuit:** a way to topologically constrain the predictions of a pursuit algorithm in order to condense the expanse of possibilities that the algorithm must consider, which enables target tracking in large environments, surpassing related methods for planning under uncertainty. This was presented in [Shkurti and Dudek, 2017] and it will be discussed in Chapter 5.

- **The complexity of pursuit:** we prove complexity theory results that show the hardness of the pursuit problem even when the follower has speed advantage over the target. This was published in [Shkurti and Dudek, 2013] and the main results appear in Appendix A.

- **Stereo vision and inertial localization and mapping:** a method that allows real-time visual navigation based on sensory inputs from a stereo camera and an inertial measurement unit. We have used this method for GPS-free navigation in many of the environments explored and the scenarios examined in the field trials mentioned above. In particular we demonstrate the real-time 3D mapping of an entire shipwreck, which is one of the few deployments of its kind. This method and experiments were extended in a collaboration with my labmate, Travis Manderson, and published in [Manderson et al., 2016]. The method and the results are presented in Chapter 7.

1.4 Statement of Originality

Most of the chapters of this thesis contain work that has been published in robotics conferences, in collaboration with and under the guidance of my supervisor, Prof. Gregory Dudek. The major results from field trials presented in this thesis could not have been achieved by a single person, and several of my labmates have made key contributions: Anqi Xu, Juan Camilo Gamboa Higuera, Yogesh Girdhar, and Malika Meghjani were indispensable in the environmental monitoring experiments between a multi-robot team and a human, both in terms of the logistics of the experiment, but also in terms of infrastructure code for the communication of different vehicles. The visual and inertial navigation work was done in collaboration with my labmate, Travis Manderson, who contributed to the numerical optimization code and validation of the software in many different robotics platforms, ranging from aerial, to ground, to underwater. Travis' expertise in hardware and software

systems integration enabled a number of outdoor experiments, many of which fall outside the scope of this thesis. I would also like to acknowledge my labmate, Nikhil Kakodkar, who made key contributions to the pursuit via inverse reinforcement learning work. He implemented the Monte Carlo Tree Search baseline in the experimental validation and was also extremely patient in giving expert demonstrations on different maps, which formed the training set for our method. Finally, I want to acknowledge Wei-Di Chang and Peter Henderson, and Anqi Xu for software and hardware infrastructure and for experimenting with different tracking models in our joint robot convoying paper.

I also want to acknowledge many fruitful discussions with Prof. David Meger, with whom I have collaborated on many projects and papers not covered by this thesis. I also had many exchanges and research discussions with Prof. Joelle Pineau, Prof. Doina Precup, and Prof. Luc Devroye, about a variety of topics mostly related to inverse reinforcement learning and machine learning in general.

1.5 Outline

This thesis begins with an extensive literature review in Chapter 2 that covers multiple areas touched upon by the research described herein. The areas include reward learning, visual attention, visual tracking, pursuit evasion, and visual navigation. In Chapter 3 we present two major field trials that focus on collaborative visual exploration and informed videography between a human and a team of robots. These experiments fully motivate the algorithmic contributions and the choice of research directions pursued in this thesis. Chapter 4 describes an active learning method for learning user-specific saliency maps, which can be used for informed exploration. Chapter 5 presents tracking algorithms that learn such a behavior model for the target using inverse reinforcement learning, or use topological constraints to model plausible future behaviors. Chapter 6 presents results and theoretical analysis of pursuit methods that do not rely on a behavior model of the target being tracked, so they cannot make informed long-term predictions of future trajectories. Chapter 7 describes our localization and mapping algorithm that fuses stereo images and inertial measurements in a nonlinear optimization framework, which has provided the basis for the GPS-free, visual navigation capabilities of the underwater robotic vehicles presented in our experiments. Finally, this thesis concludes by outlining directions for future research that are attainable based on progress made by this work.

2 BACKGROUND

2.1 Sequential Decision Making Under Uncertainty

Throughout this thesis we rely on decision-making frameworks that model the robot as a rational agent with an objective trying to plan ahead and act optimally in the world, according to external feedback from the environment. These frameworks comprise Markov Decision Processes (MDP) [Bellman, 1957], Partially-Observable Markov Decision Processes (POMDP) [Sondik, 1978][Aström, 1965] and Non-Markovian Reward Decision Processes (NMRDP) [Bacchus et al., 1997].

2.1.1 Markov Decision Processes

An MDP is a tuple $\mathcal{M} = (S, A, T, r)$ where S denotes the set of states, A denotes the set of actions available to the agent, T denotes the stochastic dynamics model or transition probability distribution $p(s_{t+1}|s_t, a_t)$. $r(s_t, a_t, s_{t+1})$ denotes the instantaneous reward that the agent receives from the environment, depending on its objective, when the agent finds itself at state s_t, takes action a_t and ends up at s_{t+1}. This reward typically ignores the next state and has the form $r(s_t, a_t)$. It can be either deterministic or stochastic. A policy, representing the agent's strategy for choosing actions at a particular state, can be deterministic $a_t = \pi(s_t)$ or stochastic $\pi(a_t|s_t)$. The *value function* of state s induced by a particular stochastic policy and reward is

$$V^{\pi}(s) = \mathbb{E}_{s' \sim p(\cdot|s,\, a),\, a \sim \pi(\cdot|s),\, r \sim p(\cdot|s,a)} \left[r + \gamma V^{\pi}(s') \right] \tag{2.1}$$

The *state-action value function* of taking action a at state s and then following policy π is denoted by:

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}_{s' \sim p(\cdot|s,\, a),\, r \sim p(\cdot|s,a)} \left[r + \gamma V^{\pi}(s') \right] && (2.2) \\ &= \mathbb{E}_{s' \sim p(\cdot|s,\, a),\, r \sim p(\cdot|s,a),\, a' \sim \pi(\cdot|s')} \left[r + \gamma Q^{\pi}(s', a') \right] && (2.3) \end{aligned}$$

In the case of planning up to a finite time horizon these functions can be written as

$$V^{\pi}(s) = \sum_{t=0}^{T} \mathbb{E}_{s_t \sim d_{\pi}^{t}(\cdot),\, a_t \sim \pi(\cdot|s_t),\, r_t \sim \pi(\cdot|s_t, a_t)} \left[\gamma^t r_t \mid s_0 = s \right] \tag{2.4}$$

$$Q^{\pi}(s,a) = \sum_{t=0}^{T} \mathbb{E}_{s_t \sim d_{\pi}^{t}(\cdot),\, a_t \sim \pi(\cdot|s_t),\, r_t \sim \pi(\cdot|s_t, a_t)} \left[\gamma^t r_t \mid s_0 = s, a_0 = a \right] \tag{2.5}$$

where $d_{\pi}^{t}(s)$ is the distribution of states induced by the execution of the policy π, or in other words, the Markov chain with stochastic transitions $p(s_{t+1}|s_t) = \sum_{a_t \in A} p(s_{t+1}|s_t, a_t)\ \pi(a_t|s_t)$.

A policy π' is better than another policy π iff $\forall s \in S, V^{\pi'}(s) > V^{\pi}(s)$. The optimal value functions are

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{2.6}$$

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) = \max_{a} Q^{*}(s,a) \tag{2.7}$$

Planning in an MDP means finding the best policy for selecting actions at each state:

$$\pi^{*}(a|s) \propto \begin{cases} 1 & \text{if} \quad a = \operatorname*{argmax}_{a'} Q^{*}(s,a') \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

MDP planning algorithms such as value iteration [Puterman, 1994] rely on the following iterative update rule for the value function:

$$V_{k+1}(s) = \max_{a} \mathbb{E}_{s' \sim p(\cdot|s,\, a)} \left[r(s,a) + \gamma V_k(s') \right] \tag{2.9}$$

Their need to perform this iteration on all states does not scale to high dimensions due to discretization. The same issue plagues solutions based on linear programming [Guestrin et al., 2011]. Several methods address this problem, ranging from sampling-based methods such as Monte Carlo Tree Search [Kocsis and Szepesvári, 2006], and several variants of reinforcement learning with parametric policies [Sutton and Barto, 1998, Szepesvari, 2010, Bertsekas and Tsitsiklis, 1996, Mnih et al., 2013, Sutton et al., 1999].

2.1.2 Partially Observable Markov Decision Processes

A Partially Observable MDP is a tuple $\mathcal{P} = (S, A, T, r, O, Z)$ where all elements are the same as in MDPs, except Z, which is the set of possible observations and $O(z|s)$,

the observation model, i.e. the distribution of observations given the current state. The crucial difference between MDPs and POMDPs is that in MDPs the true state of the system is fully observed after each action. In POMDPs the state is a latent variable upon which observations z depend. The observable history of a POMDP is denoted by $h_t = (z_0, a_0, z_1, a_1, ..., a_{t-1}, z_t)$. The posterior over states given this history is called the belief over states $b(s_t, h_t) = p(s_t|h_t)$ and it can be computed recursively using Bayes Filters [Thrun et al., 2005].

$$b(s_t, h_t) \propto O(z_t|s_t) \sum_{s_{t-1} \in S} p(s_t|s_{t-1}, a_{t-1}) b(s_{t-1}, h_{t-1}) \tag{2.10}$$

A POMDP policy can be deterministic $a_t = \pi(h_t)$ or stochastic $\pi(a_t|h_t)$. The value functions corresponding to a history h_t induced by a particular stochastic policy and deterministic reward is

$$V^{\pi}(h_t) = \mathbb{E}_{\substack{s_t \sim b(\cdot, h_t),\, a_t \sim \pi(\cdot|h_t) \\ s_{t+1} \sim p(\cdot|s_t, a_t),\, z_{t+1} \sim O(\cdot|s_{t+1})}} [r(s_t, a_t) + \gamma V^{\pi}(h_t a_t z_{t+1})] \tag{2.11}$$

Finding the policy that has the highest value function in this case suffers from the curse of dimensionality (exponential state-space) and the curse of history. Some of the algorithms that address this problem are based on sampling and tree search [Ross et al., 2014, Silver and Veness, 2010, Shani et al., 2013b]. More recent methods rely on viewing planning as an inference problem and using variational inference to optimize the policy which is parameterized using [Igl et al., 2018].

2.1.3 Non Markovian Reward Decision Processes

Another decision theoretic framework for planning which stands in between MDPs and POMDPs are Non Markovian Reward Decision Processes (NMRDP). Similarly to MDPs they are a tuple $\mathcal{M} = (S, A, T, r)$ and the state of the system is observed at each step. The main difference with MDPs is that the reward function depends on the history of states and not only on the current state. This is a useful abstraction for modeling scenarios where the agent gets a reward only when a sequence of events or states are visited, and otherwise no reward is obtained [Bacchus et al., 1996]. The majority of existing planning methods for this category of decision processes traditionally relied on representing the sequence of states mentioned above in terms of Linear Temporal Logic [Emerson, 1990]. Converting the LTL specification into a new MDP with expanded state space allows the use of MDP planning algorithms to solve this problem [Bacchus et al., 1997], although this does not address the curse of

dimensionality. More recent methods have relied on optimizing a policy parameterized as a Recurrent Neural Network via recurrent policy gradients [Wierstra et al., 2010, Heess et al., 2015, Thiébaux et al., 2006].

2.2 Reward Learning

In all the decision theoretic frameworks outlined above the reward function of the agent was considered to be known (i.e. a model of it was given) or at least sampled from the environment. In many prediction problems we care about building a model of this reward function by observing the behavior of the agent, so that we can optimize and plan *on behalf* of the agent. Having access to the expert's reward function means at least two things: first, that a robot with a completely different mechanical embodiment can attempt to optimize that same function as the expert; second, that the optimization of the reward function can take place in states that were not part of the demonstrations, as long as they are representative of the task being performed.

Reward estimation has been studied from an engineering point of view as an instance of system identification, or *inverse optimal control*. It can also be seen from a machine learning perspective as a problem of estimating energy-based models [LeCun et al., 2006], learning the parameters of probabilistic graphical models, or learning the reward function assuming optimal behavior examples from a Markov Decision Process, whose dynamics and state space are known, but whose reward is not.

The first example of reward estimation was done by Kalman in the context of estimating the parameters of linear quadratic controllers [Kalman, 1964]. More recently, the problem of *inverse reinforcement learning* (IRL) was introduced in [Ng and Russell, 2000], in which we want to estimate the reward function of an MDP from a set of optimal trajectories (sequences of state-action pairs) that were generated from its optimal policy. IRL is clearly a degenerate problem in that there are many reward functions that would explain observed optimal behavior[1]. For example, multiplying any reward function by a positive scalar will not change optimal behavior for that MDP. Therefore, we need additional assumptions for constraining the estimated reward functions. It is worth mentioning that robustly estimating rewards and inspecting them properly has assumed a central role in discussions around AI safety [Amodei et al., 2016]. Given that reward estimation can be seen as a system identification problem, it is worth discussing issues of identifiability or lack thereof,

[1] Paul Christiano's [Christiano, 2018] and Jacob Steinhardt's blog [Steinhardt, 2018] on pitfalls and considerations for reward estimation are recommended reading.

and how to actively select how to test the behavior of the system in order to extract maximum information about its reward function [Amin and Singh, 2016].

Reward estimation in general and IRL in particular is not constrained to observing the behavior of a single agent in a vacuum. In cooperative IRL [Hadfield-Menell et al., 2016] for example a human has access to the true reward, but the robot can only estimate a distribution of reward parameters, and interacts with the human to reduce the uncertainty of this reward distribution.

Other instances of IRL learn rewards based on demonstrated behavior in POMDPs [Choi and Kim, 2011]. Related to this is the problem of *inverse planning*, typically used in cognitive science papers, where we for example try to estimate the navigation destination of a moving agent by observing its trajectory so far [Baker et al., 2009], under the assumption that they plan to move efficiently. Methods in this category are useful for predicting pedestrian behavior, both short-term and long-term [Karasev et al., 2016]. Problems like this also arise in psychology and economics, under the term *structural estimation*.

It is worth mentioning that instead of modeling and inferring the behavior and beliefs of agents using techniques from probabilistic graphical models, one can opt for more expressivity and represent agent behavior in terms of probabilistic programs[Evans et al., 2017, Roy, 2017]. They combine general-purpose programming languages and probabilistic graphical models, together with algorithms to perform efficient inference on the distributions resulting from program execution. For instance, one can represent uncertainty in planners, beliefs, state, rewards, and perform inference, all under one framework.

2.2.1 Inverse Optimal Control

In optimal control we are given a cost function and a set of constraints, such as dynamics, control limits, or state constraints, and we are asked the control vectors that will drive the system to minimize the cumulative cost over a period of time. In inverse optimal control we are given optimal trajectories of the system and the dynamics, and we are asked to infer the cost function. Such algorithms have been shown in the context of the Linear Quadratic Regulator [Kalman, 1964], Linear Quadratic Regulator with Gaussian noise [Chen and Ziebart, 2015], as well as in nonlinear optimal control problems [Englert et al., 2017, Puydupin-Jamin et al., 2012]. Some of this literature is broadly related to inverse optimization in economics, for example computing the cost function of linear programs given an optimal solution [Ahuja and Orlin, 2001].

2.2.2 Maximum Entropy Inverse Reinforcement Learning

In this framework the expert demonstrator is treated as planning in an MDP whose parametric reward function $r_\theta(s,a)$ is unknown, to be estimated from the demonstrated trajectories. We assume the existence of feature vectors $f(s) = [f_1(s), f_2(s), ..., f_N(s)]^\top$ where s is the discrete state of the system. For example, for navigation these N feature maps could be boolean images that indicate semantic labels over satellite maps, such as roads, vegetation, trees, automobiles, buildings, as well as dilations of these boolean maps. We also assume that the instantaneous reward is a linear transformation of the features, $r_\theta(s,a) = \theta^\top f(s)$, parameterized by the weight vector θ. Maximum Entropy IRL factors the probability distribution over trajectories $\tau = (s_0, a_0, s_1, ..., a_{T-1}, s_T)$ as an instance of the the Boltzmann distribution:

$$p(\tau|\theta) = \frac{1}{Z(\theta)} \exp\left(\sum_{t=0}^{T} \theta^\top f(s_t)\right) \tag{2.12}$$

where $Z(\theta)$ is the partition function that acts as a normalization term. In this formulation, paths that have accumulated equal reward will be assigned the same probability mass, and differences in cumulative reward will be amplified exponentially. This distribution arises from the constrained optimization problem of maximizing entropy subject to first-moment matching constraints, which specify that the expected feature count from trajectories drawn from the estimated distribution should be the same as the empirical mean feature count in the dataset S of demonstration trajectories. More formally, $p(\tau \mid \theta)$ is the solution vector to the following problem:

$$p(\tau|\theta) = \underset{q \in \mathcal{M}}{\text{argmin}} \; KL(q \;||\; \text{uniform}) = \underset{q \in \mathcal{M}}{\text{argmax}} \; H(q) \text{where} \tag{2.13}$$

$$\mathcal{M} = \{q : q \geq 0, \; \sum_{\tau} q(\tau) = 1, \; \mathbb{E}_{\tau \sim q}[f_i(\tau)] = \frac{1}{|D|} \sum_{\tau \in D} f_i(\tau) \;\; \forall i\} \tag{2.14}$$

In other words, it is the closest discrete distribution to uniform that satisfies the empirical features measured in the demonstration dataset D. It is worth mentioning that Eqn. 5.1 assumes deterministic dynamics for the system being modeled, as well as no noise in the observation of the demonstrated trajectories. Learning in the Maximum Entropy IRL model can be done by maximizing the log-likelihood

function of the trajectories in the demonstration dataset D, as follows:

$$\theta^* = \underset{\theta}{\text{argmax}} \sum_{i=1}^{|D|} \log p(\tau_i|\theta) \tag{2.15}$$

$$= \underset{\theta}{\text{argmax}} \sum_{i=1}^{|D|} \sum_{t=0}^{T_i} \left(\theta^\top f(s_t^{(i)}) - \log Z(\theta)\right) \tag{2.16}$$

Nonlinear Rewards & Reducing Assumptions Maximum Entropy IRL was extended to model deep, nonlinear rewards on a planar grid in [Wulfmeier et al., 2015]. It was later used in [Wulfmeier et al., 2017] to estimate traversability and driveability of terrain given trajectories from cars driving in different parts of Oxford. In this work the dynamics model is still assumed to be known and deterministic, and the states and actions are discrete.

Finn et al. [Finn et al., 2016b] extended the Maximum Entropy framework by assuming deep, nonlinear reward function, continuous states and actions, and unknown dynamics to be estimated from the training dataset. The major computational issue here is how to compute the partition function, which in high dimensions is intractable. In this work they resort to importance sampling for computing the partition function, and trajectory optimization to guide the proposal distribution for the importance sampling step. This method was demonstrated to work in manipulation scenarios, however, it is sensitive to the estimate of the partition function. This line of work was shown [Finn et al., 2016a] to have connections to GAIL [Ho and Ermon, 2016], because Maximum Entropy problems can be interpreted as trying to discriminate between trajectories that were demonstrated and trajectories generated by the current trajectory distribution or current policy estimate. This has close connections to the method of *contrastive divergence* in Conditional Random Fields and other similar probabilistic models, such as Restricted Boltzmann Machines.

Inverse Optimal Control is cast as policy search in [Doerr et al., 2015], so as to avoid the repeated invocation of an inner RL method inside the MaxEnt IRL reward update loop in Step 4. Specifically, this method uses black-box optimization in order to find the parameters of a policy that directly maximizes the similarity of policy-generated trajectories to the set of demonstrated trajectories (a stochastic objective). The advantage of this is that it is simple to implement, can handle continuous systems and nonlinear rewards. The inherited disadvantage of black-box optimization (Covariance Matrix Adaptation in their case) is that it is better suited for lower-dimensional systems, and could prove to be sample-inefficient.

Levine and Koltun [Levine et al., 2011] formulated the reward using Gaussian

Processes, thus modeling uncertainty around the average reward function, and allowing for continuous states and actions. In [Levine and Koltun, 2012] the same authors showed how the maximum likelihood problem over observed trajectories under a nonlinear reward could be formulated in terms of iterative linear approximations of the likelihood by linearizing the reward and the dynamics.

Boularias et al. [Boularias et al., 2011] targeted continuous states and actions, linear rewards, and stochastic dynamics. They also changed the MaxEnt optimization problem in 2.14, into the following problem, which is called Relative Entropy IRL:

$$p(\tau|\theta) = \underset{q \in \mathcal{M}}{\operatorname{argmin}} \; KL(q \;||\; \beta) \tag{2.17}$$

$$\mathcal{M} = \{q : q \geq 0, \; \sum q = 1, \; |\mathbb{E}_{\tau \sim q}[f_i(\tau)] - \frac{1}{|D|} \sum_{\tau \in D} f_i(\tau)| \leq \epsilon_i \;\; \forall i\} \tag{2.18}$$

where $\beta(\tau)$ is a distribution that favors trajectories that satisfy the dynamics, and the set of constraints allows for suboptimal demonstrations from the expert. In [Aghasadeghi and Bretl, 2011] the problem addressed involved systems with continuous states and actions, stochastic dynamics, and used the path-integral method [Theodorou et al., 2010] for forward reinforcement learning, which is based on sampling-based optimization.

Finally, Sermanet et al. [Sermanet et al., 2017] used reward functions whose features come from pretrained deep networks on ImageNet, and they identify the ones that explain a set of video demonstrations. Since the object being generated is now an image and not a trajectory, computing the partition function is intractable. To overcome this difficulty this work makes a very aggressive assumption which says that video frames are independent. This work offers a few heuristics to segment steps shown in the video, and it demonstrates that the learned reward is useful for reinforcement learning for tasks such as opening a door. This is however unproven in large environments.

2.2.3 Bayesian Inverse Reinforcement Learning

The majority of IRL problem formulations presented in the section above have been focusing on maximum likelihood or maximum-a-posteriori estimation to get a single hypothesis for a reward function that explains the observed trajectories. With the exception of [Levine et al., 2011] which relies on Gaussian Process, and Cooperative IRL [Hadfield-Menell et al., 2016], none of these methods provide measures of uncertainty or a sense of ambiguity about the estimated reward function. Given that one needs to take into account issues of identifiability of the reward [Amin and

Singh, 2016], because the demonstrations are not sufficiently rich, it is important to take a Bayesian approach to the reward estimation problem, and consider the full posterior of rewards that is consistent with the data and a reward prior.

Bayesian IRL [Ramachandran and Amir, 2007] was the first paper to introduce this approach to IRL. This work shows that both the problem of reward estimation and the problem of apprenticeship learning and policy search based on a distribution of rewards require computing the expected value of the reward's posterior distribution. The paper introduces a rapidly-mixing MCMC sampler for estimating this expected value, and shows that better estimation accuracy can be obtained by selecting good reward priors. The main assumption that enables all of this is that the distribution of a trajectory is factorized into independent state-action pairs, as follows:

$$p(\tau|\theta) = \prod_{t=1}^{T} p((s_t, a_t)|\theta) = \prod_{t=1}^{T} \frac{\exp(Q^*_\theta(s_t, a_t))}{Z_t(\theta)} \tag{2.19}$$

where $Q^*_\theta(s_t, a_t))$ is the optimal state-action value function, given a reward with parameters θ. This is similar to the aggressive assumption in [Sermanet et al., 2017]. Notice the difference between this distribution and the one in 5.1, where the partition function couples all the state-action pairs.

Bayesian IRL was extended to the case where demonstrations come from multiple reward functions [Choi and eung Kim, 2012]. The number of clusters of possible rewards was modeled according to a nonparametric Dirichlet process mixture model.

Risk bounds on the expected value difference between the any policy and the optimal policy, based on a distribution of rewards, are given by [Brown and Niekum, 2017]. In this Bayesian IRL method, multiple possible rewards are sampled from the posterior, in order to evaluate the risk. In addition, this work provides risk-aware policy iteration and policy selection algorithms. The main drawback is that, like most Bayesian IRL methods, it requires a planning/RL step inside the reward update loop in order to compute $Q^*_\theta(s_t, a_t))$, which is inefficient.

Finally, a paper that is related to Bayesian IRL is Inverse Reward Design [Hadfield-Menell et al., 2017]. In this work the specified reward for an MDP world is treated as a proxy of the true reward that the user intended. Similar to reward shaping, methods, a proxy reward can be a better choice for more efficient planning. It can also be easier to specify than the true reward. This paper treats proxy rewards as observations of the user's intended reward and associates them with the MDP world in which they were trained. This means that if the agent is tested on an MDP world

that differs from where it was trained[2], its reward function will have to be treated as uncertain and planning might need to be risk-averse.

2.2.4 Task-Level Inverse Reinforcement Learning

In many demonstration settings the process being showcased involved multiple sub-tasks whose execution completes the main task of interest. Cooking with the guidance of a recipe, for example, involves a sequence of such sub-tasks. Changing from the leftmost lane to the rightmost on a highway also involves a series of sub-tasks. Ideally, the demonstrator would not have to provide a demonstration for each subtask and then specify the order in which they should be executed; the robot should be able to infer them from a single demonstration. In other words, the robot should be able to do *sub-task segmentation* given a demonstration trajectory, and learn both high-level behaviors and low-level sub-task-dependent rewards.

Niekum et al. [Niekum et al., 2015, Niekum et al., 2013] address the automated segmentation of trajectories using a Beta Process Autoregressive Hidden Markov Model, which is a nonparametric hierarchical generative model that can represent infinitely many sub-tasks, and switches among them. These sub-tasks are clustered into states in a finite-state machine which describes the high-level behavior of the system. Dynamic Movement Primitives are used to control motion between states in the finite state machine. In this method additional demonstrations are requested when transitions in the finite state machine are uncertain. The method is demonstrated in furniture assembly tasks.

Similar ideas on sub-task segmentation appear in [Krishnan et al., 2017], where the objective is to identify transition states, where trajectories are generated from a switching linear dynamical system. The sub-tasks are determined by a Dirichlet Process that affects a set of Gaussian Mixture Models. The difference with Niekum et al. [Niekum et al., 2015, Niekum et al., 2013] is that the state of the system includes both kinematic and visual features from deep neural networks. The main application examined is robotic surgery. This technique was extended to deal with delayed rewards in Sequential Windowed IRL [Krishnan et al., 2016] where trajectory segmentation and learning of sub-tasks is followed by learning sub-task-specific rewards using either model-based MaxEnt IRL 5.1 or model-free cost-function learning in Linear Quadratic Regulators. Finally, when the sequence of sub-goal-specific rewards is learned, this method invokes Q-learning to learn a policy that will maximize the learned sequence of rewards. This method was evaluated on simulated parallel parking and real surgical robotics tasks.

[2]The main example in the paper involves testing the agent on MDP grid world where there is lava, while all the demonstrations came from environments where no lava existed.

It is also worth mentioning that we can frame the joint problem of task and motion planning, or hybrid (discrete and continuous) control, in terms of context-free grammars. These grammars are more expressive than deterministic finite-state automata and Turing Machines. They involve a sequence of production rules and terminal symbols that can be used to generate structured language. For example, in [Dantam and Stilman, 2012] they were used to represent policies for task and motion planning problems in robotics. Doing IRL while at the same time fitting a policy that will respect the context-free grammar is still very much an open research problem. This of course has many connections with imitation learning based on learned program representations [Xu et al., 2017, Reed and de Freitas, 2015].

2.2.5 Query-Efficient Reward Learning

In the previous sections the majority of papers formulated the reward estimation problem in terms of learning in probabilistic graphical models. This lead to issues regarding the estimation of partition functions and in many occasions lead to having to invoke an RL method inside the IRL reward update loop, which is inefficient.

Sadigh et al. [Sadigh et al., 2017] take an energy-based method approach, by trying to directly estimate the reward parameters using active supervised learning. Their main application task is learning a person's driving style as a linear reward over a set of fixed features. The contribution of the paper is that active learning can be used to generate *driving simulation scenarios* that will provide the most information about the human's driving style. The user provides preference feedback, by choosing which generated trajectory they prefer. This ends up being quite query-efficient, however, the simulation scenarios are limited in their fidelity to the real world. Nevertheless, this is a very promising approach. This work was extended in [Basu et al., 2018] to allow users to express preferences over binary comparisons between a queried pair of trajectories, but also to express which feature was most responsible for their binary preference. This work shows that actively choosing comparison-feature queries on pairs of trajectories, such that they will provide maximal information about the user's reward function, converges faster to the reward parameters than simply using comparison queries, as in [Sadigh et al., 2017]. The main limitation of this work is that the set of features being considered is fully interpretable and tailor-made for the scenario of inferring driving styles in a low-fidelity simulator.

Active learning for Bayesian IRL has been addressed in [Lopes et al., 2009]. The learning method in this work asks the human for another action demonstration

at the state where the entropy/uncertainty of the estimated policies, from a set of rewards sampled from the posterior, is maximal.

Other approaches, such as Maximum Margin Planning [Ratliff et al., 2006] are query-efficient without resorting to active learning. This method learns cost maps for traversability and navigation from a set of features, as well as value functions for navigation. It estimates a linear reward over those features by trying to match either trajectories or state-action frequency counts from the demonstration dataset, and also a value function based on that reward. The problem is setup as a max-margin quadratic programming problem, and an efficient sub-gradient optimization method is proposed to solve it. The main limitation of this method is that it operates on a discretized state-action space and its constraints scale linearly with the number of state-action pairs and training examples.

Another approach for estimating the expert's reward function, distinct from most of the existing literature in IRL, can be found in [Metelli et al., 2017]. This work considers the policy parameters that make the policy gradient zero, to find potential maxima, minima, or saddle points of the expected cumulative reward objective. This leads to finding feature maps that can be used to linearly represent the expert's reward function, thus leading to automatic feature construction. Then the method computes the Hessian of the expected cumulative reward objective to find the maxima among the critical points, and selects the reward that least deviates from the expert's demonstrations. This method however, has been only validated on low-dimensional systems.

2.2.6 Preference Elicitation

Preference elicitation [Braziunas, 2006, Furnkranz and Hullermeier, 2010] is the problem of recovering the latent ranking, or relative weighting, that a user places on a set of available items/features. We already saw examples of preference-based reward learning in [Sadigh et al., 2017, Basu et al., 2018], but a similar method for preference-based Bayesian IRL can be found in [Rothkopf and Dimitrakakis, 2011].

The general version of the problem also includes applications in combinatorial auctions [Blum et al., 2003, Sandholm and Boutilier, 2006, Braziunas and Boutilier, 2010, Boutilier, 2002], where the auctioneer needs to estimate the valuations of each of the bidders before setting the price of the items of interest. There are close ties between designing queries for preference elicitation and the field of optimal experiment design.

One of the challenges of using preference-based learning to estimate rewards is that the user's answers might be inconsistent over time or violate transitivity.

In [Evans et al., 2016] the authors deal with such inconsistent observations by modeling the agent via a probabilistic program, in order to model sub-optimality and noise at different steps of the planning process. A distribution of possible rewards can be recovered by doing Bayesian inference.

2.3 Visual Attention and User Specifications

2.3.1 Bottom-Up Attention Models

Computational models of visual attention in robotics promise to increase the efficiency of visual search and visual exploration by enabling robots to focus their field of view and sensory stream towards parts of the scene that are informative according to some definition [Frintrop et al., 2010]. So called *bottom-up* attention methods define an image region as salient if it is significantly different compared to its surroundings or from natural statistics. Some of the basic features for bottom-up attention have included intensity gradient, shading, glossiness, color, motion [Frintrop et al., 2010, Tsotsos, 2011, Zhaoping, 2014] as well as mutual information [Bruce and Tsotsos, 2008, Gottlieb et al., 2013].

2.3.2 Top-Down Attention Models

The ability of bottom-up features to determine and predict fixations is not widely accepted [Underwood et al., 2006], especially when a top-down task is specified, such as "find all the people in the scene". *Top-down* attention models are task oriented, with knowledge coming externally from a user that is looking for a particular object. Some of the first attention models to have combined these two types of attention are: Wolfe's Guided Search Model [Wolfe, 1994] which comprises a set of heuristics for weighing bottom-up feature maps, and the Discriminative Saliency Model [Gao and Vasconcelos, 2005], which defines top-down cues as the feature maps that minimize the classification error of classes specified by the user. There's also the Contextual Guidance Model [Oliva et al., 2003, Torralba et al., 2006] uses the gist descriptor [Oliva and Torralba, 2001] that provides a summary of the entire scene to guide the set of possible locations where the desired target might be. Finally, there's the Selective Tuning Model [Tsotsos, 1995], which relies on a hierarchical pyramid of feature maps, the top of which is biased or determined by a task and the lower levels of which are pruned according to whether they contribute to the winner-takes-all or soft-max processes that are applied from one level of the hierarchy to the next. Notably, it does not only operate in a feedforward fashion.

2.4 Visual Exploration

2.4.1 Why Should Robots Explore?

In many cases exploration is treated as a secondary task that aims to reduce the robot's uncertainty in order to facilitate a primary task. For example, exploration in reinforcement learning is typically based on taking actions that aim to reduce the entropy of the distribution of possible dynamics models, either internal to the robot, or also external to the physics of environment [Houthooft et al., 2016], or value functions [Osband et al., 2016, Lipton et al., 2016]. In other words, exploration is seen as active learning in order to reduce the space of possible models and get closer to estimating the true one. In multi-task learning robots often explore to learn useful behaviors independent of the particular goal [Eysenbach et al., 2018]. Robots perform *frontier-based exploration* [Yamauchi, 1997] in order to reduce their uncertainty about the true map of the environment [Sim et al., 2004], and eventually cover it completely.

2.4.2 What Should Robots Explore?

Rather than treating visual exploration as a secondary task, we want to give it first-class status and reward the robot for recording images that are important to the user in addition to being representative of the environment. Combining bottom-up and top-down attention mechanisms for the purpose of getting the user the data they need is one of our objectives in this thesis. Top-down task specification expresses the user's evolving preferences about what kind of visual content is desired. We treat this bottom-up and top-down visual attention model as a user-tunable *reward function/saliency map* for visual content that guides the exploration of the robot so that it records more footage of scenes that are deemed important by the user. Not all visual exploration strategies obtain user input. Many of them focus the camera towards parts of the scene that are surprising with respect to a summary of the history of observations [Girdhar and Dudek, 2011, Girdhar and Dudek, 2012, Girdhar and Dudek, 2014a, Paul et al., 2014]. We argue in this thesis that exploration should be combined with visual search and the robot should balance surprise-based exploration with recording data that is going to be useful to the user.

2.5 Pursuit and Tracking

2.5.1 Pursuit Evasion Games

In order to enable systems for human-robot collaborative visual exploration, one essential element is to allow a robot to robustly track the human collaborator in the presence of misdetections, lack of line-of-sight, noise and occlusion.

This problem is related to the large body of existing work on the theoretical bounds for maintaining continuous surveillance of a moving target. This problem has received extensive consideration in the literature of pursuit-evasion games [Isaacs, 1969]. For most work in this category, issues related to the optimization of visibility based on the structure of the environment are critical, so there are natural connections with the classic "art gallery" problem and its variants [O'Rourke, 1987].

Generally, continuous visibility maintenance has been considered mostly for the case of adversarial targets. Such is the case for example in [Sourabh Bhattacharya, 2008, Bhattacharya and Hutchinson, 2009, Isler et al., 2005, Stiffler and O'Kane, 2012]. The question of identifying whether a target is being cooperative or evasive and modifying the pursuer's plans in response to this degree of information is something that currently lacks a precise characterization in the literature. It is worth mentioning nevertheless, that if the target is adversarial there exist settings under which the evader can escape the follower's field of view, no matter what the follower does [Sourabh Bhattacharya, 2008, Bhattacharya and Hutchinson, 2009].

The tracking problem has also been addressed by approximate sampling-based POMDP solvers such as SARSOP [Hsu et al., 2008], which extends a Monte Carlo Search Tree and maintains both the upper and lower bounds of the state-action value function, which enables pruning of provably suboptimal actions. The proposed method, however, is demonstrated to work on small grids and runs in the order of minutes.

In [Lavalle et al., 1997] the problem of planning the trajectory of a single observer trying to maximize visibility of a fully-predictable or partially-predictable target is considered. The problem setting is augmented in [Murrieta-Cid et al., 2005] which examines planning the paths of several observer robots whose goal is to provide continuous visual coverage of several unpredictable targets.

2.5.2 Vision-Based Tracking

The extensive literature on visual tracking can be separated into *model-based* and *model-free* methods, each with their own set of advantages and drawbacks. In model-free tracking the algorithm has no prior information on the instance or class of objects it needs to track. Algorithms in this category, such as [Yu et al., 2008], are typically initialized with a bounding box of an arbitrary target, and they adapt to viewpoint changes through weakly-supervised learning. The TLD tracker [Kalal et al., 2012], for example, trains a detector online using positive and negative feedback obtained from image-based feature tracks. In general, tracking systems in this category suffer from *tracking drift*, which is the accumulation of error over time either from false positive identification of unseen views of the target, or errors due to articulated motion, resulting in small accumulating errors leading to a drift away from the target object.

In model-based tracking, the algorithm is either trained on or has access to prior information on the appearance of the target. This can take the form of a detailed CAD model, such as in [Manz et al., 2011], which uses a 3D model describing the geometry of a car in order to improve tracking of the vehicle in image space. Typically, line and corner features are used in order to register the CAD model with the image. We want to avoid these methods because of their susceptibility to errors in terms of occlusion and non-rigid motion.

Works such as [Bolme et al., 2010, Nam and Han, 2016] use convolutional neural networks and rely on supervised learning to learn a generic set of target representations. Template tracking methods, such as [Comaniciu et al., 2003, Yang et al., 2009] model the object with a (static or dynamic) target template and compute the motion as the transformation which minimizes the mismatch between a new candidate patch and the last template. Generative methods have been proposed in order to model more appearance variations of the object. In the last few years at the Visual Object Tracking (VOT) Challenge, the top performing trackers use correlation filters as well as convolutional neural networks to extract features.

2.6 Robotic Platforms

2.6.1 Aqua Amphibious Robot

Aqua, shown in Fig. 2.1, is a six-legged amphibious robot that can both swim underwater and walk on land. It maneuvers in water by actuating its six flippers, and its aluminum shell is designed to operate at depths up to 40 m. It is powered by

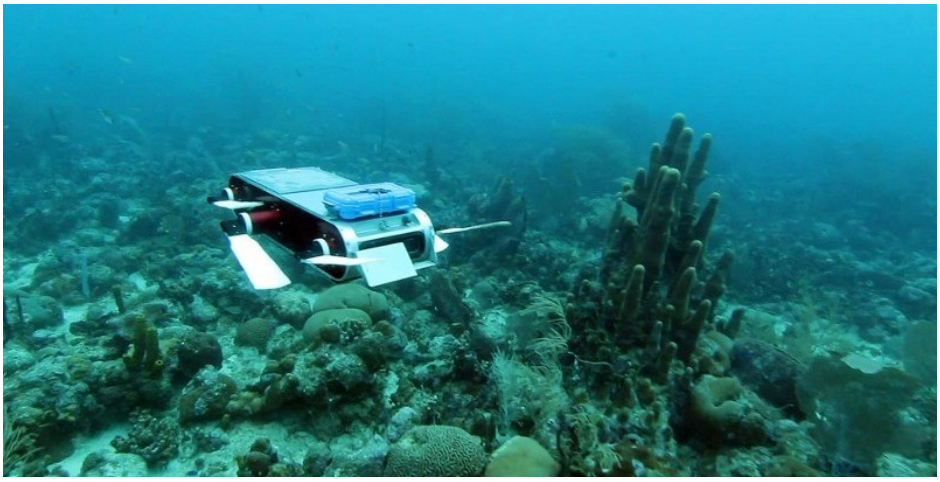


FIGURE 2.1: The Aqua amphibious robot navigating over a coral reef.

high-capacity Lithium-Ion batteries, and can operate under full load for more than five hours underwater.

Aqua is equipped with a variety of sensors within its waterproof shell, which includes: three USB cameras (two front-facing and one at the back), an inertial measurement unit (IMU), a pressure-based depth sensor, and an XBee digital radio transceiver. The two front-facing cameras are used for visual navigation and tracking, while the back camera is typically used for human-robot interaction with the diver. We also equipped it with an externally-mounted sensor kit to facilitate wireless communications, since the AUV's metallic shell acts as a Faraday cage and therefore highly attenuates the transmission of radio signals. This detachable and self-powered sensor kit contains a GPS unit and an XBee module, and augments Aqua's sensing capabilities by relaying GPS data and messages from the MARE surface vehicle using the XBee communications link. Sensor processing and high-level planning is carried out by an embedded computer which interacts with another computer that regulates the six leg motors in a real-time operating system.

2.6.2 Unicorn Unmanned Aerial Vehicle

The Unicorn UAV is a kite-sized fixed-wing aerial vehicle. This robot has a 1 m wingspan and is built from expanded polypropylene (EPP) foam, which is used to absorb and dissipate the collision force during touchdown. The Unicorn's brushless motor is powered by Lithium Polymer batteries, which allows the vehicle to operate at average ground speeds of 14 m/s for up to 30 minutes of flight time.

This UAV is equipped with multiple sensors, including an IMU, a GPS, and pressure-based altitude and speed sensors. These devices are integrated with the on-board autopilot micro-processor, which uses them to navigate autonomously

FIGURE 2.2: The Unicorn is a fixed-wing UAV with an on-board autopilot microprocessor and a gimbal-mounted camera.

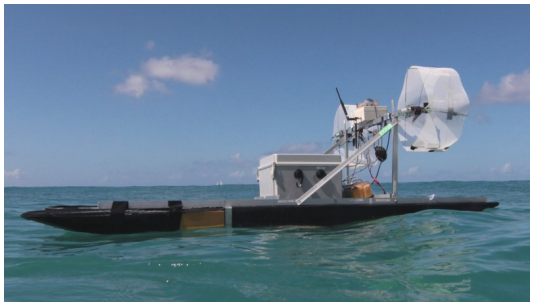


FIGURE 2.3: The Marine Autonomous Robotic Explorer (MARE) is a catamaran robotic airboat that can operate in turbulent open water environments.

based on waypoint directives issued from the home base. Communication between the autopilot and the home base is established via radio frequency using a high-power XTend modem; this allows the UAV to be controlled at multi-kilometer ranges. The Unicorn is also equipped with a CCD camera mounted on a pan-tilt gimbal, which allows the home base to receive live aerial feed through an analog radio frequency channel.

2.6.3 MARE Autonomous Surface Vehicle

The Marine Autonomous Robotic Explorer (MARE) [Girdhar et al., 2011] is a robotic airboat developed to explore coral reefs and shallow seabeds. Its open catamaran, twin-pontoon design provides sufficient hydrodynamic stability to operate in turbulent open water environments. This vehicle is actuated using air propellers in a differential drive configuration, which is preferred over water-based propulsion mechanisms as it causes less underwater disturbance to shallow reefs. MARE's motors are powered by Lithium Polymer batteries, which provides over two hours of continuous operations.

Most of the electronics on-board are stored inside a large water-proof enclosure at the center of the chassis. The primary computing unit is a netbook computer that is responsible for interfacing with the motor micro-controller, powering and collecting data from sensors, and managing high-level autonomous behaviors. MARE is a vision-centric platform and is equipped with a high-definition downward-facing camera. It also uses an IMU and a GPS device to track its location and pose in the water. Furthermore, MARE is capable of communicating over a variety of channels, including a WiFi link used to stream video, low-power XBee and long-range XTend radio transceivers used to relay information between the Unicorn UAV and Aqua, and an analog transceiver that enables manual tele-operation at multi-kilometer distances.

3 COLLABORATIVE HUMAN-ROBOT ENVIRONMENTAL MONITORING

This chapter includes two large scale field robotics experiments which both demonstrate and at the same time motivate the entire set of problems chosen in this thesis. Both of these experiments show how a robot or a multi-robot team can act in collaboration with human scientists to help them collect useful data on a larger scale than what is possible through manual data collection, as shown in Fig. 3.1.

The first experiment describes how a team of heterogeneous robots can help a marine biologist collect the underwater data that she needs without requiring her to dive. The second experiment shows how an underwater robot can follow a diver around, while at the same time directing the camera towards interesting parts of the scene, according to a user-specified model. In the next chapters we will describe theoretical and practical improvements on many aspects of these systems.

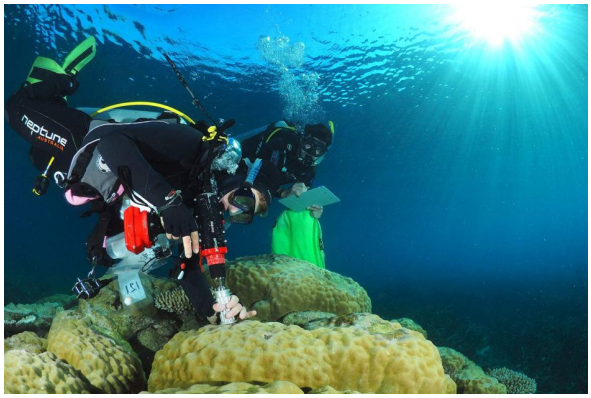


FIGURE 3.1: Two divers, one manually taking notes about a single coral head, the other taking samples. At the Great Barrier Reef.

3.1 Robot Team Explores On Behalf Of Scientists

In this section we present a multi-robot system developed for environmental monitoring and surveying, which operates in the aerial, water surface, and underwater domains. Our primary objective is to use autonomous robots to perform environmental monitoring, for instance by helping marine scientists conduct repeated inspections in a consistent, efficient, and comprehensive manner. To this end, we have deployed a heterogeneous robot team that interacts with human experts through

the Internet to identify areas of interest based on live aerial feedback. The team then autonomously collects visual footage of these areas at different scales and from multiple domains.

We are especially motivated to help marine biologists survey the long-term health of coral reefs. Coral reefs are extremely precious ecosystems: they occupy less than 0.1% of the world's ocean surface, yet they provide a habitat for 25% of all marine species [Mulhall, 2007]. Unfortunately, coral reefs have been decaying at an alarming rate in recent decades [Hodgson and Liebeler, 2002], and there has been a long-standing need among scientific communities to identify methods for their preservation [Rogers et al., 1994].

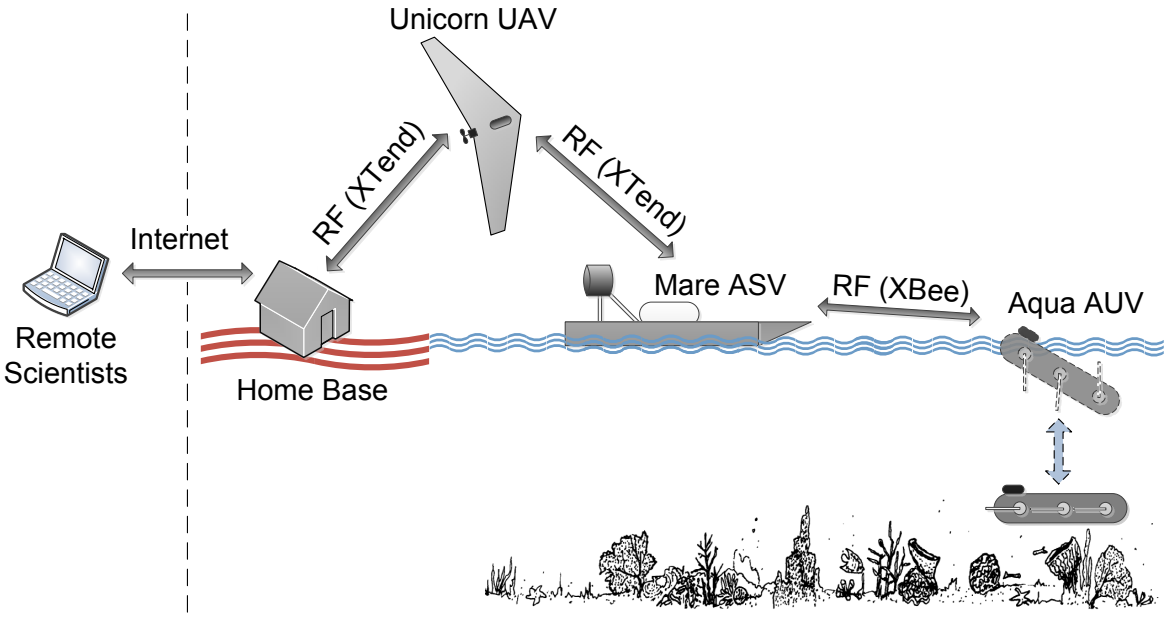


FIGURE 3.2: Our heterogeneous multi-robot system used for the monitoring of marine environments in collaboration with remote scientists.

Coral reefs are monitored conventionally by scientists, who must dive to reef sites on a regular basis to inspect their health visually. This approach is laborious and slow, since the scientists have to travel potentially large distances between live coral sites. A common complementary strategy is to plan each dive beforehand by identifying potentially rich reef patches based on aerial footage [Rogers et al., 1994], although its effectiveness depends on having up-to-date satellite imagery with sufficient visual resolution and clarity.

We address the drawbacks of conventional reef monitoring methods by automating the data collection process using a team of robots, which is comprised of an autonomous flying vehicle, an autonomous surface vehicle, and an autonomous underwater vehicle. During a typical monitoring session, our flying vehicle continuously collects aerial footage of a reef region and relays them to scientists via the Internet. Based on this live footage, the scientists then suggest sites for further

inspection to the team of robots, which subsequently coordinate with each other to autonomously collect visual footage of these target regions. Our system thus provides comprehensive visual coverage of the specified reef sites, both from a broad aerial view and a close-up underwater view. This autonomous robotic team also significantly improves the efficiency of the monitoring process, by shifting the scientists' focus from laborious data collection to the selection of survey regions.

This is the first human-robot coordination system where a remote expert (marine biologist) guides in real time a team of heterogeneous robots operating in air, sea surface and underwater, to inspect the health of a coral reef. Over 18 kilometers of flight and 1.5 kilometers of underwater traverses validate the robustness of our approach. In the sections below we will elaborate on individual components in this integrated multi-robot system, along with other important aspects of the marine monitoring tasksuch as scheduling and coordination of various roles occupied by the human and robot participants. We then discuss our field trial results, which demonstrate the efficiency of this multi-robot system for marine biologists to study coral reefs.

3.1.1 Heterogeneous Multi-Robot Team

This team of robots operates in a top-down hierarchical structure that arises due to the diversity in each vehicle's operational range, and the need to communicate with remote scientists and the home base. This hierarchy, depicted in Fig. 3.2, is also beneficial to the monitoring process: the flying vehicle can obtain coarse visual coverage of the entire reef region quickly, which allows the surface vehicle and underwater robot to then focus on inspecting the most potentially interesting subregions at a finer resolution, while minimizing the time spent traveling over less important regions.

- The Unicorn plane is responsible for performing coverage of the entire survey region and provide up-to-date large-scale aerial imagery to the remote human scientists. After obtaining the expert-selected inspection sites, the plane then re-broadcasts these waypoint directives to the other two robots underneath, while continuing to collect updated aerial footage of the entire region.

- The MARE boat serves two primary roles: it is used to cache waypoint directives received from the Unicorn plane, and it also relays these messages to the Aqua underwater robot when it surfaces. These roles are needed because the plane has limited battery life, making it unable to wait until the Aqua underwater robot surfaces.

- The Aqua [Sattar et al., 2008b] underwater robot is responsible for gathering fine-scale imagery by performing close-up inspection of the target sites. While Aqua mainly operates underwater to navigate to these sites and collect footage, it also regularly ascends to the surface to listen for further messages from MARE and to update its localization using GPS.

3.1.2 Aqua: Control and Porpoising Motion

Aqua's motion is regulated by an autopilot that allows both attitude control and depth control based on the IMU and depth sensor readings. For the purposes of our marine monitoring task, we deployed a GPS-based waypoint follower on top of the autopilot. This navigation system iteratively implements the following sequence of actions: ascend to the surface, collect GPS position estimates, descend to a fixed depth, adjust bearing towards current waypoint, and move forward for a given amount of time.

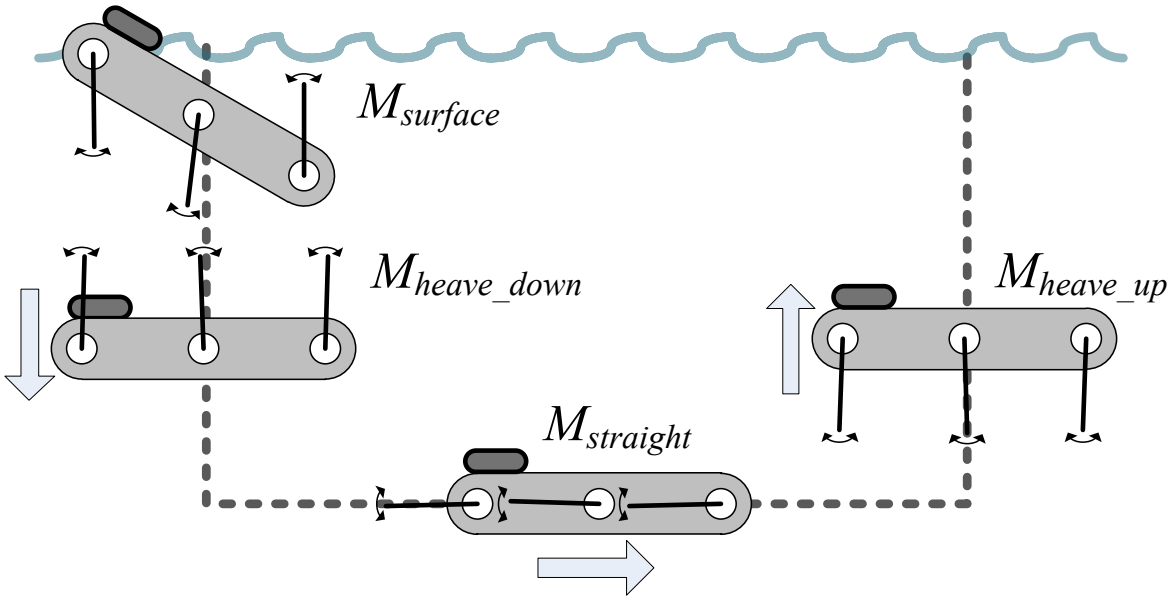


FIGURE 3.3: The four states of the *porpoising motion* for the Aqua underwater robot.

The combination of ascending, descending, and forward movement constitutes the *porpoising motion* of the Aqua robot. This navigation strategy was chosen to address several fundamental challenges imposed by the underwater environment on existing long-range localization algorithms. For instance, vision-based techniques for position estimation must cope with the possibility of severely limited underwater visibility, whereas acoustic localization methods would need to address multipath interference issues when operating in shallow underwater environments. We

therefore chose the porpoising motion and GPS localization for increased reliability. To enable this motion, we configured the autopilot to implement the following three modes:

- $M_{straight}$: straight forward motion at a fixed depth;
- M_{heave}: controlled heave for ascent and descent; and
- $M_{surface}$: sustained backwards pitch at 45 degrees to elevate the external sensor kit above the sea surface.

The autopilot implements proportional (P) controllers for the pitch and roll axes, and a proportional-derivative (PD) controller for the yaw axis. The latter derivative gain is essential to counteract the slow response time of this vehicle about the yaw axis [Giguere et al., 2006].

While executing $M_{straight}$, the autopilot maintains a straight trajectory along a fixed heading angle at constant depth by regulating the roll angle about 0° and the pitch angle based on the target depth. The M_{heave} and $M_{surface}$ modes rely on a hovering gait described in [Sattar et al., 2009], in which the roll and pitch gains are downscaled from $M_{straight}$ by a factor of ten to minimize oscillatory behaviors.

Since sea water is conductive, it is infeasible to receive GPS signals underwater. This was corroborated through empirical studies where we determined that less than 1 cm of seawater over a 12 cm^2 patch GPS antenna is sufficient to preclude GPS reception Thus, Aqua's body must be inclined backwards to elevate the GPS antenna above the water surface. The $M_{surface}$ mode implements this behavior by combining a target pitch angle of 45° with an upwards heave command. In this mode, Aqua's four back flippers are oriented downwards and oscillate with a large amplitude, while its two front flippers are pointed upwards with minimal oscillation. This results in both a net positive force and a net moment, which counteracts forces due to the back of the robot being raised above the water surface.

3.1.3 Interaction with Marine Scientists

This multi-robot marine monitoring system incorporates input from marine biologists, who are responsible for identifying marine sites that are worth inspecting at a finer scale, based on the coarse-scale aerial footage captured by our plane. These scientists, however, are not responsible for direct tele-operation control over any of the robots; robot navigation and the data collection process are both carried out autonomously. We developed a web interface that allows scientists off-site to remotely monitor aerial images obtained by our plane and mark points of interest on individual images. These coordinates are transmitted as target GPS waypoints to the

home base, and then broadcasted sequentially to the Unicorn plane, to the MARE boat, and finally to the Aqua underwater robot. This web interface allows scientists to interact with our robot team within the visual task domain while concealing underlying communication and control aspects, to ensure a smooth and intuitive user experience.

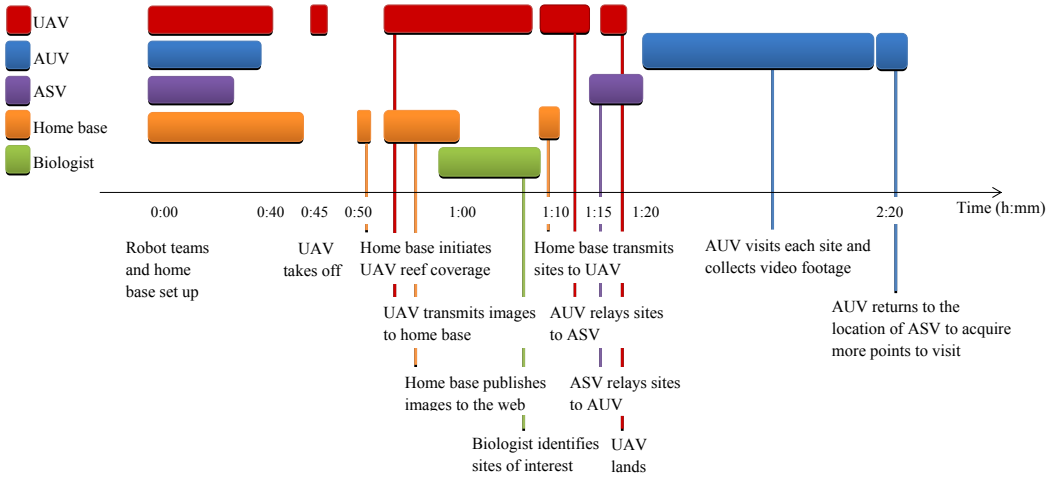
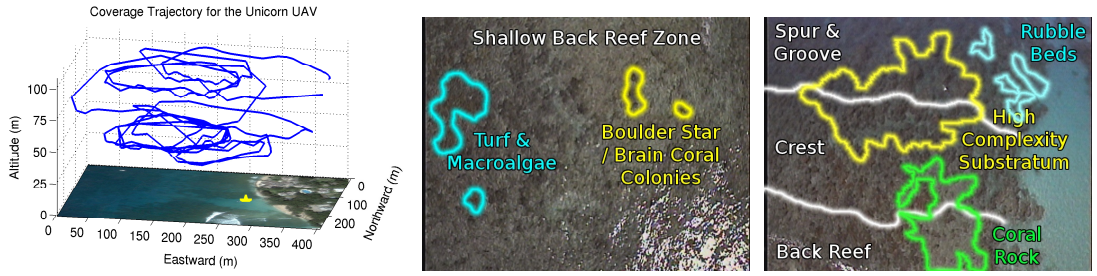


FIGURE 3.4: Event timeline for our muli-robot coral reef monitoring sessions, lasting 2 hours and 30 minutes on average. Abbreviations: Umanned Aerial Vehicle, Autonomous Surface Vehicle, Autonomous Underwater Vehicle. Note: timeline is not to scale.

3.1.4 Coral Reef Monitoring Trials

We conducted an extensive field trial comprised of multiple coral reef monitoring sessions, to evaluate the feasibility of our approach and to optimize system parameters. Fig. 3.4 depicts a simplified timeline for each monitoring session, starting with the launch of the Unicorn plane and ending with the Aqua underwater robot visiting and recording underwater footage at a number of sites of interest within a tropical reef region. These sites were chosen by a biologist located about 4,000 km away from our reef region, who interacted with our system in real time through a web interface.

Each monitoring session lasted over 2 hours long, though a significant portion of that time was devoted to setting up the various robots at distinct operational sites, both on land and at sea. Each vehicle is supported by a small team of roboticists with the exclusive purpose of passively monitoring their operations during each session. Furthermore, given the complexity of communications arising from the nature of the experiment, a home base team was deemed necessary. Members of the home base team were responsible for monitoring the timing of the entire experiment and informing other teams of event progressions.



(a) (b) (c)

FIGURE 3.5: (a) The Unicorn plane carried out aerial coverage of a 150 m × 100 m reef region at both 100 m and 50 m altitudes. Star icon depicts location of the MARE boat. (b) The resulting images depict high resolution reef views, which enabled our expert biologist to identify diverse coral colonies and other areas of interest, and to estimate the relative depth of the reef, based on texture and color cues. (c) Frames acquired at 50 m provided close-up views while still being able to cover all three zones found on fringing reefs, namely *spur and groove*, *crest*, and *back reef*; the high complexity substratum area, characterized by strong variations in color and texture, is an especially interesting site to inspect from a biologist's perspective.

The results presented in this section are from a single marine monitoring session, in which our team of robots operated within a 150 m × 100 m coral reef region. Although this operational area may be considered small according to some marine monitoring standards, the current range of our system is predominantly limited by the plane's operational capabilities, and hence can be easily resolved by upgrading to more efficient and higher capacity batteries, or switching to a more capable aerial vehicle.

We recorded both visual data and state information from all three vehicles, to allow the team's performance to be benchmarked and improved upon. Two important metrics used in the evaluation of our field trials consist of the durations for each phase in the reef monitoring process, and the distances traversed by the three robots.

Our monitoring session began with the launch of the Unicorn plane, which was directed to cover the entire reef region in a circular orbit. The plane flew at speeds between 10 m/s and 20 m/s in moderately turbulent wind conditions, and carried out aerial coverage first at 100 m and subsequently at 50 m, as shown in Fig. 3.5(a). Aerial footage from the plane was being continuously streamed to the home base through a radio frequency channel, and subsequently relayed to our web server. Therefore, this allowed the remotely-located biologist to inspect the entire reef region using aerial views at two different scales, within 15 minutes following the launch of the plane.

The scientist identified 4 sites of interest for up-close inspection after analyzing the aerial views for 10 minutes, and issued the corresponding waypoint directives to the home base using the web interface. These directives were then transmitted to the plane, which began re-broadcasting the commands repeatedly while orbiting above the boat in its coverage of the reef region. The boat received all four target locations, after 25 minutes since the launch of the plane. After completing both of its tasks of providing aerial footage of the coral reef and relaying waypoint directives from the biologist, the plane proceeded to land while delegating the rest of the monitoring workload to the other two robots.

The boat was operating as a caching station and relay unit, by re-broadcasting the target locations that it had received from the plane continuously until the underwater robot had confirmed their reception. Upon receiving the target locations, the underwater robot activated its GPS waypoint follower to visit them one-by-one, navigating via the porpoising motion to each of the specified waypoints. Through the porpoising motion, the underwater robot's piecewise linear path incorporated both localization and forward motion. The resulting trajectory executed by the underwater robot is shown in Fig. 3.6.

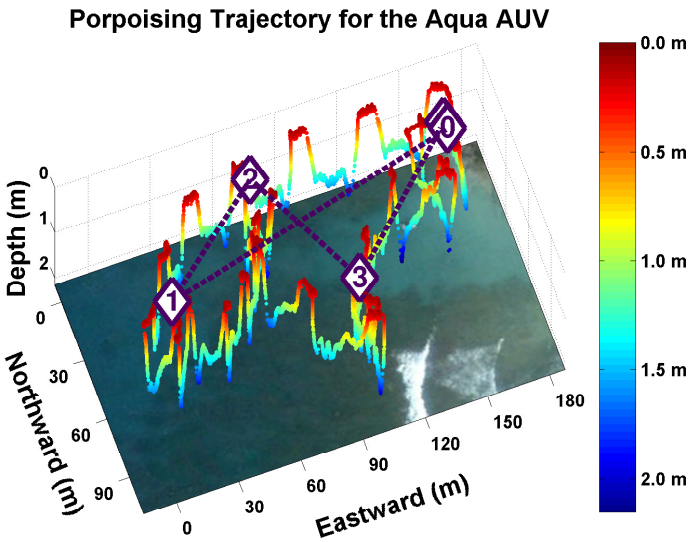
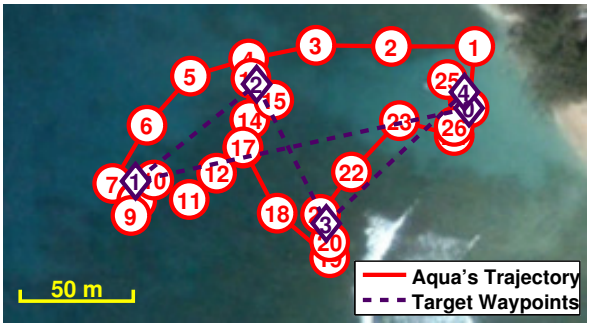
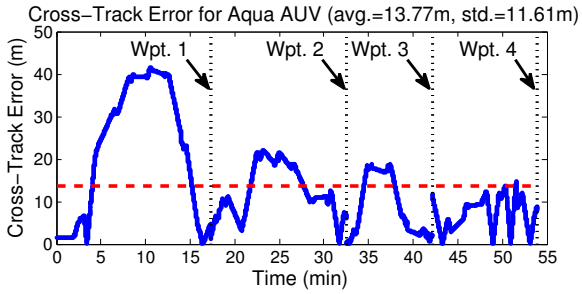


FIGURE 3.6: Porpoising trajectory of the Aqua underwater robot.

The Aqua underwater robot autonomously carried out the remainder of the monitoring task, namely visiting all four target locations selected by the biologist, in about an hour. 20 minutes of that time were devoted to forward motion at about 0.6 m/s. The remaining 40 minutes were devoted to heaving up and down and GPS localization on the surface. As shown in Fig. 3.8 the underwater robot spent on average about 50 s at each porpoising stop to collect GPS readings with small spatial



(a)



(b)

FIGURE 3.7: (a) Overhead view of the Aqua underwater robot's trajectory. Diamond icons depict target locations suggested by the biologist for further inspection. Circular icons denote porpoising ascents during this session. (b) The cross-track error for Aqua's trajectory suggests that it was pushed off-course due to strong currents.

variance, and 40 s moving forward. We found that, although significant, this time interval was required to assure an accurate position estimate could be obtained.

As illustrated by Fig. 3.7(a), the porpoising trajectory of the Aqua robot did not exhibit straight line paths from one waypoint to the next. Fig. 3.7(b) shows that the cross track error of the GPS waypoint follower based on that motion was 13 meters on average. Deviations from the specified path are due to two factors: one is the presence of strong currents and waves, especially when the robot is heaving to the surface, and the other is the bearing correction based on the IMU yaw readings. The effects of these two factors on the resulting trajectory are independent and additive. This can be seen by considering the fact that each forward motion segment lasted the same amount of time, and thus noting that the variances in distances traveled between porpoising stops (red edges in Fig. 3.7(a)) is solely due to strong currents. In contrast, the heading errors and the resulting curvature of the trajectory is due to the drift of the IMU yaw, while the robot is pitching on the surface in order to obtain concentrated GPS readings. Despite these path deviations, the Aqua underwater robot reached each of the target locations at the accuracy of commercial GPS at $<$ 5 m, which was suffient for it to capture underwater footage of the designated coral

patches. The underwater robot returned to the initial location of the boat 90 minutes after the launch of the plane, at which point the experiment ended successfully, with the entire session lasting for two and half hours.

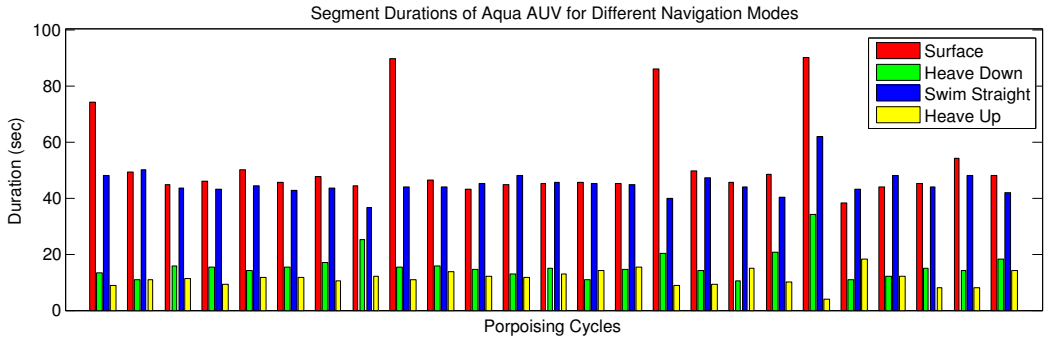


FIGURE 3.8: Durations for each of the porpoising modes during a navigation session for the Aqua underwater robot.

3.2 A Robot Videographer Explores in Tandem with Scientists

We posit that in order for a robot to collect visual data of the environment that will ultimately be useful to the user, surprise-based exploratory behavior is insufficient. We argue that we need a user model to recognize the type of data that matters most to the user. In other words, we need to balance identification of interesting/salient scenes with visual exploration for surprising scenes. Previous work [Girdhar and Dudek, 2014a] has addressed curiosity- and surprise-based visual scene exploration. We focus on the former problem of identifying parts of the scene that will be salient according to a user-specified model.

3.2.1 Modeling Visual Rewards

We model the user's pixel-level visual reward map as an encoder-decoder model, shown in Fig. 3.9, where the input is an RGB image and the output is a 1D image with values in [0,5]. 0 denotes no interest and 5 denotes a very interesting part of the scene. While pixel-level estimation of the user's visual reward function can be seen as having unnecessarily high resolution, and superpixel-level modeling can also be as useful for the purposes of visual navigation and search, we have opted for pixel-level rewards in order to have access to a broader set of labeled semantic segmentation datasets from different settings, such as driving environments [Yu et al., 2018] and various household and outdoor environments [Zhou et al., 2017].

The network architecture that we use is SegNet [Badrinarayanan et al., 2015], a convolutional encoder-decoder architecture, shown in Fig. 3.9. The loss function we use is the categorical cross-entropy for C-category classification:

$$L = -\sum_{c=1}^{C} \bar{p}_c \log(p_c) \tag{3.1}$$

where $\bar{p}_c$ denotes the true probability that the given pixel is of class c, and p_c is the estimated probability output by the network.

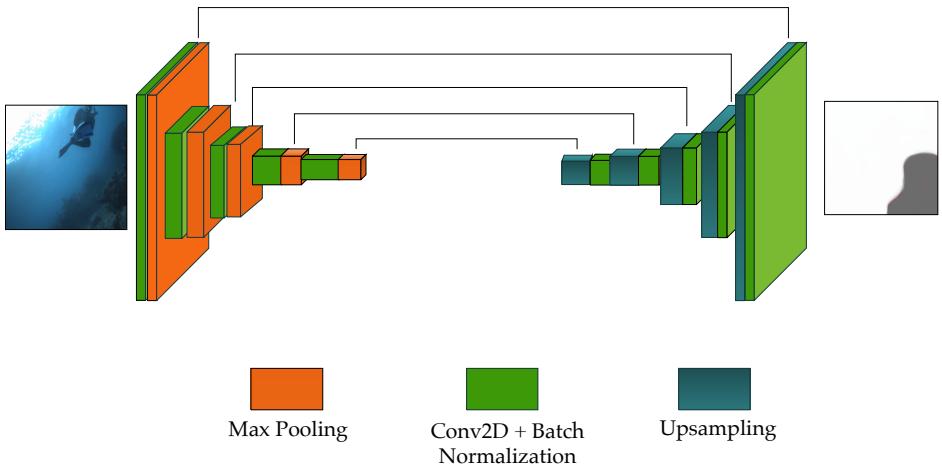


FIGURE 3.9: SegNet encoder-decoder architecture. The indeces of the max pooling operations are copied into the decoder's upsampling operations, thereby reducing memory and avoiding interpolation. Rectified Linear Units are the activation functions used in all layers. The output of the network is the user's visual reward map, here shown preferring coral compared to sand and the diver.

3.2.2 Field Trials

We present a system that demonstrates the concept of a robot videographer in an underwater coral monitoring scenario, where a robot follows a human diver and at the same time collects useful data for later analysis. The physical setup is shown in Fig. 3.10. The forward camera is used for diver tracking and visual servoing, so that the robot does not have to plan in the long-term or navigate, while the back camera is used for videography of interesting visual content as the robot rolls, following the diver. The user's visual reward model has been trained using supervised learning on underwater datasets, totalling about 10K pixel-level labeled images. The details of the training and evaluation procedure are presented in Section 4.4.

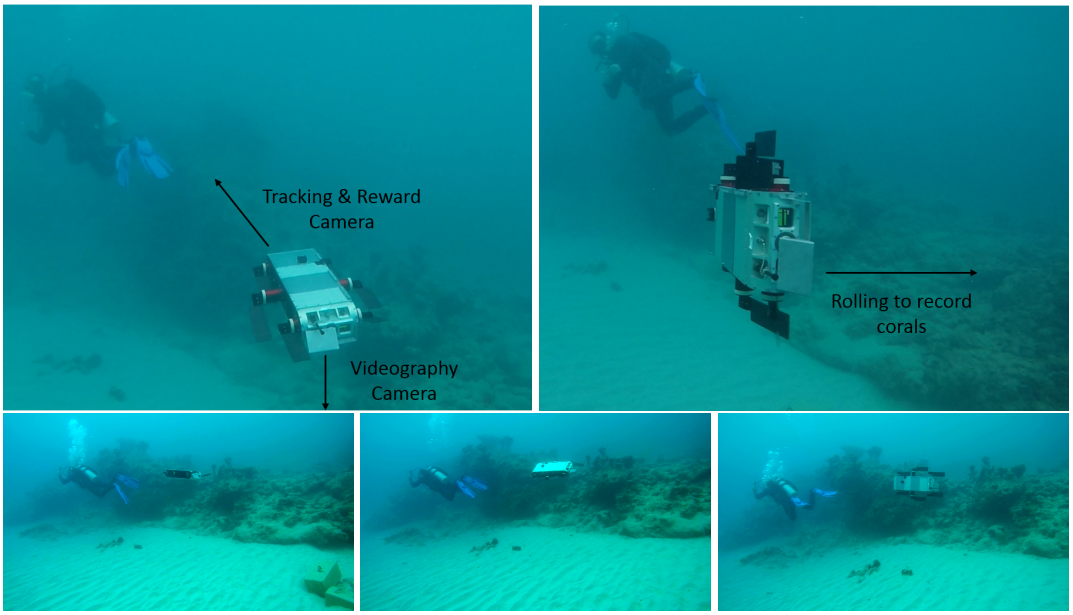


FIGURE 3.10: The Aqua robot following a scientist who inspects coral. The robot has been given an a priori model of the scientist's reward model based on input from the front camera. The robot rolls in order to direct its downward-looking videography camera towards parts of the scene that are of interest to the scientist (in this case, coral).

3.3 Discussion

These two systems are some of the first instances of collaborative human-robot exploration. The heterogeneous multi-robot team is able to collectively cover large areas, however, it has the disadvantage of the robots being too reliant on the scientist to collect data. In fact, the scientist has to be involved throughout the entire experiment because none of the robots know what type of data is deemed valuable. This is addressed by the second system presented above, but at the cost of the robot being again too reliant on the human scientist, this time for navigation. Our goal is to have robots that are capable of navigating and searching for interesting visual content on their own, separately and away from the scientist, and then eventually relocate them and call their attention to any interesting scenes that they might have observed. This would truly open the road to collaborative exploration among a team of robots and a team of humans.

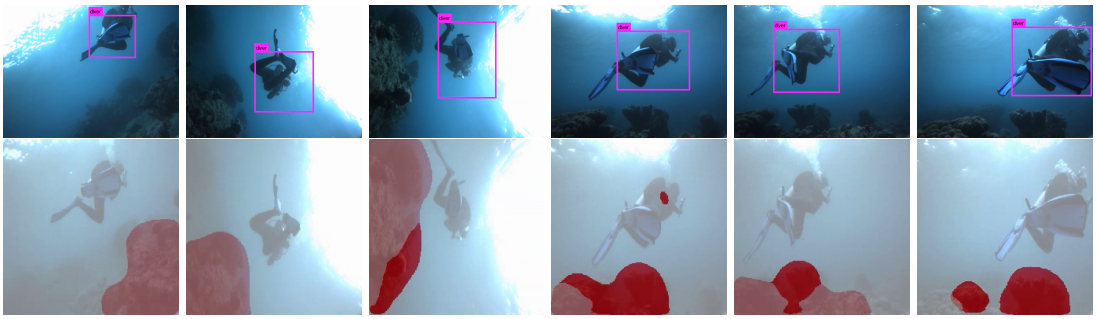


FIGURE 3.11: Images from the front camera of the Aqua robot are used for diver tracking and pixel-level reward estimation, which will determine which part of the scene the robot will roll towards to record. In the bottom images, white denotes uninteresting content, while red denotes very interesting content.

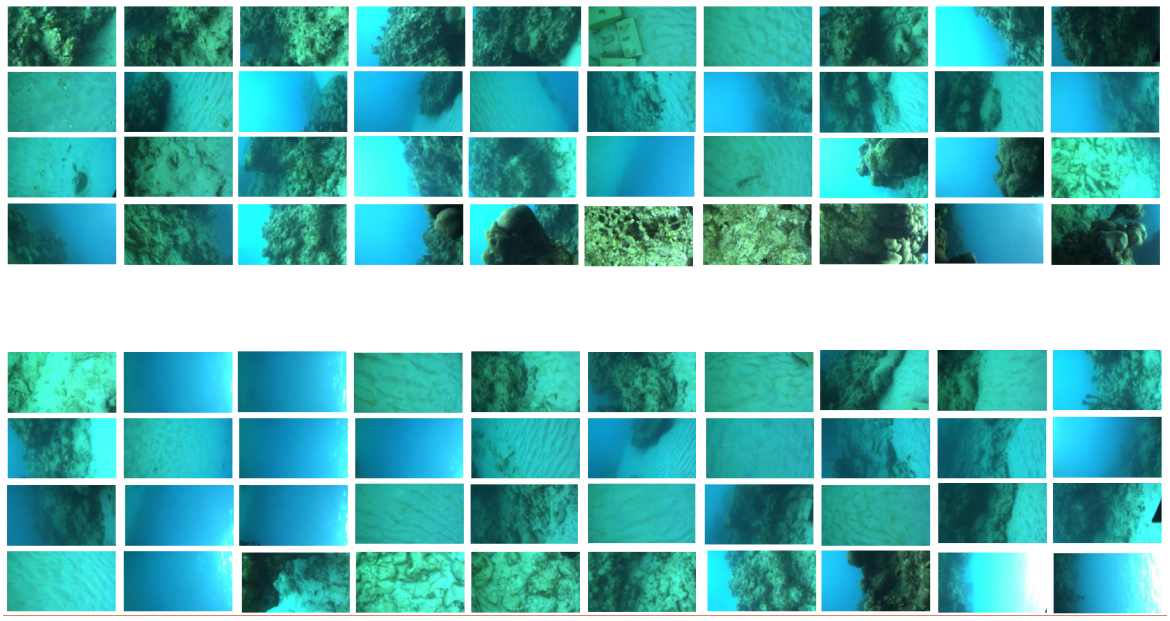


FIGURE 3.12: Top: images recorded using a rolling policy informed by the trained user model recorded sand about 17% of the time. Bottom: periodic rolling policy, agnostic to image content, ended up recording sand almost 50% of the time.

4 ACTIVE LEARNING OF VISUAL REWARD FUNCTIONS

In this chapter we focus on how to make the *specification* process of the user's visual reward model as efficient as possible – we do not want scientists to spend a lot of time providing expert annotations. To this end, in this chapter we evaluate whether active learning methods can provide any advantages in terms of sample-efficiency.

4.1 Model Uncertainty vs Aleatoric Uncertainty

There are two types of uncertainty that arise when modeling a stochastic phenomenon: epistemic (or model) uncertainty, and aleatoric uncertainty. Model uncertainty refers to the type of uncertainty that is reduced to zero asymptotically, as we collect more samples. Aleatoric uncertainty describes inherent stochasticity in the phenomenon that will not reduce with more data.

The typical way of illustrating their difference is by using the example of throwing a fair die. In the limit of throwing it infinitely many times we are going to be sure that the outcome follows a uniform distribution from one to six. Having converged to a single distribution/model was the same as reducing the model uncertainty to zero. That said, trying to predict the outcome of the next throw is still stochastic, and no matter how much data we collect we are not going to be able to reduce the entropy of the uniform distribution. This irreducible uncertainty in observation is aleatoric uncertainty.

4.2 Model Uncertainty via the Bootstrap vs Monte Carlo Dropout

To obtain model uncertainty it is necessary to go beyond directly modeling the distribution $p(y|x, D)$ assuming a single set of weights for the network. Note here that y is the reward map, x is the given image, and D is the training set. We need to perform model averaging, $p(y|x, D) = \int_w p(y|w, x)p(w|D)dw$ in order to explicitly take into consideration the possible weights, w, of the neural network. In order to

draw weight samples from $p(w|D)$ and evaluate $p(y|w,x)$ we use either an ensemble of networks trained using the bootstrap method [Efron and Stein, 1981, Breiman, 1996], or Monte Carlo Dropout [Gal and Ghahramani, 2016a]. We note that these are not the only options for representing uncertainty in neural networks [Blundell et al., 2015, Lakshminarayanan et al., 2017, Zhang et al., 2017, Sun et al., 2019].

4.2.1 The Bootstrap Method

The bootstrap is one of the most significant advances in modern statistical theory. It provides a theoretically grounded way of obtaining variance estimates (and other properties) of arbitrary estimators, by estimating these properties from an approximate distribution. This distribution is typically obtained by resampling with replacement. The main idea is to sample datasets $D_1, ..., D_M$, each of which is constructed from the original i.i.d. dataset D via sampling $|D| = N$ elements with replacement, so that each resulting dataset D_i might have multiple copies of a labeled example and the same size as D. Then, the bootstrap method trains M different copies of the estimator, in our case a neural network as shown in Fig. 3.9, each on a different dataset, D_i, which results in different weights. The aggregate output reported at the end is the empirical average of the M outputs and the variance is estimated by computing the empirical variance of the M network outputs. In our case the output of each network is a pixel-level categorical distribution.

It is worth mentioning that each dataset D_i contains about 63% of the unique labeled examples in D. To see why this is the case, consider a single dataset D_i. The probability of an element from D not being selected in D_i after N samples with replacement is $1 - 1/N$. After N such draws the probability of an element not being selected is:

$$(1 - 1/N)^N \xrightarrow[N\to\infty]{} e^{-1} \approx 0.37 = 1 - 0.63 \tag{4.1}$$

The chief advantage of the bootstrap method has traditionally been its simplicity of implementation. In the case of neural networks, however, implementing the bootstrap method comes with a set of difficulties. First, for networks with a significant GPU memory footprint, such as the SegNet architecture, there are often hardware limits with respect to M, and how many of them can be in memory at any point in time. This makes training slower. Second, similar issues exist with respect to making physical copies of the dataset: minibatch training needs to be modified to ensure sampling with replacement. We do this by increasing the weight of an image by the number of times it has been selected for replacement by the training set sampling procedure.

4.2.2 Monte Carlo Dropout

Dropout is a method for randomly sampling which connections in the network should be independently dropped, in the sense that their output should be multiplied by zero. This effectively removes activation units from the network and produces variability in weights and outputs. When originally introduced in [Hinton et al., 2012, Srivastava et al., 2014] Dropout was used for regularization at training time via model averaging. Recent work [Gal and Ghahramani, 2016b] has shown that by approximating $p(w|D)$ by a distribution over weights with a Bernoulli mask applied to them, we can estimate a predictive distribution that approximates $p(y|x)$, and therefore both the empirical mean and variance of the approximating distribution can be computed via multiple forward passes along the network.

Specifically, [Gal and Ghahramani, 2016b] approximate $p(w|D)$, where $w = \{W_1, W_2, ..., W_L\}$ are the weight matrices of the layers in a multilayer perceptron, by a simpler distribution $q_\theta(w)$, defined as follows:

$$W_i = M_i \cdot \text{diag}([z_{i,j}]_{j=1...K_i}) \tag{4.2}$$

$$z_{i,j} \sim \text{Bernoulli}(p_i) \tag{4.3}$$

Here p_i is the probability of keeping unit i, j, and the variational parameters are $\theta = (M_1, ..., M_L)$. The objective of approximating $p(w|D)$ can be expressed in terms of minimizing KL divergence:

$$\theta^* = \text{argmin}_\theta \, KL(q_\theta(w)||p(w|D)) \tag{4.4}$$

Using this approximatation to the original weight posterior, $p(w|D)$, we have the following approximation of the predictive distribution $p(y|x, D)$:

$$q_{\theta^*}(y|x) = \int_w p(y|w, x) q_{\theta^*}(w) dw \tag{4.5}$$

The end result is that instead of computing predictive mean and variance, we can sample weights multiple times from $q_{\theta^*}(w)$ and for each sample evaluate the network at test-time, without making any significant architectural changes, as shown in Fig. 4.1. This will give us the empirical mean and variance of $q_{\theta^*}(y|x)$, which we can use for active learning.

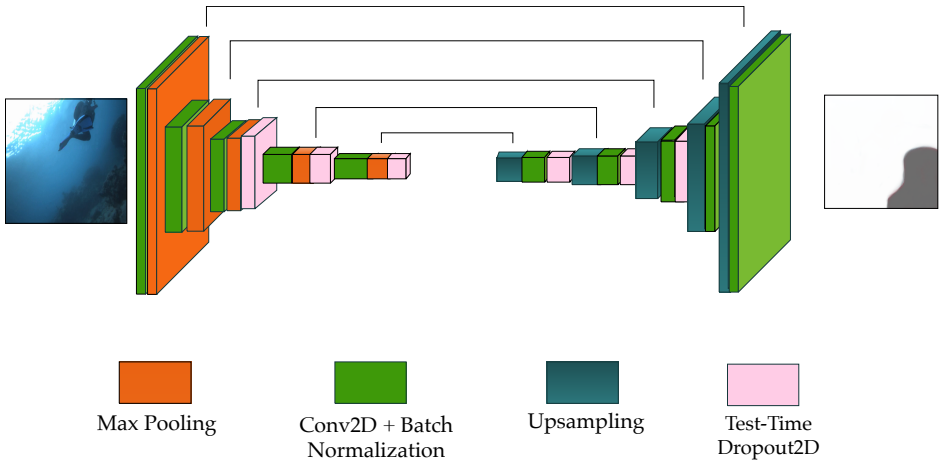


FIGURE 4.1: Bayesian SegNet

4.3 Active Learning of Visual Rewards

Now that we have an estimate of model uncertainty in terms of empirical variance, we evaluate if active learning approaches can help reduce the number of annotations that the user has to provide, by showing the user images to label such that model uncertainty is going to be most reduced. Active learning has seen significant activity in the last two decades [Seung et al., 1992, Balcan et al., 2007, Geifman and El-Yaniv, 2017, Lewis and Gale, 1994, Roy and McCallum, 2001, Settles, 2010]. Recently, active vision-based learning has also been used in the context of deep convolutional neural networks, in the context of categorical image-level classification [Gal et al., 2017], for example on datasets such as MNIST or CIFAR-10. To the best of our knowledge, active learning has not been carefully examined for deep semantic segmentation.

The method we use here for active reward learning, shown in Algs. 1 and 2, is a variant of the *query-by-committee* algorithm [Seung et al., 1992]. This method is characterized by a training algorithm, an acquisition/selection criterion, a pool of unlabeled data, and a set of classifiers/learners whose output on the unlabeled data we can measure efficiently. The label acquisition criterion is a measure of disagreement between the learners, for example model uncertainty as presented previously. In our case, the learners are the neural networks outlined in Fig. 3.9 or 4.1, obtained via a bootstrap ensemble or test-time dropout. The training procedure is supervised learning based on the growing set of labeled data, using the Adam optimizer. In the case of the bootstrap ensemble each model in the ensemble is trained separately, while in the Bayesian encoder decoder, training-time Dropout is deactivated and a single network is trained.

Algorithm 1 Query-By-Committee Active Learning Algorithm (Bootstrap)

$L := \{\}$ // set of labeled data
$U := X$ // set of unlabeled data
$//w_i :=$ weights for learner $p(y|x, w_i) \quad \forall i = 1...M$
$S :=$ batch size for new labels to be queried
while U not empty **do**
    $w_i \leftarrow w_{i,0}$ initialize learner weights
    $L_i :=$ sample with replacement from L
    $w_i^* \leftarrow \text{TRAIN}(w_i,\ L_i,\ U) \quad \forall i = 1...M$
    $u_j \leftarrow \text{UNCERTAINTY}(\{w_i^*\}_{i=1...M},\ x_j) \quad \forall x_j \in U$
    QUERY labels y_j for unlabeled batch $B \subseteq U$ of size $|S|$, with highest u_j
    $U \leftarrow U \setminus B$
    $L \leftarrow L \cup \{(x_j, y_j)\ :\ x_j \in B\}$

Algorithm 2 Query-By-Committee Active Learning Algorithm (Dropout)

$L := \{\}$ // set of labeled data
$U := X$ // set of unlabeled data
$//w_i :=$ weights for learner $p(y|x, w_i) \quad \forall i = 1...M$
$S :=$ batch size for new labels to be queried
while U not empty **do**
    $w \leftarrow w_0$ initialize network weights
    $w^* \leftarrow \text{TRAIN}(w,\ L,\ U)$
    Test-time Dropout sample weights $w_i^*, i = 1...M$
    $u_j \leftarrow \text{UNCERTAINTY}(\{w_i^*\}_{i=1...M},\ x_j) \quad \forall x_j \in U$
    QUERY labels y_j for unlabeled batch $B \subseteq U$ of size $|S|$, with highest u_j
    $U \leftarrow U \setminus B$
    $L \leftarrow L \cup \{(x_j, y_j)\ :\ x_j \in B\}$

Uncertainty: We measure label uncertainty of an unlabeled image x in two ways:

1. Variance in the pixel class probabilities among all learner models

$$\sum_{i=1}^{H\times W}\sum_{c=1}^{C}\mathbb{E}_w\left[(p(y_i=c|x,w)-\mathbb{E}_w[p(y_i=c|x,w)])^2\right] \tag{4.6}$$

   where i denotes pixels and c denotes possible classes for each pixel. The variance is taken with respect to the possible models.

2. Disagreement in the mode of pixel-level predictions between each model and the average model's mode prediction

$$y_{w,i}^{\text{mode}}(x) = \underset{y_i}{\text{argmax}}\ p(y_i|x,w) \tag{4.7}$$

$$y_{avg,i}^{\text{mode}}(x) = \underset{y_i}{\text{argmax}}\ \mathbb{E}_w\left[p(y_i|x,w)\right] \tag{4.8}$$

$$\sum_{i=1}^{H\times W}\mathbb{E}_w\left[y_{w,i}^{\text{mode}}(x)\neq y_{avg,i}^{\text{mode}}(x)\right] \tag{4.9}$$

We note that, unlike the two acquisition functions described above, information gain is the actual quantity of interest for active learning [Gal et al., 2017, Houlsby et al., 2011, Cohn et al., 1996, MacKay, 1992]:

$$\begin{aligned}\mathbb{I}(y\ ;\ w\ |x,L) &= \mathbb{H}(w|L)-\mathbb{E}_{y\sim p(y|x,L)}\,\mathbb{H}(w|L\cup\{(x,y)\}) &(4.10)\\ &= \mathbb{H}(y|x,L)-\mathbb{E}_{w\sim p(w|L)}\,\mathbb{H}(y|x,w) &(4.11)\end{aligned}$$

The reason we do not use it in our case is that Eqn. 4.10 requires sampling segmentations from the average model $p(y|x,L)$. On the other hand, estimating $\mathbb{H}(y|x,w)$ requires segmentation samples from $p(y|x,w)$. In both cases, the number of possible segmentations is $C^{H\times W}$ where C is the number of possible classes for each pixel and $H\times W$ is the resolution of the image. This output space is much larger than C, which is what previous active learning methods have considered.

We also note that computing $\mathbb{H}(y|x,w)$ where y is a segmentation is much easier in the case where $p(y|x,w)$ is modeled as a conditional autoregressive model, such as PixelCNN [van den Oord et al., 2016a] and its variants [van den Oord et al., 2016b], where the distribution over possible segmentations is modeled as

$$p(y|x,w) = p(y_1|x,w)\prod_{i=2}^{H\times W}p(y_i|y_{1\ldots i-1},x,w) \tag{4.12}$$

In this case the entropy is easily computed as a sum

$$\mathbb{H}(y|x,w) = \mathbb{H}(y_1|x,w) + \sum_{i=2}^{H\times W} \mathbb{H}(y_i|y_{1\ldots i-1},x,w) \tag{4.13}$$

as long as the entropy is easy to compute for the distribution $p(y_i|y_{1\ldots i-1},x,w)$, which is the case for the Categorical distribution. Although we experimented extensively with this approach we found that the segmentation output produced by Conditional PixelCNN models of $p(y|x,w)$ did not produce accurate segmentation outputs compared to convolutional encoder-decoder networks. In light of this we did not further examine the thread of approximating the information gain using autoregressive models, and instead we mainly relied on the variance and disagreement measures presented above.

4.4 Evaluation

We evaluate the bootstrap- and dropout-based active learning algorithms in Algs. 1 and 2. We compare the pixel-level semantic segmentation accuracy of the two acquisition functions in Eqns. 4.6 and 4.9 on the following datasets:

- A subset of the MIT Scene Parsing dataset, ADE20K [Zhou et al., 2016], containing 6K randomly selected images and their annotations.
- The Berkeley Deep Drive dataset [Yu et al., 2018], containing 7K labeled images for semantic segmentation training and 1K for testing.

Representative images from the two datasets are shown in Fig. 4.2. We compare our two acquisition functions against the baseline of randomly selecting the next batch of images to label. We run each experiment setting with 3 random seeds, unless otherwise noted. There are various sources of stochasticity in the algorithms presented above. For example, training is done using the Adam optimizer [Kingma and Ba, 2014], a variant of stochastic gradient descent, where the order of seeing labels varies according to the seed. For dropout the seed affects sample weights and, thus, the uncertainty estimates. For bootstrap estimates the seed affects which subset of the original labeled dataset each member of the ensemble will be trained on. We note that each training session was run for 200 epochs with the default parameters of the optimizer, and batch size of 8.

Dropout: Our findings show that on average the disagreement measure outperforms random selection by 2%, when $M = 5$ models are sampled from the weight posterior. On the MIT Scene Parsing dataset, shown in Fig. 4.3, the performance

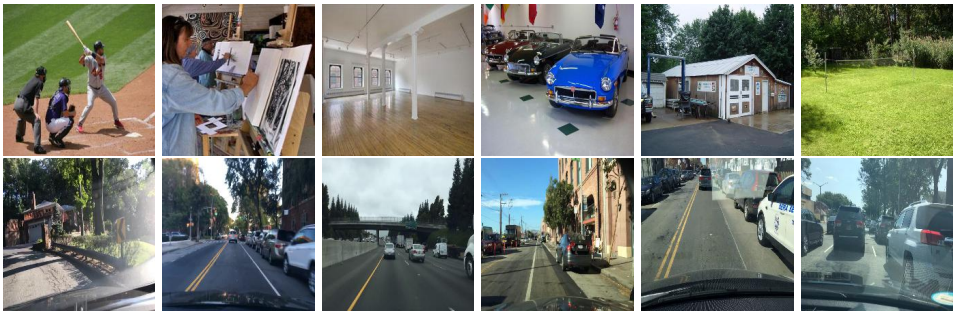


FIGURE 4.2: Top: sample images from the MIT Scene Parsing dataset. Bottom: sample images from the Berkeley Deep Drive dataset.

of selection by variance has high variability and on average performs worse than the random baseline. One likely reason might be that the test accuracy of the system is reported based on the most likely segmentation. Other likely segmentations from the output of the average model are essentially thrown out. The disagreement criterion operates directly on the most likely segmentation, while the variance criterion takes into account the entire categorical distribution of each pixel. It is possible that the variance criterion focuses on images where there is disagreement among models for less likely pixel labels.

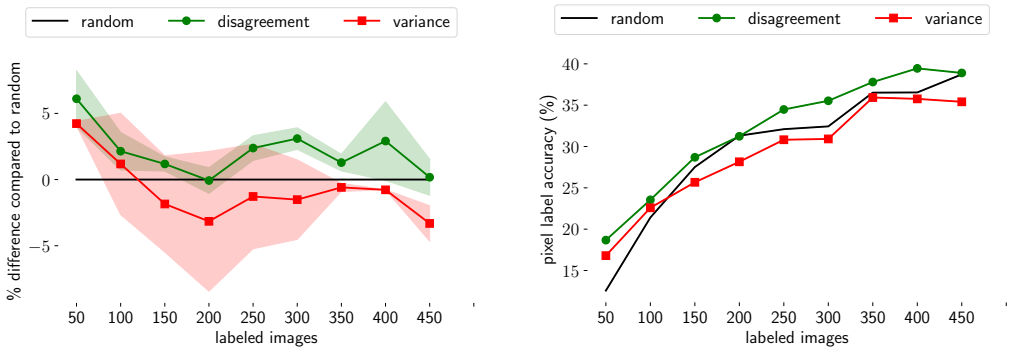


FIGURE 4.3: Dropout-based active learning on the MIT Scene Parsing dataset. The number of possible classes for each pixel is C=150. Left: the difference between selection according to variance and disagreement compared to uniformly random selection. Right: Percentage of correctly labeled pixels. Standard deviation is plotted for 3 random seeds.

On the Berkeley Deep Drive dataset, the disagreement criterion is on average 1% better than random, while the difference between variance and random is insignificant. It is worth mentioning that the performance of active learning approaches is highly dataset dependent. The most illustrative work that shows this is [Mussmann and Liang, 2018], which evaluates the data efficiency of a commonly used active learning algorithm across 21 datasets from OpenML, and finds that there is

inverse correlation between data efficiency and the error of a classifier that has access to the full dataset.

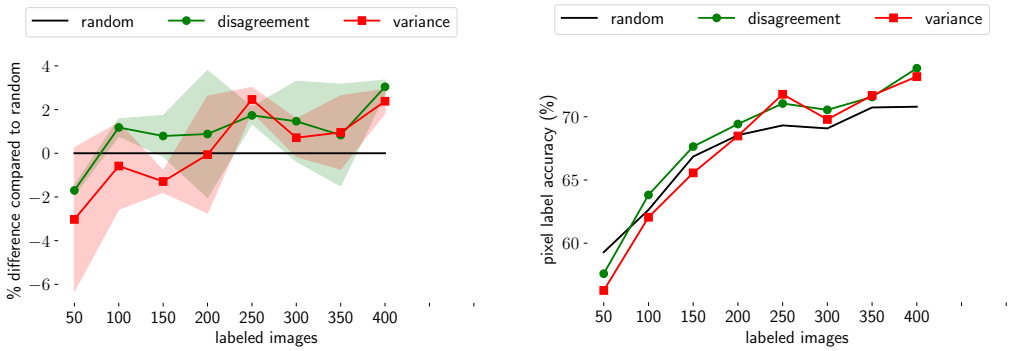


FIGURE 4.4: Dropout-based active learning on the Berkeley Deep Drive dataset. The number of possible classes for each pixel is C=19. Left: the difference between selection according to variance and disagreement compared to uniformly random selection. Right: Percentage of correctly labeled pixels. Standard deviation is plotted for 3 random seeds.

Bootstrap: The comparison here is much more time-consuming than for dropout because we train each model in the ensemble serially. The bootstrap ensemble that we used for these experiments consists of $M = 5$ models. Our findings for the MIT Scene parsing dataset indicate that the disagrement selection strategy performs worse than the random selection, on average by about 0.5%, whereas on the Berkeley Deep Drive dataset it outperforms random by about 0.3% on average.

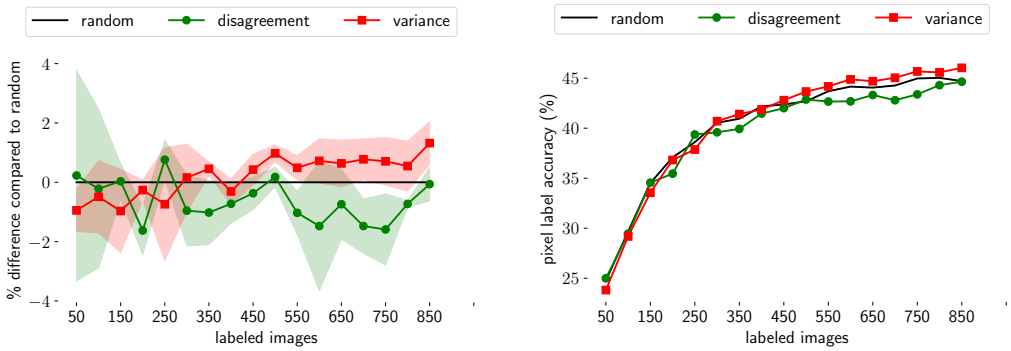


FIGURE 4.5: Bootstrap-based active learning on the MIT Scene Parsing dataset. The number of possible classes for each pixel is C=150. Left: the difference between selection according to variance and disagreement compared to uniformly random selection. Right: Percentage of correctly labeled pixels. Standard deviation is plotted for 3 random seeds.

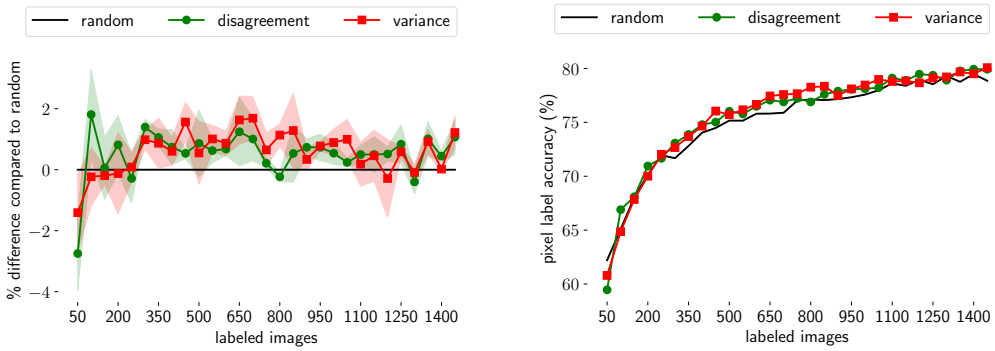


FIGURE 4.6: Bootstrap-based active learning on the Berkeley Deep Drive dataset. The number of possible classes for each pixel is C=19. Left: the difference between selection according to variance and disagreement compared to uniformly random selection. Right: Percentage of correctly labeled pixels. Standard deviation is plotted for 3 random seeds.

4.5 Discussion

The improvements of the query-by-committee active learning method that is used here compared to random selection of images to annotate is very modest. The performance improvement achieved by active learning algorithms in general depends on the distribution of the underlying data. One case in which these methods are expected to produce notable improvements is when the data forms a small number of well-separated clusters and the labels are identical for all data grouped under the same cluster. It is unclear whether this is the case for semantic segmentation on the datasets we tested herein. Evaluations of active learning methods on a large set of datasets (with uncertainty sampling as the active selection criterion) have been conducted in [Mussmann and Liang, 2018]. The main observation is that the data efficiency of active learning is inversely correlated to the error rate of the resulting classifier on the collection of datasets that was examined. It is also reported that previous works on active learning provide a mixed set of conclusions for the gains of active learning of random sampling: from worse than random [Yang and Loog, 2018], to no gain [Schein and Ungar, 2007], to moderate gain [Luo et al., 2013], to double data efficiency [Tong and Koller, 2002]. Further work is needed to characterize the possible gains from active learning from a given data distribution, or methods to transform that distribution so that active learning methods can be more effective than random sampling.

5 MODEL-BASED PURSUIT

In this chapter we propose a model-based predictive pursuit algorithm that enables a robot photographer to maintain visual contact with a subject whose motion is independent of the photographer. We explicitly address the case where visual contact is lost and propose an algorithm that aims to minimize the expected time to recover it. This is novel with respect to previous work, particularly pursuit-evasion games, which do not consider the scenario of intermittent loss of visibility, which is inevitable in practice. In many variants of pursuit-evasion games, once line-of-sight is lost, the game ends. Our work addresses this issue by developing methods that implement the broad algorithmic template shown in Alg. 3:

Algorithm 3 Template For Model-Based Pursuit

```
1: while true do
2:     if subject is in field of view then
3:         do reactive or model-predictive control
4:     else
5:         predict subject's location at t, t + 1, ..., t + H based on history
6:         visit predictions so as to minimize expected time to find the subject
```

The sections that follow address lines 5 and 6 in Alg. 3. In particular, we propose and evaluate two methods for making predictions: First, by making predictions for the subject's behavior based on a navigation reward that is learned through past observations. Second, in the absence of past observations, we make trajectory predictions that are topologically distinct, so that they cover a diverse set of possible behaviors.

5.1 Modeling the Subject's Behavior via Inverse RL

We make use of navigation demonstrations by the target, which we exploit via Maximum Entropy Inverse Reinforcement Learning (IRL) [Ziebart et al., 2008] in order to estimate a reward function that expresses their preferences over the terrain features of the map. For example, the target might prefer to take paved roads, as opposed to grass or sand, in order to navigate to its destination. We can integrate that information in a behavior model for more accurate predictions, thus increasing

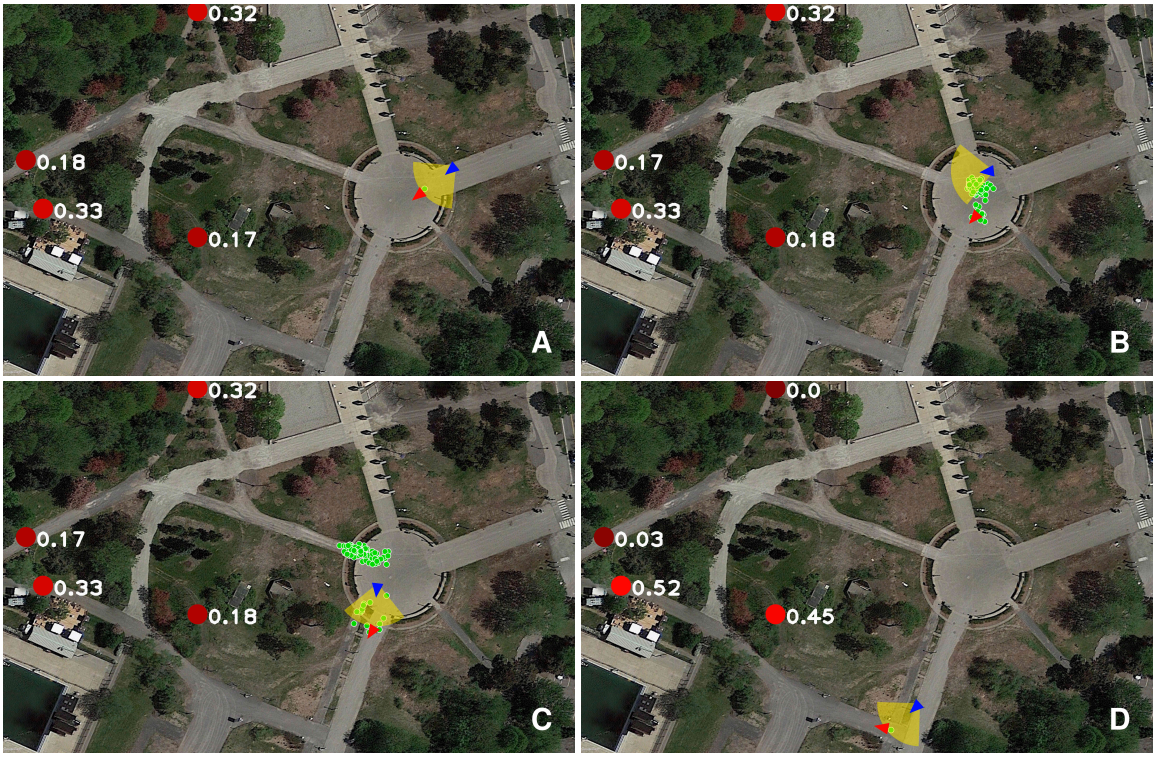


FIGURE 5.1: Red denotes destinations. (A) The follower sees the target. (B) Visual contact is lost. Destinations are predicted and particles (green) start to diffuse. (C) Particles split on two different roads. The follower chooses one group of particles as more promising. (D) The follower re-establishes visual contact. Destinations on the top side of the map become unlikely.

the chance of re-establishing visual contact. Most importantly, feature preferences can be learned in a data-efficient way from as few as ten demonstrations, on a single map, and are typically transferable to other maps that share the same appearance and semantic classes. Using the reward learned from IRL and treating the target as an efficient navigator with respect to that reward, we identify a class of stochastic policies that can very efficiently simulate target behavior without having to perform planning at runtime for the purpose of prediction. This learned predictive and generative model for plausible paths marks an improvement over our prior work on this topic [Shkurti and Dudek, 2017], which did not make use of prior knowledge of the target's navigation behavior. We assume that the robot photographer receives noisy observations of the target within a limited-range and field of view sensor, but has no other means of communication. We also assume that the target navigates purposefully while heading for a destination, and its motion is independent of the actions of the photographer, as opposed to being evasive or cooperative.

Our method operates on both positive and negative observations and maintains

a belief distribution on the target's possible locations. While the target remains unseen, the belief expands spatially and uncertainty grows. Our planning algorithm generates actions that shrink uncertainty or even completely *clear* it. The key enabling insight that allows us to plan over the vast array of potential target paths is the use of the learned navigation reward, which we use to reduce the set of possible hypotheses into a few possibilities that are compatible with the target's demonstrated navigation behavior and the set of available destinations.

5.1.1 Maximum Entropy Inverse Reinforcement Learning

Making long-term and short-term predictions about the future behavior of a purposefully moving target requires that we know the instantaneous reward function that the target is trying to approximately optimize. Given a set of demonstration paths that trace the target's motion on a map, we can infer which parametric transformation of features of the map explains them, using the framework of Maximum Entropy Inverse Reinforcement Learning (IRL) [Ziebart et al., 2008].

We assume the existence of feature vectors $f(s) = [f_1(s), f_2(s), ..., f_N(s)]^\top$ where s is a location on the map. In our case, these N feature maps are boolean images that indicate semantic labels over satellite maps, such as roads, vegetation, trees, automobiles, buildings, as well as dilations of these boolean maps. We also assume that the instantaneous reward is a linear transformation of the features, $r_\theta(s, a) = \theta^\top f(s)$, parameterized by the weight vector θ. As mentioned in the background section, Maximum Entropy IRL factors the probability distribution over trajectories $\tau = (s_0, a_0, s_1, ..., a_{T-1}, s_T)$ as an instance of the the Boltzmann distribution:

$$p(\tau|\theta) = \frac{1}{Z(\theta)} \exp\left(\sum_{t=0}^{T} \theta^\top f(s_t)\right) \tag{5.1}$$

where $Z(\theta)$ is the partition function that acts as a normalization term. As mentioned in the background section, this distribution arises from the constrained optimization problem of maximizing entropy subject to first-moment matching constraints, which specify that the expected feature count at each state should be the same as the empirical mean feature count in the demonstration dataset. It is worth mentioning that Eqn. 5.1 assumes deterministic dynamics for the system being modeled, as well as no noise in the observation of the demonstrated trajectories. Maximizing the log-likelihood function of the trajectories in the demonstration dataset D, is done as

follows:

$$\theta^* = \underset{\theta}{\text{argmax}} \sum_{i=1}^{|D|} \log p(\tau_i|\theta) \tag{5.2}$$

$$= \underset{\theta}{\text{argmax}} \sum_{i=1}^{|D|} \sum_{t=0}^{T_i} \left(\theta^\top f(s_t^{(i)}) - \log Z(\theta)\right) \tag{5.3}$$

This optimization problem is convex for deterministic dynamics, so gradient-based optimization methods are sufficient to solve it. In our case we use the exponentiated gradient ascent method from [Kitani et al., 2012] to obtain an estimate of the optimal reward weight parameters. The gradient of the scaled log likelihood function $\mathcal{L}(\theta) = \frac{1}{|D|}\sum_{i=1}^{|D|}\sum_{t=0}^{T_i}\left(\theta^T f(s_t^{(i)}) - \log Z(\theta)\right)$ is

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{|D|}\sum_{i=1}^{|D|}\sum_{t=0}^{T_i}\left(f(s_t^{(i)}) - \nabla_\theta \log Z(\theta)\right) \tag{5.4}$$

$$= \frac{1}{|D|}\sum_{i=1}^{|D|}\sum_{t=0}^{T_i}\left(f(s_t^{(i)}) - \frac{dZ(\theta)/d\theta}{Z(\theta)}\right) \tag{5.5}$$

$$= \frac{1}{|D|}\sum_{i=1}^{|D|}\sum_{t=0}^{T_i} f(s_t^{(i)}) - \sum_\tau p(\tau|\theta) f_\tau \tag{5.6}$$

$$= \bar{f} - \sum_s p(s|\theta,\pi) f(s) \tag{5.7}$$

where $f_\tau = \sum_{s\in\tau} f(s)$ is the vector of features accumulated by the trajectory τ, $p(s|\theta,\pi)$ is the probability of visiting state s from a policy that solves the MDP with the induced reward function r_θ, and $\bar{f}$ is the empirical average of features in the dataset. The Maximum Entropy IRL algorithm for estimating the optimal reward weights is shown in Alg. 4.

Algorithm 4 MaxEnt IRL

1: D : set of demonstrated paths
2: $\theta \leftarrow \theta_0$
3: **while** not converged **do**
4: Use value iteration to solve MDP(θ) for the optimal policy $\pi(a|s,\theta)$
5: Compute state visitation frequencies $p(s|\theta,\pi)$
using dynamic programming as in [Ziebart et al., 2008]
6: Compute gradient as in Eqn. 5.7
7: $\theta \leftarrow \theta \exp(\eta \nabla_\theta \mathcal{L}(\theta))$ [Kitani et al., 2012]

5.1.2 Terrain-Based Prediction Model For Navigation

Once we learn the parameters of the target's reward function from a set of example paths we can make predictions about the target's actions. We treat the target as an efficient navigator through the environment, approximately optimizing its trajectory while trying to reach its destination. We want to account for the fact that the target's optimization process is approximate. In existing literature [Ziebart et al., 2009, Tamar et al., 2016], one way to perform approximate planning on MDPs, and also in particular in step 4 of Alg. 4, is done through *softmax value iteration*. Instead of the standard update rules in value iteration:

$$Q^*_\theta(s,a) = r_\theta(s,a) + V^*_\theta(s') \tag{5.8}$$
$$V^*_\theta(s) = \max_a Q^*_\theta(s,a) \tag{5.9}$$

the max operation is replaced by a softmax:

$$\widetilde{Q}_\theta(s,a) = r_\theta(s,a) + \widetilde{V}_\theta(s') \tag{5.10}$$
$$\widetilde{V}_\theta(s) = \operatorname*{softmax}_a \widetilde{Q}_\theta(s,a) \tag{5.11}$$

where $(s,a) \to s'$ is a known dynamics model. In order to sample paths on which the target can move towards one of the available destinations, we perform offline computation of one approximate value function per destination d, denoted here by $\widetilde{V}^{(d)}_\theta(s)$. This denotes the approximate value of the optimal path from location s to destination d on the map. We use similar notation for the state-action value function $\widetilde{Q}^{(d)}_\theta(s,a)$.

Given a value function for each destination, we can model the distribution of paths, conditioned on their endpoints, by comparing a path's accumulated value to the value of the optimal path on those endpoints:

$$p(\tau_{A\to B}|\text{start } A, \text{dest } C) \propto \frac{\exp\left(R_\theta(\tau_{A\to B}) + \widetilde{V}^{(C)}_\theta(B)\right)}{\exp\left(\widetilde{V}^{(C)}_\theta(A)\right)} \tag{5.12}$$

This model is useful in order to perform destination prediction, given the history of a path so far:

$$p(\text{dest } C|\text{start } A, \tau_{A\to B}) \propto p(\tau_{A\to B}|\text{start } A, \text{dest } C)p(\text{dest } C) \tag{5.13}$$

where $p(\text{dest } C)$ is the initial prior distribution on the destinations. In our case this is uniform, and the set of possible destinations is already known and of small size.

Offline pre-computation of the value function for each destination d also gives us access to a destination-conditional stochastic policy:

$$\pi_\theta(a|s,d) = \exp(\widetilde{Q}_\theta^{(d)}(s,a) - \widetilde{V}_\theta^{(d)}(s)) \tag{5.14}$$

which amplifies the advantage of action a at the given state. If iteratively applied from any location in the map, this stochastic policy typically leads to an attractor[1] at the given destination d. Examples of this are shown in Fig. 5.2. In effect this gives

FIGURE 5.2: 100 paths sampled by iterative application of the stochastic policy of Eqn. 5.14 at test time. Homotopically-distinct samples are possible, as long as the approximate value function is comparable between the two homotopy classes. In practice, we observed that this is not a typical event.

us a fast and direct way of sampling paths from any location to a specific destination, as opposed to performing MCMC or other sampling methods on Eqn. 5.12. This has a crucial effect on our probabilistic pursuit algorithm, because it enables the generation of possible paths without any planning at runtime for the purposes of prediction, which allows the pursuer to dedicate its resources to formulating a navigation plan that will re-establish visual contact.

5.2 Model-Based Single-Follower Probabilistic Pursuit

We want to enable a single agent to persistently follow a target that is independent of the follower and which moves purposefully in an environment with obstacles.

[1] As long as the softmax value iteration has converged and the feature maps in the given environment are not conflicting.

We assume the follower is equipped with: (a) knowledge of the map and its pose in it (b) demonstrated trajectories that the target has executed in the past in similar environments, revealing its preferences with respect to both potential destinations and routes that lead to them.

We pose the probabilistic pursuit problem as an integrated combination of planning and prediction of the future long-term behavior of the target's state. One way to see this is as a reinforcement learning problem, where the discrete state s_t of the pursuer-target system is defined as $s_t = [x_t, y_t, d_t]$ where x_t and y_t denote the planar poses (row, column, and yaw) of the target and the pursuer respectively, and d denotes the latent destination of the target. The instantaneous reward is the indicator function of whether the follower sees the target or not. In particular, if we let $\text{FOV}(y_t)$ denote the field of view of the follower at time t, then the reward function is $r(s_t, a_t) = \mathbb{1}_{[x_t \in \text{FOV}(y_t)]}$, where a_t is the action that the follower takes at that time.

The follower does not always have full information about the pose of the target, so it needs to maintain a belief $bel(s_t) = p(s_t|h_t)$ about the system's state s_t, given a history vector $h_t = [z_{1:t}, a_{1:t}]$, where $z_{1:t}$ is the history of sensor observations [2]. We assume that the follower has full knowledge of its own state as well as of the map, so the $bel(s_t)$ is really[3] $bel(x_t, d_t)$, which factors into $p(x_t|h_t)p(d_t|h_t)$. Therefore, this formulation allows us to describe the problem as a Partially-Observable Markov Decision Process (POMDP), more specifically as a Belief MDP, for the target's pose and a classification problem for its future destination. We assume that the initial distribution over destinations is uniform.

The POMDP transition model $\mathcal{T}_\theta(s_{t+1}, s_t, a_t) = p(s_{t+1}|s_t, a_t, \theta)$ is stochastic, and depends on the behavior of the target, parameterized here by the vector θ of reward weights, which we learn through inverse reinforcement learning. The transition model factors into $p(s_{t+1}|s_t, a_t, \theta) = p(x_{t+1}|x_t, d_t, \theta)p(y_{t+1}|y_t, a_t)p(d_{t+1}|d_t, x_t)$. We assume for simplicity that the follower's dynamics model is deterministic[4], so $p(y_{t+1}|y_t, a_t) = \mathbb{1}_{[(y_t, a_t) \to y_{t+1}]}$. We also assume that the destination of the target is a discrete latent variable that remains constant throughout the duration of the experiment, so $p(d_{t+1}|d_t, x_t) = \mathbb{1}_{[d_t = d_{t+1}]}$. This formulation is related to the POMDP-lite [Chen et al.,] and the Hidden Parameter MDP [Doshi-Velez and Konidaris, 2016] formulations. The POMDP transition model is therefore $\mathcal{T}_\theta(s_{t+1}, s_t, a_t) = p(x_{t+1}|x_t, d_t, \theta)$ which is the learned stochastic policy in Eqn. 5.14.

[2] $z_t = \emptyset$ indicates not seeing the target. $z_t = x_t$ indicates it was seen

[3] This assumption allows us to decouple the tracking problem from that of Simultaneous Localization and Mapping, although this work is not affected should one need to model localization uncertainty.

[4] Note that the methods presented here do not rely on this assumption.

One of the advantages of using behavior prediction models based on reward parameters learned through IRL, is that we have access to a very fast generative model $(x_{t+1}, y_{t+1}, z_{t+1}, r_{t+1}) \sim \mathcal{G}_\theta(x_t, y_t, d_t, a_t)$ that simulates the system. The more representative this simulator is of the target's actual behavior, the more certain the follower is about the value of each available action.

The POMDP transition model that we use accounts for the possibility of false negative detection errors (not recognizing the target when it is in the field of view of the follower), which is a common case for visual object detectors, particularly at large distances. On the other hand, we assume no false positive detections (recognizing the target when it is in fact not there). The observation model we use has false negative detection probability of q:

$$p(z_t = \emptyset | s_t) = \begin{cases} q & \text{if} \quad x_t \in \text{FOV}(y_t) \\ 0, & \text{otherwise} \end{cases} \tag{5.15}$$

The pursuit problem of finding the target after it has escaped the follower's field of view can be formulated as finding a pursuit policy $\pi(a_{t+1}|h_t)$ that maximizes the follower's value function[5] $V^\pi(h_t) = E_{\pi,\mathcal{T}_\theta}[R_t|h_t]$, where $R_t = \sum_{i=t}^{t+T} \gamma^{i-t} r(s_t, a_t)$ is the discounted cumulative reward function. Solving this problem exactly is in general intractable, so sampling-based POMDP solvers, such as [Silver and Veness, 2010, Shani et al., 2013a] have been introduced in the literature.

Instead of resorting to general-purpose POMDP solvers that need to maintain some level of suboptimal exploratory behavior, we design an algorithm that exploits domain knowledge about the problem and uses a greedy planning behavior to determine which location to navigate to next, in order to maximize the chance of re-establishing visual contact with the moving target. We demonstrate in the results section that this approach outperforms state-of-the art POMDP solvers and provides a better solution to the problem. In the sections that follow we present an analysis of the different components of this algorithm, which is described in detail in Alg. 5.

5.2.1 Particle Filter and Bayesian Updates of the Belief

We represent the belief $bel(x_t)$ about the target's location as a set of particles $x^{(i)}$, whose weights $w^{(i)}$ are updated by a particle filter. When a particle is created we associate it with a fixed speed, a fixed destination $d^{(i)}$ and a single path $\gamma^{(i)}$ to that destination. It travels along this path without deviation. The destination of the

[5] Note that the follower's value function and policy are different than the target's value function and policy.

particle is sampled according to Eqn. 5.13. The sequence of states on the path are sampled iteratively through actions from the learned policy in Eqn. 5.14.

We model the motion of the particle as always traveling at that speed, and we rely on including enough particles in the belief to represent a wide range of speed combinations, without modeling variable speed on any single particle. This is a crucial element because it allows us to deterministically predict where each particle is going to be at a specific time.

The transition model: The design choice of fixed speed and fixed path makes the problem of predicting where the particle is going to be in a few time steps purely deterministic. In our case the only stochasticity for a particle's transition model arises from the initial choice of destination $d^{(i)}$, the reference path $\gamma^{(i)}$ to it, and the fixed speed $v^{(i)}$ at which the particle traverses the path. After those stochastic samples have been drawn the motion of the particle is deterministic. So, the particle dynamics model, $p(x_{t+1}^{(i)}|x_t^{(i)}, d^{(i)}, a_t) = \sum_{\gamma,v} p(x_{t+1}^{(i)}|x_t^{(i)}, d^{(i)}, a_t, \gamma, v)p(\gamma, v|x_t^{(i)}, d^{(i)}, a_t)$, depends entirely on the sampled path and speed because after the independence assumptions it can be simplified to $p(x_{t+1}^{(i)}|x_t^{(i)}, d^{(i)}, a_t, \gamma^{(i)}, v^{(i)})p(\gamma^{(i)}|d^{(i)})p(v^{(i)})$. The conditional transition model $p(x_{t+1}^{(i)}|x_t^{(i)}, d^{(i)}, a_t, \gamma^{(i)}, v^{(i)})$ is completely deterministic [Ng and Jordan, 2000] in our system.

We model the speed distribution for particles using two criteria: first, that we guarantee that the maximum possible target velocity is assigned to many particles, so that their diffusion can catch up with the target's motion, even if it is being evasive; second, that no particles are too slow because they might cause the follower to stay behind and try to clear up hypotheses that move very slowly, while the target progresses to its destination. Thus, we set:

$$p(v) = \begin{cases} \text{Uniform}[v_{\max}/2,\ v_{\max}] & b = 1 \\ \mathbb{1}_{[v=v_{\max}]}, & \text{otherwise} \end{cases} \tag{5.16}$$

where $b \sim \text{Bernoulli}(0.5)$ and $v_{\max}$ is the maximum speed of the target. Approximately half the particles have full speed under this model while the remaining half will be at least half as fast as the target. Again, this encourages progress towards the known destination.

The observation model of the particle filter incorporates a maximum range and field of view limitation to model depth sensors such as the Kinect, but also to reflect the fact that today's target trackers are not generally reliable at large distances. The observation model may also encode particularities of the detector, such as its susceptibility to false negative errors as well as dependence to viewpoint, which is becoming less crucial given recent advances in object detection using supervised

deep learning. The observation model we use in the particle filter is the one presented in Eqn. 5.15.

The choice of the particle filter as the main Bayesian filtering mechanism was made because it is able to incorporate *negative information*, in other words, not being able to currently see the target makes other locations outside the field of view more likely. This is something that most other filters cannot provide.

5.2.2 Pursuer Navigation

Our algorithm assumes for simplicity that navigation is done on a 2D plane. We start by using the Voronoi diagram of the environment to find points that are equidistant from at least two obstacles, and on top of that structure we build the Generalized Voronoi Graph [Choset, 1997]. Its edges include points that are equidistant to exactly two objects, while its nodes are at the remaining points of the Voronoi diagram, either as meetpoints of at least three edges, or as endpoints of an edge into a dead-end in the environment. These two structures provide a roadmap for navigation in the environment on which graph-based path planning takes place.

One of the main advantages of using the structure of the GVG in order to facilitate topological and shortest path queries is that for realistic environments its size is usually small enough to be suitable for real time reasoning. The path obtained from executing a shortest-path algorithm on the GVG is not suitable for fast navigation, in terms of length, curvature and appearance. Shortest paths on the GVG are not globally optimal paths in the rest of the environment. We partially address this issue by using an iterative refinement procedure, where we replace parts of the path that are joinable by a straight line with the points on that line, until little improvement is possible. This is essentially the Douglas-Peucker algorithm, referred to here as REFINE.

We use Yen's K-shortest paths algorithm [Yen, 1971] on the GVG to compute multiple topologically-distinct paths to a given goal, and then we iteratively refine each of the paths to reduce their length. We use this as a heuristic in order to avoid the scenario where the shortest path found on the GVG belongs to a homotopy class that does not contain the globally optimal path to the goal. After executing Yen's algorithm and refining the paths, we return the one with shortest length as an estimate of the shortest path.

Once we have the shortest path, we compute the optimal velocity and linear acceleration controls that will traverse the path in minimum time. We assume that the follower dynamics is omnidirectional $y' = \phi(y, a) = y + a\delta t$, which simplifies significantly optimal control for path execution.

5.2.3 Pursuit Algorithm

Our pursuit algorithm incorporates two modules, depending on whether the target is currently in view or not. If it is, the follower uses reactive feedback PID or model-predictive control to reach a so called "paparazzi" reference frame behind the target (or whichever the desired viewing pose happens to be). This frame of reference gets updated as the target moves in compliance with the surrounding environment, so as to not hit any obstacles. This is illustrated in Fig. 5.3. When visual contact is lost, planning replaces reactive following. We sample destinations and paths for the particles, as explained in the previous section, and we start the sampling and resampling steps in the particle filter according to incoming observations. During a sequence of incoming observations with no detections, the distribution of the destinations $p(d_t|h_t)$ is not updated; it is only updated when the target is detected.

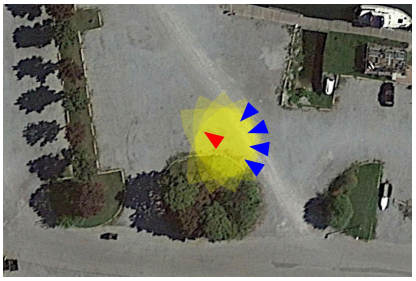


FIGURE 5.3: Paparazzi frames around the current target pose. They describe configurations from which the follower can observe the target. Trees are treated as obstacles. Yellow denotes the field of view. Better seen in color.

The follower plans a path that will lead to a high-value location, as outlined in Alg. 5, and executes that path. When the end of path is reached, if unsuccessful, the follower plans another one. If the target re-enters the field of view, the path hypotheses and the particles are discarded and reactive control resumes its operation.

In the previous section we insisted on being able to deterministically query where a particle is going to be at a particular time in the future. The main reason behind this was to be able to compute the possible times and places at which the follower could visually intercept it. To this end we assume the functionality of a function called $T, \tau = \text{MIN-TIME-TO-VIEW}(\text{GVG}, y_t, p)$, which is a trajectory planner in space and time that enables the follower to navigate from its current state y_t to the paparazzi frame p in minimum time, so as to bring itself in view of a potential target location, as quickly as possible. T is the time it will take to navigate to p, and τ is the trajectory that is planned. We implemented MIN-TIME-TO-VIEW for the case where the follower is an omnidirectional robot, by using iterative refinement of the shortest path obtained from the GVG, as described in the previous section[6].

We restrict the set of candidate locations for visual interception to be points along the reference paths of the particles, for the sake of computational efficiency.

[6]Trajectory optimization and following is required for other types of dynamical systems, but that is outside the scope of this paper.

Our pursuit algorithm computes the minimum times to reach paparazzi frames for a set of waypoints along these reference paths. The possible waypoints include the final destination of the target. So, heading directly to the destination without intermediate stops is one of the considered strategies. The algorithm then selects as the next navigation waypoint the paparazzi frame that sees the particle with the highest ratio of probability over distance , and heads over to reach that paparazzi frame. This is a greedy nearest neighbor algorithm, and we term it NNm for "nearest neighbor pursuit along multiple homotopy classes", or more simply, "topological pursuit."

Algorithm 5 NNm($z_t,\ bel(x_t) = \{(x_t^{(i)}, w^{(i)})\}_{i=1...M},\ y_t$)

z_t : the follower's observation
$bel(x_t)$: the follower's belief about the target's pose
y_t : the follower's pose
B : set of potential destinations
if $z_t = x_t$ i.e. the target is visible **then**
    $bel(x_{t+1}) \leftarrow$ PF-UPDATE(GVG, $bel(x_t), z_t, y_t, B$)
    $\bar{x}_t \leftarrow$ desired paparazzi frame based on x_t (e.g. behind the target)
    $\delta\theta, \delta r \leftarrow y_t - \bar{x}_t$
    $a_t \leftarrow$ feedback control from $\delta\theta, \delta r$
    $y_{t+1} \leftarrow \phi(y_t, a_t)$
    Update $p(d_t|h_t)$ as in Eqn. 5.13
    NNm($z_{t+1},\ bel(x_{t+1}),\ y_{t+1}$)
else
    for $i = 1...M$ **do**
        Sample destination $d_t^{(i)} \sim p(d_t|h_t)$ as in Eqn. 5.13
        Sample path $\gamma^{(i)} \sim p(\gamma|d_t^{(i)})$ as in Eqn. 5.14
        Sample speed $v^{(i)} \sim p(v)$ as in Eqn. 5.16
        Assign path $\gamma^{(i)}$ and speed $v^{(i)}$ to particle $x_t^{(i)}$
    $T_{ji}, \tau_{ji} \leftarrow$ MIN-TIME-TO-VIEW(GVG, y_t, p_{ji}), $\forall j, i$
        for paparazzi frames p_{ji} sampled along $\gamma^{(i)}$
    $\Delta l_{ji}, \Delta\theta_{ji} \leftarrow$ total length and rotation in the path π_{ji}
    $I \leftarrow (i, j)$ such that particle $x_t^{(i)}$ is predicted to reach
        $p_{ji} \in \gamma^{(i)}$ close to follower's arrival time T_{ji}
    $i^*, j^* \leftarrow \underset{(i,j)\in I}{\text{argmax}}\ \frac{w^{(i)}}{\Delta l_{ji} + \Delta\theta_{ji}}$
    while not reached $p_{j^*i^*}$ **do**
        $bel(x_{t+1}) \leftarrow$ PF-UPDATE(GVG, $bel(x_t), z_t, y_t, B$)
        $a_t \leftarrow$ next action in follower's trajectory $\tau_{j^*i^*}$
        $y_{t+1} \leftarrow \phi(y_t, a_t)$
        $t \leftarrow t + 1$
    Go to step 5

5.3 Evaluation and Results

5.3.1 Setup

We set up a simulation environment in order to benchmark our algorithm against existing MDP and POMDP solvers. This environment includes 20 different aerial images, with top-down view, shown in Fig. 5.7, each of which covers areas where the dominant semantic labels of the terrain are: roads, vegetation, trees, buildings, and vehicles.

Each map was annotated offline by human annotators, who also provided examples of target trajectories to pre-specified destinations, from various starting points on the aerial image. The 280 target trajectories, which were demonstrated across the range of all maps, expressed preference for roads and vegetation, avoiding trees and buildings, which were treated as obstacles.

We compared our algorithm with variations on the three following baseline methods: (i) full-information pursuit, (ii) UCT [Kocsis and Szepesvári, 2006], and (iii) POMCP [Silver and Veness, 2010]:

(i) Full-information pursuit refers to the variant of the problem, where the follower knows *a priori* the trajectory of the target, so it can make use of techniques similar to Model Predictive Control (MPC), which recompute a control sequence at each time step, based on known dynamics, and select the best action at each time step, discarding the rest of the plan. Following similar rationale, at each time step, our full-information pursuit algorithm replans a trajectory from the current state of the follower to the closest valid paparazzi frame for the target, and executes the first action prescribed by that trajectory, similarly to MPC. The performance of the full-information pursuer provides an upper bound on the performance of pursuit methods, in which the state of the target may be latent.

(ii) UCT is a sampling-based MDP solver that grows a Monte Carlo Search Tree, using the Upper Confidence Bound (UCB1) rule to trade off exploration vs exploitation. We apply UCT to the variant of our problem in which the state of the target is revealed to the follower at each time step, but the future states of the follower-target system are stochastic, according to the latent preferences of the target, such as destination, route, type of terrain etc.

(iii) POMCP is one of the state-of-the-art sampling-based POMDP solvers. Like UCT, it also performs Monte Carlo Tree Search. It differs from UCT by the fact that each state node in the tree contains action edges that lead to observation nodes, as opposed to other state nodes. It avoids propagating and updating the particle filter belief at each simulation path through the tree, by sampling a particle from the belief according to its weights in the filter, and simulating its evolution through

the tree. POMCP addresses the same version of the problem that our method does. It is worth noting that both in our method as well as in POMCP we use the same particle filter settings, and the same learned IRL reward weights, so the prediction mechanism for the target behavior is identical among the two methods[7].

We compared our methods against these baselines across approximately 4500 experiment scenarios in total, each of which was repeated under the exact same settings across 5 different episodes, to get an estimate of variance in the outcomes of each scenario. These experiment scenarios included variations on:

- the start and destination
- the follower's / target's maximum speed
- the time limit allowed to UCT and POMCP to generate a single action
- the probability of false negative detections
- the speed profile of the target, as it executes its trajectory

For the performance of UCT and POMCP baselines we report the success rate for planning each action within 5 seconds; this of course implies offline operation for these methods, whereas our topological pursuit method runs in real-time at 4Hz.

5.3.2 Findings

Our experiments demonstrate that our method outperforms both POMCP and UCT when the follower and the target share the same maximum speed. This is better shown in Fig. 5.4, which demonstrates the duration of successful pursuit (i.e. how long the follower sees the target) relative to the full-information pursuit, which achieves 100% success rate, managing to maintain visual contact with the target during the entire duration of the experiment. In Fig. 5.4 the probability of false negative observations is set to 0.5, which means that half the time, when the target is in the field of view of the follower, it is not detected, in addition to other times when visual contact is necessitated by the structure of the environment. Our method scores on average approximately 40%, whereas POMCP is at 15% and UCT is under 5%. The main reason why POMCP and UCT perform poorly in this regime of equal maximum speed for the follower and the target is that any suboptimal pursuit actions performed early on in an episode have critical consequences throughout its duration, in the sense that there is little time to recover visual contact if the target is

[7] Also for both UCT and POMCP the Upper Confidence Bound exploration constant was set to $\sqrt{2}$, and the default leaf expansion policy is the uniform distribution.

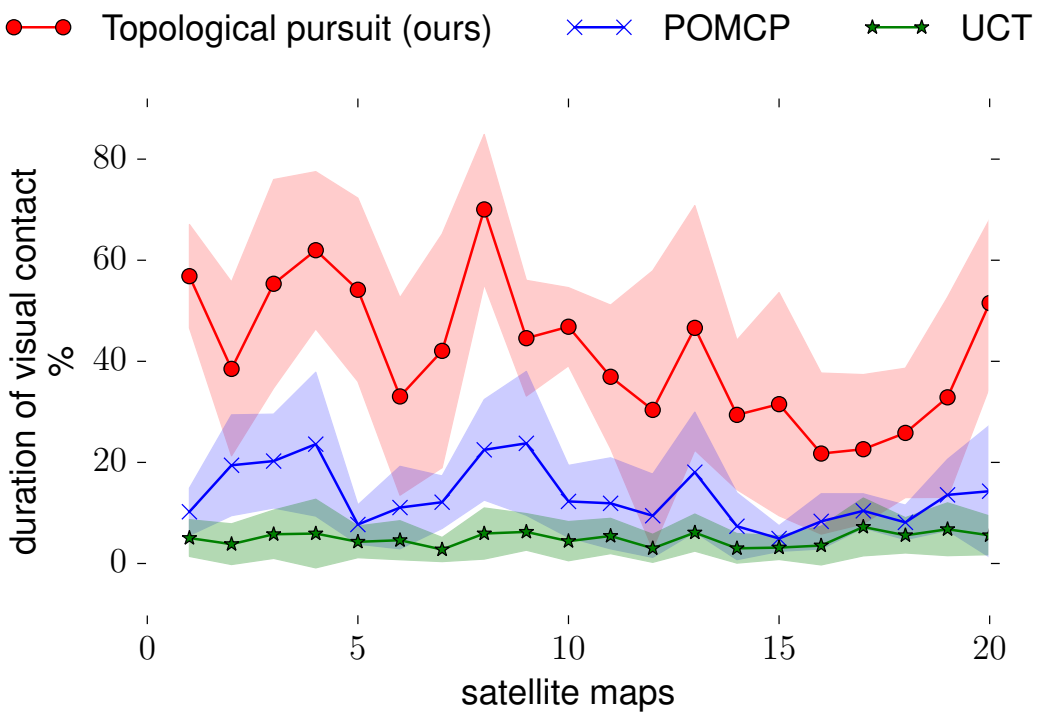


FIGURE 5.4: Relative duration of visual contact across each available test maps. Bars denote 1σ standard deviation. Averages were taken with respect to target trajectories and episodes. The horizontal axis denotes map id.

moving at full speed. POMCP and UCT are prone to committing such suboptimal moves due to their tendency to explore and be optimistic in the face of uncertainty, whereas our algorithm is greedy and focused on navigating to a single destination at a time, so it does not make use of exploratory actions. Additionally, without an explicit penalty term in the design of the reward function, actions produced by UCT and POMCP tend to have high variance as well, which is problematic for deployment in a real vehicle.

As the follower becomes faster relative to the target, suboptimal actions for exploration become less critical because there is time for the pursuer to recover visual contact with the target. This is better illustrated in Fig. 5.5, which shows that the tracking performance of POMCP gradually improves as the follower becomes twice as fast as the target. T his is not surprising as POMCP is an anytime algorithm. Our method, however, does not demonstrate the same property. One hypothesis that might explain this has to do with step 21 of Alg. 5, where we are searching over points along the sampled particle paths, such that the particles are projected to arrive there approximately at the same time as the follower, which means that they are suitable to become rendezvous locations. As the relative speed of the follower increases, the range of locations that are reachable by the target at the same time as

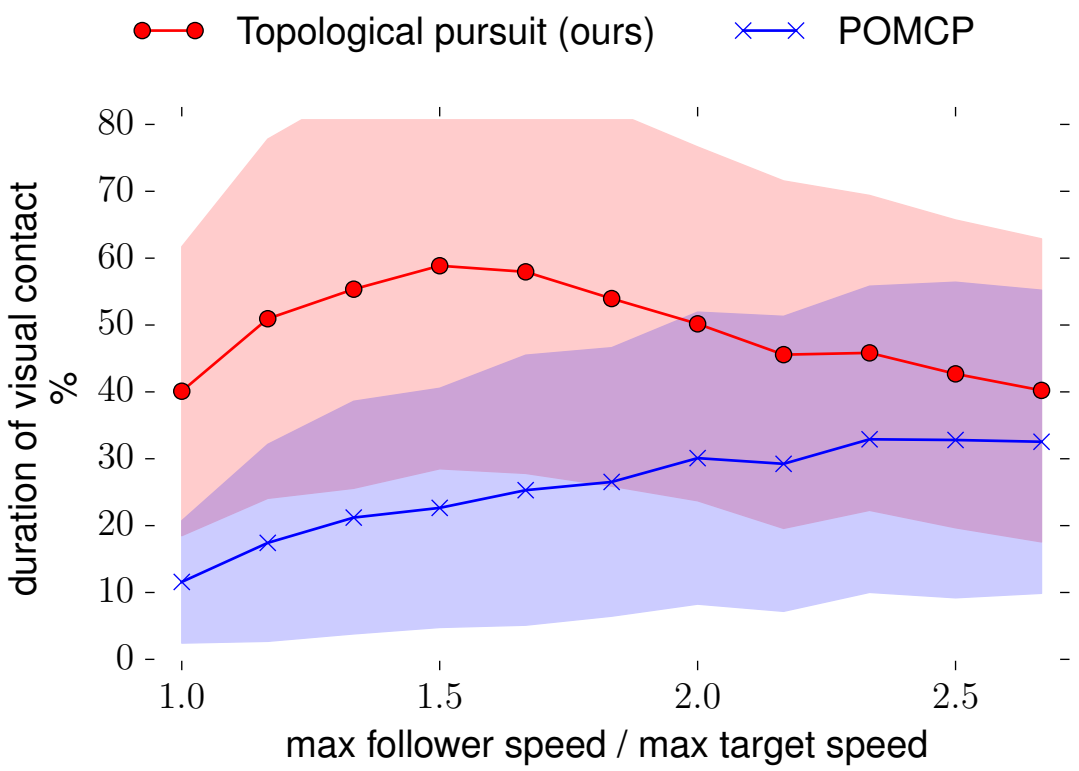


FIGURE 5.5: Relative duration of successful pursuit as a function of maximum speed advantage of the follower. Bars denote 1σ standard deviation. Averages were taken with respect to maps, target trajectories, and episodes.

the follower becomes smaller and smaller, which means that the number of options for the next best paparazzi frame to navigate to is gradually reduced. As the number of good hypotheses for rendezvous locations becomes more limited the quality of pursuit deteriorates.

It is worth mentioning, however, that many practical pursuit scenarios belong in this regime, where the pursuer does not need to be more than twice as fast as the target (aerial photography with vehicles carrying heavy camera equipment, or diving assistant underwater robots). In this neighborhood of relative speed advantage our algorithm still outperforms Monte Carlo Tree Search-based methods.

The third axis of variation that we examined is the false negative detection rate, whose effect is shown in Fig. 5.6. Our method is more robust compared to POMCP, with more than $3\times$ longer tracking on average, when the observation model is prone to false negative errors with probability up to 0.5, which is significant, particularly in vision-based tracking and detection at large distances, where the target occupies a small part of the image.

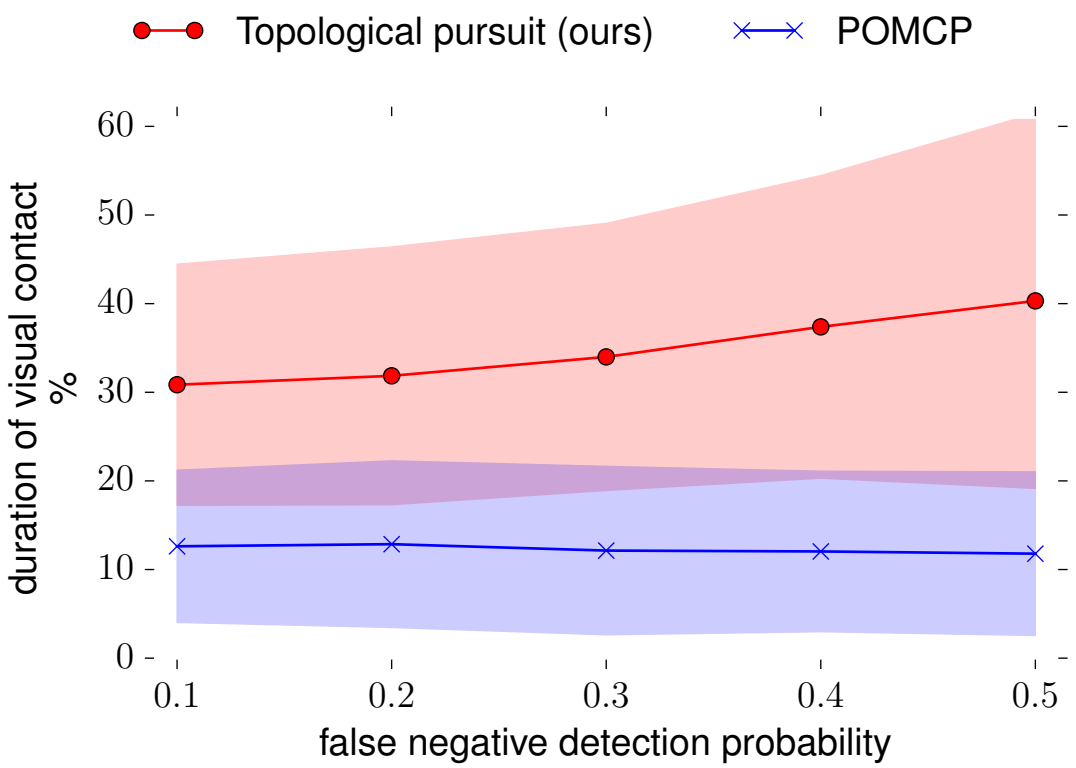


FIGURE 5.6: Relative duration of visual contact as a function of the false negative detection rate. Bars denote 1σ standard deviation. Averages were taken with respect to maps, target trajectories, and episodes.

5.4 Modeling the Subject's Behavior via Topologically Distinct Trajectories

What can we do when no historical data about the subject behavior exist and therefore, the reward cannot be estimated from data? The simple option of modeling the target as a random walk is a bad idea because of lack of any destination and impractically long hitting times for any location (in many cases infinite[8]). On the other extreme, assuming perfect information about which route the target is going to follow and which speed it is going to use, is also not a realistic assumption.

We take the middle ground and we assume extra information about the long-term behavior of the target, namely that we know its final destination, but we do not know the route it is going to take to get there, nor the speed at which it will be travelling. We assume that the target is trying to efficiently reach its destination, even if it does not choose the optimal route. So, it tries to make monotonic progress towards reaching it. This precludes looping behavior, or strange intermediate stops, and it is an assumption inspired by the concept of *navigation functions* [LaValle, 2011] in

[8]For example, the expected time for Brownian motion starting at 0 to hit a horizontal line is infinity.

FIGURE 5.7: Some of the satellite maps and the human-annotated target paths that we used in our test set. Red denotes destinations.

potential fields and control theory. It is also worth mentioning that in this behavior model the target is completely oblivious to the actions of the follower, unlike in pursuit-evasion games in which the target has to come up with best responses to the possible actions of the follower.

Since we are to model the target as an efficient navigator, we need to predict or sample routes along which it will be traveling. Given the target's destination, and lacking any other information, for instance about visual landmarks along the route, or preferred routing information about the target, the only actionable information that the follower has in order to rank some routes more highly than others is the topology and visibility structure of the environment. Intuitively, we want to generate plausible paths that do not deviate from the length of the shortest path by a lot, but manage to explore a rich set of alternatives about how the target might want to go to its destination. In this work we search among these alternatives, and make predictions about the target's short- to long-term behavior, by using the concept of topologically distinct shortest paths.

5.4.1 Related Work

The use of computational topology methods [Hatcher, 2000, Edelsbrunner and Harer, 2010] in robotics has seen a rennaissance, particularly in the domain of planning. Some of the first examples of this line of work were [Bhattacharya et al., 2010a, Efrat et al., 2006], which presented a method to constrain the shortest path found from A^* search to lie on a particular homotopy class, and which was later extended to 3D workspaces in [Bhattacharya et al., 2011a] through the concept of homology. The

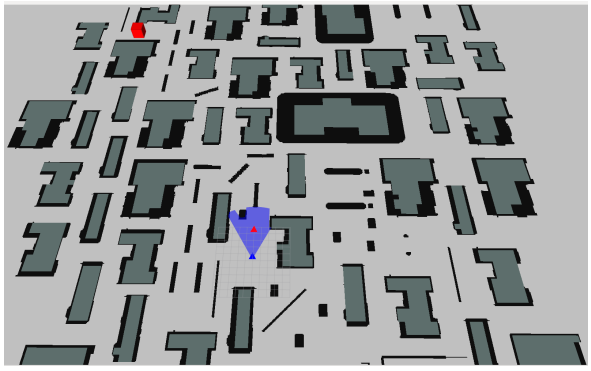


FIGURE 5.8: A view of the initial configuration of the simulator. The blue robot is the follower, with an associated limited field of view. The red robot in the field of view is the target. The red cube denotes the destination of the target. Videos and more info about the project can be found at http://www.cim.mcgill.ca/~mrl/topological_pursuit

difference between the concepts of homotopy and homology is better illustrated in [Narayanan et al., 2013] and [Hatcher, 2000, Edelsbrunner and Harer, 2010]. They are equivalence relations in the space of paths, according to how they traverse the space of obstacles (how they form a sequence of winding numbers around obstacles). We say that two paths are topologically distinct if they belong to different homotopy classes. Two paths that share the same endpoints are *homotopic* if there is a continuous function that deforms one into the other, without hitting any obstacles in the environment. Homotopy is a stricter constraint than homology. In homotopy, the order of winding around obstacles matters, while in homology it does not.

5.4.2 Topologically Distinct Short Paths via the GVG

It is important to clarify at this point that all the upcoming sections make the following two assumptions: (a) the pursuit is being done purely in 2D worlds[9], and (b) the homotopy classes considered contain simple paths, i.e. paths without self-intersections, having a winding number of zero for any obstacle. The concept of enumerating simple homotopy classes as a way of reasoning about the set of paths that they represent is promising because for many 2D environments these classes represent sufficiently many paths to cover the free space. They also allow planning at a level of abstraction that is more robust to changing paths, or small changes in the environment, than other representations such as occupancy grids.

We start by using the Voronoi diagram of the environment to find points that are equidistant from at least two obstacles, and on top of that structure we build the

[9] In 3D, one can reason about persistent topological features [Bhattacharya et al., 2011b, Pokorny et al., 2016b]

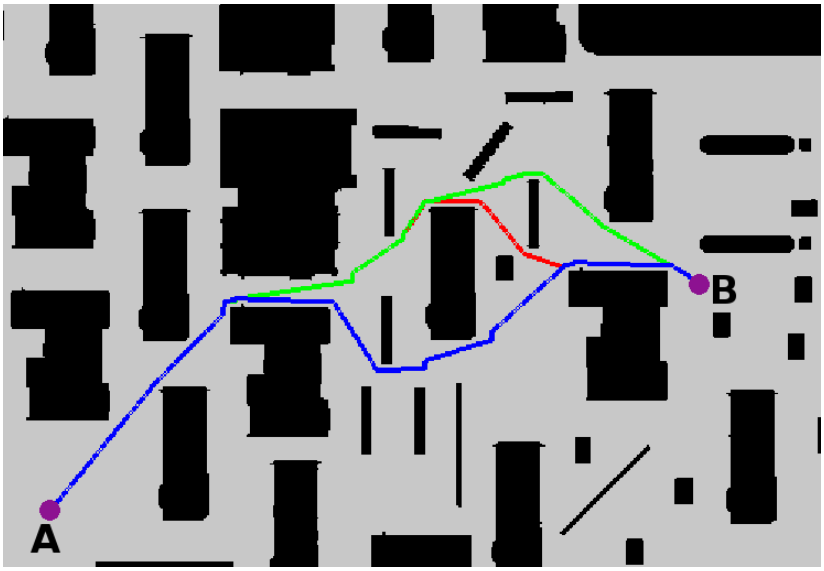


FIGURE 5.9: Three homotopically distinct paths found by applying Yen's 3-shortest paths on the GVG and then refining the paths to reduce their length.

Generalized Voronoi Graph. Its edges include points that are equidistant to exactly two objects, while its nodes are at the remaining points of the Voronoi diagram, either as meetpoints of at least three edges, or as endpoints of an edge into a dead-end in the environment. These two structures provide a roadmap for navigation in the environment on which graph-based path planning and navigation takes place.

One of the main advantages of using the structure of the GVG in order to facilitate topological queries is that its size is usually small enough for realistic environments, which makes it suitable for real-time reasoning. The other advantage is that shortest path queries on the GVG yield simple (no loops) paths which suit the notion of progress to the destination that we mentioned above.

The path obtained from a shortest-path algorithm on the GVG is likely not plausible for prediction in terms of length, curvature and appearance. Shortest paths on the GVG are not globally optimal paths in the rest of the environment. We partially address this issue by using an iterative refinement procedure, where we replace parts of the path that are joinable by a straight line with the points on that line, until little improvement is possible. Thus, we get plausible short paths from the GVG.

We use Yen's K-shortest paths algorithm [Yen, 1971] on the GVG to compute multiple paths to a given destination and then we iteratively refine each of the paths to reduce their length. An example outcome of this process is depicted in Fig. 5.9, which shows three topologically distinct short paths. While this heuristic does not guarantee global optimality with respect to length, it produces plausible alternatives for predictions in real time. For example, computing up to 50 distinct short paths requires less than a second on a modern machine.

5.4.3 Computational Complexity

The computational complexity of Yen's K-shortest path algorithm is $O(K|V|S)$ where $|V|$ is the number of nodes of the GVG graph, and $O(S)$ is the computational complexity of the shortest path algorithm used as a subroutine for Yen's. In our case we use Dijkstra, which has complexity $O(|E| + |V|\log|V|)$. GVG is a planar graph, which implies according to Euler's formula that $|V| - |E| + F = 2$, where F is the number of faces corresponding to the graph (regions enclosed by edges – in our case obstacles – and the outer region). Since for any undirected graph $\sum_{v \in V} \deg(v) = 2|E|$ we have $2F \geq 2|E| - 2|V| = \sum_{v \in V} \deg(v) - 2|V|$. We separate GVG nodes into the set V_1 of endpoints (degree 1) and the set $V - V_1$ of meetpoints (degree ≥ 3). We conclude that $2F \geq |V_1| + 3|V - V_1| - 2|V| = |V| + 2|V - V_1| - 2|V| = |V| - 2|V_1|$, or equivalently $2F + |V_1| \geq |V - V_1|$. This means that the number of meetpoints is at most linear in the number of obstacles and the number of endpoints.

That said, we are counting on the fact that the reduced form of the GVG [Choset and Burdick, 2000] is a roadmap of a smooth version of the environment, and thus an efficient representation of the world, with a bounded number of endpoints per obstacle. In particular, it is more efficient than occupancy grid or pointclouds. For example, the cluttered world shown in Fig. 5.9 contains about 90 obstacles and is represented by about 200 GVG nodes. In practice, the follower could get away with generating fewer alternatives because physically searching all the route hypotheses could require a lot of time, or very high speed. In the next few sections we describe how to use these paths as hypotheses for prediction.

5.4.4 Ranking Topologically Distinct Paths

In order to induce some ranking between the available topologically distinct paths γ we use their total length as their differentiating factor. $l(\gamma)$ denotes the length of that path. We use the softmin operator to turn these lengths into probabilities:

$$p(\gamma) = \exp(-l(\gamma)/\tau) / \sum_{\gamma_i} \exp(-l(\gamma_i)/\tau) \tag{5.17}$$

where τ is the softmin temperature parameter. In the limit, when τ reaches 0 softmin becomes the hard minimum, in other words it assigns full probability to the minimum length path. Higher values of τ encourage more exploration among paths of nearby lengths.

5.5 Evaluation and Results

We validate the performance of our pursuit algorithm, as well as the benefit added from involving topological information, in two ways: first, by comparing our algorithm's performance to pursuit policies of expert humans; second, by comparing our algorithm to two other tracking algorithms, called NNs and NNr, that do not reason about multiple homotopy classes when they predict the motion of the target. Instead, they use the same pursuit algorithm as NNm. NNs stands for "nearest-neighbor pursuit with target prediction along the shortest path to the destination." It predicts feasible and likely interception points along the shortest path to the known destination. NNr stands for "nearest-neighbor pursuit with target prediction along a short path in a randomly-chosen homotopy class." The random choice among available homotopy classes in NNr is done according to the softmin rule in Eq. 5.17, which favors short paths.

5.5.1 Setup

In order to enable both of these types of comparisons we created a simulator which includes two representations of omnidirectional robots, one of which is equipped with a camera sensor, with a limited field of view (about 50 degrees) and a limited viewing range (8 meters). These limits were set based on sensors like the Kinect.

Our simulator indicates that the target is in view with probability 1 if and only if it is indeed, so it models only true positives and true negatives. At each point in time the robot has access to its own pose, as well as the relative pose of the target, provided it is visible. If not, then the follower does not get to see where the target is on the map – it needs to make informed guesses. The only helpful information it has is the eventual destination of the target. We selected worlds that are quite complex, with many sharp turns across small distances, as is shown in Fig. 5.8, in order to make it easier for visibility to be lost, and likely for early mistakes to be penalized highly in terms of performance in the pursuit.

We set up 40 pursuit tasks in total, all with the same starting configuration for the two robots, but with different destinations for the target, one per task. In all initial configurations the follower could see the target. The follower was controllable by joystick for the human users, or by position and velocity control by our planners. We decided to cluster these 40 tasks into eight separate groups, each of which consisted of 5 pursuit tasks at different destinations. Groups 1-4 were used for the comparison between NNm and human pursuit policies, while groups 5-8 were used

for comparing algorithmic policies, namely NNm against NNr and NNs. The maximum speeds of the follower vs the target are shown here:

Group	1	2	3	4	5	6	7	8
Follower Max Speed	2	2.1	3	4	3	3.4	3.8	4.2
Target Max Speed	2	2	3	4	3	3	3	3

We found that allowing the target to be faster than the follower produced low success rates both for humans and our algorithms, particularly in the range of high speeds mentioned above, so we decided against further examining it as a viable scenario. The 20 pursuit tasks in the comparison of humans vs NNm were done in a single map, while the remaining 20 comparisons between algorithms were performed in 5 different maps that presented various degrees of difficulty and challenges for pursuit.

Each of the 40 tasks lasted from 20 seconds up to about a minute. The trajectories of the target were prerecorded offline from human users, and they were played back for each task. These trajectories were chosen so that they took advantage of the full speed of the target. While they do not account for evasive behaviors such as hiding at a fixed place, moving backwards, or doing loops and self-intersections, they can be characterized as efficient, goal-directed trajectories that do not always go through the optimal route. This can easily be seen in Fig. 5.10, where we plot how much the 20 target trajectories used for Groups 1-4 deviated from the optimal paths. The fact that the target trajectories are not following optimal routes can even better be demonstrated through the concept of *homotopy signatures* [Bhattacharya et al., 2010b]. A homotopy signature is a vector of size equal to the number of obstacles in the map. Each entry contains the winding angle of the trajectory with respect to a point on an obstacle in the environment. In our case, since we are not modelling targets that do loops, this signature simplifies to a binary vector. In this case, an entry in the homotopy signature vector is true if the trajectory passes "to the left" of the associated obstacle and false otherwise.

We used these concepts to compare how the signatures of each of the 20 target trajectories compare to the signatures of their respective optimal paths. The results are shown in Fig. 5.11, and they indicate that an intelligent follower will indeed need to make decisions that involve more homotopy classes than that of the shortest path.

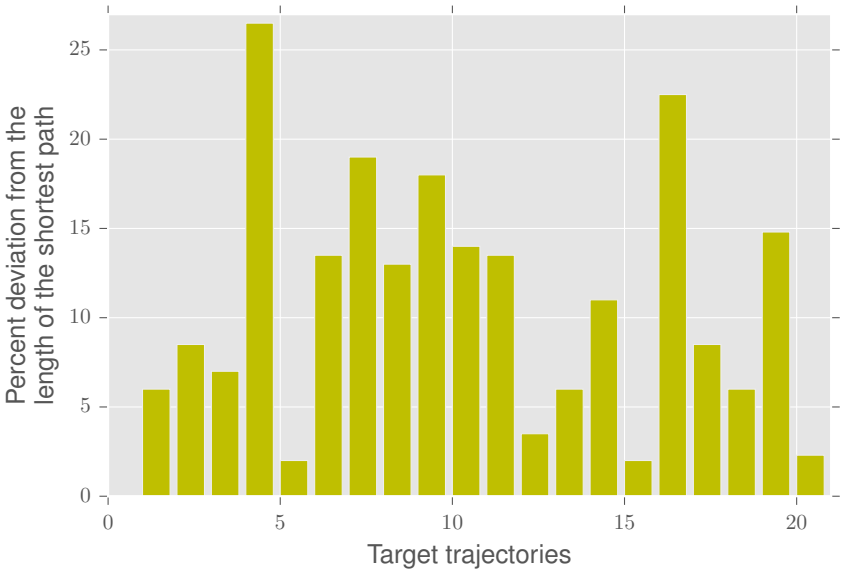


FIGURE 5.10: The percentage of deviation in length of the target's trajectories compared to the optimal path from the initial configuration to the final destination.

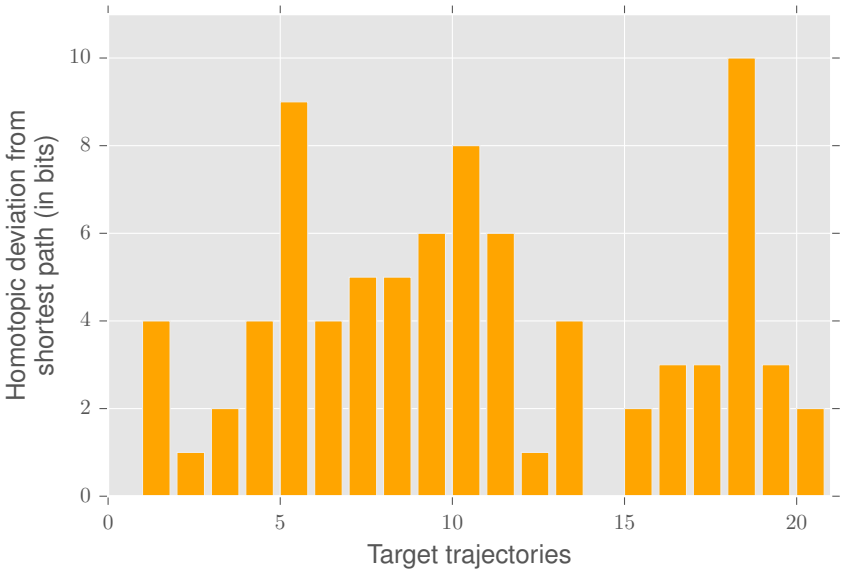


FIGURE 5.11: Bit difference of homotopy signature vectors between each target trajectory that was recorded for Groups 1-4 vs. the shortest paths in each scenario. The median is 4, meaning that in half of the pursuit tasks, if we were to deform the target's trajectory to the shortest path, we would hit 4 out of the approximately 90 obstacles present in the map.

5.5.2 Human Baseline for Probabilistic Pursuit

We recruited 10 people, most of them expert roboticists in their late twenties, all of them with prior experience in using joystick control, and one third of them reporting experience in First-Person Shooter games. We allowed each user to spend as much time as necessary to familiarize themselves with the controls of the simulator and the high speed behavior through practice sessions. Once each task began, the simulator showed them their current score, namely in how many camera frames they had visual contact with the target. It also showed them the time left for each task through a countdown timer. All users reported paying minimal attention to both of these numbers as they were focused on the game of pursuit. The results of this user study are shown in Fig. 5.12.

Humans exhibit the largest standard deviation in their pursuit performance compared to NNm. Our multi-homotopy exploring algorithm performs at least as well as the human baseline in 14/20 pursuit tasks, especially the ones of Groups 3,4, which require fast reaction. Our algorithm does better, in some cases even by 10%. This advantage is dependent on and indicative of the environment's complexity, because any mistake made early on in the pursuit can have a large impact on the user's final score for the task. It is worth mentioning that human performance on the closely related Traveling Salesman Problem is close to optimal for small size problems (of 10 to 20 cities) [Macgregor and Ormerod, 1996]. The planning component of some of these algorithms takes about a second to compute (when visual contact is lost), depending on the number of hypotheses being considered. In that time the target is allowed to move and might have disappeared. To account for this need for planning during the pursuit, we perform the comparisons between algorithms offline. NNm runs in real time if we allow lower resolution in the discretization of time during the planning step, and limit the number of hypotheses generated. The input for both the human participants and the three algorithms was identical: the same target trajectories were replayed to all agents in their respective comparisons.

5.5.3 Benchmarking Algorithmic Performance

In order to precisely quantify what the added benefit of exploring multiple homotopy classes was, we compared our algorithm against two other algorithms, called NNs and NNr, which use the same pursuit logic but differ in the way they predict the future motion of the target. NNs makes predictions in the homotopy class along the shortest path to the known destination, while NNr makes predictions along a randomly sampled homotopy class, according to the softmin distribution

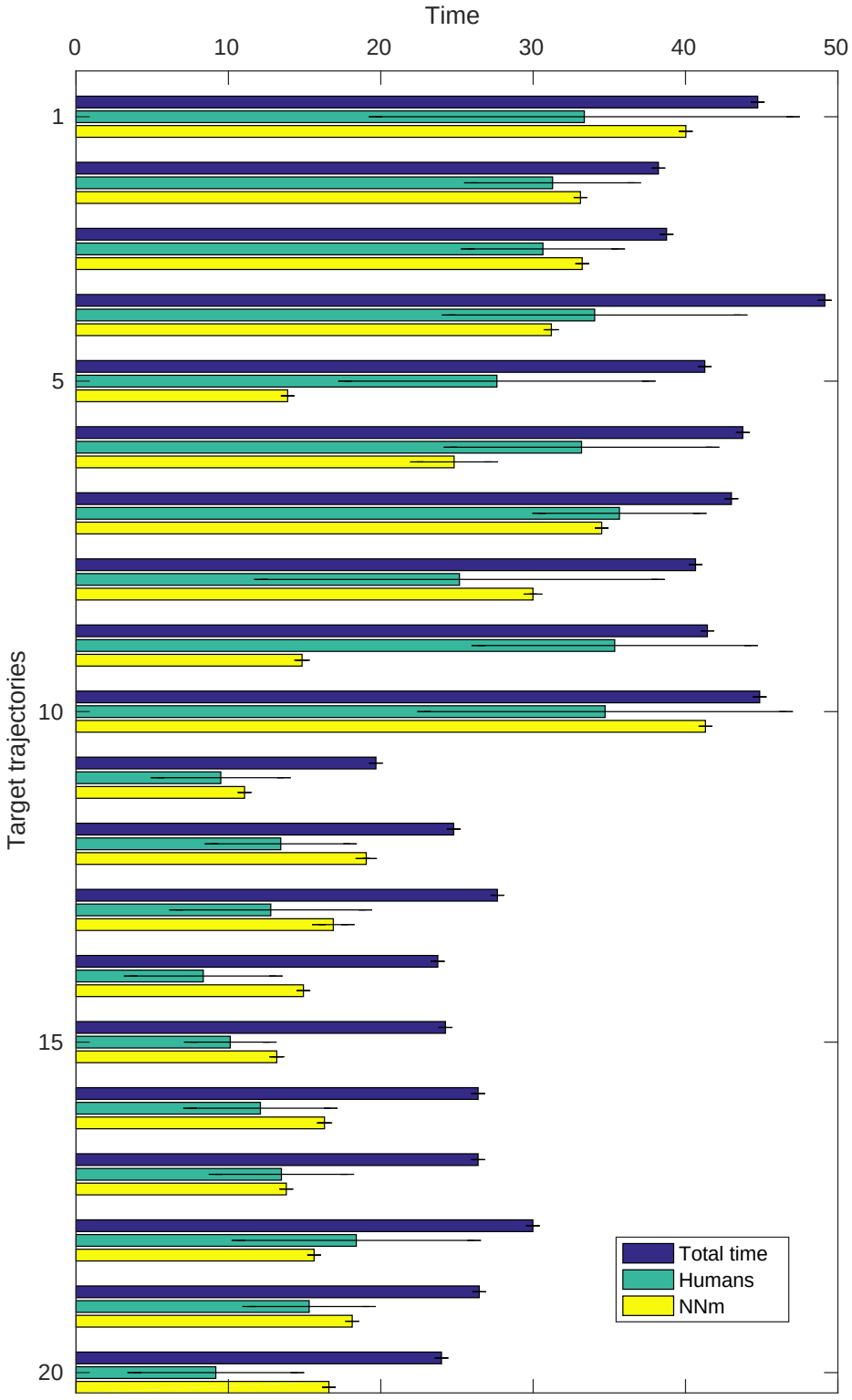


FIGURE 5.12: Comparison between average human performance and the NNm pursuit algorithm (Nearest Neighbor with multi-homotopy prediction). The top bar in each triple is the average total time required for the completion of the trajectory. The middle is the average time humans managed to remain in visual contact with the target robot during that time. The bottom is the average time that NNm maintained visual contact. NNm performed at least as well as humans in 14/20 trajectories. (Seen better in color)

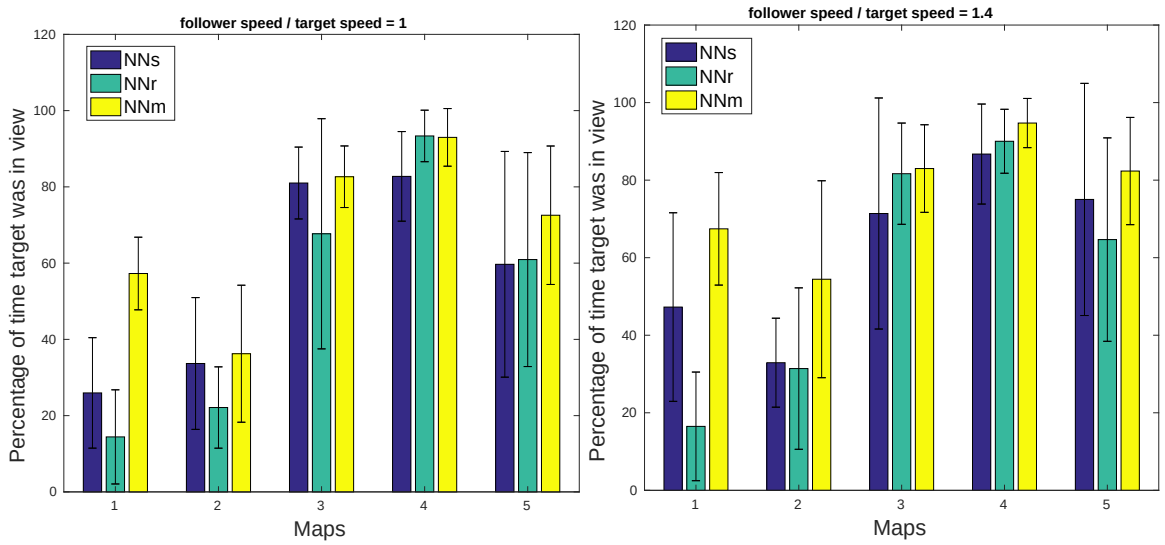


FIGURE 5.13: Average percentage of time that the target was in view during pursuit in different maps. Higher is better. NNm is the third column, and it does at least as well or outperforms the other two methods.

in Eq. 5.17, which favors short paths. We set the temperature for NNm and NNr to $\tau = 2$ for all the experiments shown above.

We compared these algorithms in 5 different maps under varying relative speed settings for the follower and the target, as outlined in Groups 5-8. We found that in challenging maps with small mean free path score, and obstacle layout that required many turns, our algorithm NNm outperformed NNs and NNr. In less challenging maps where the obstacle density is lower we found that the difference among the three algorithms was insignificant. The high density of obstacles in the challenging maps makes reasoning about multiple homotopies necessary because they can be seen as plausible perturbations of the shortest path, which the target might consider as valid routes to its destination. This effect was present across all relative speed variations in Group 5-8. NNm did at least as well as NNs and NNr in all 20 tasks and outperformed them in half.

5.6 Discussion

Although these results are promising in simulation scenarios, there are multiple factors that complicate transfer to or scaling to real world pursuit scenarios. The first obstacle is full knowledge of the 2D or 3D map. In many scenarios, building an accurate 3D map requires expensive sensors, such as LiDAR or multiple well-calibrated cameras. In both cases we get a 3D pointcloud representation of the map, which is not useful for visibility planning – a mesh representation of the

map would be required. Also, the assumption of having prior data to formulate a behavior model of the subject is in many cases quite significant. Finally, the automatic recognition of possible goals that the subject might be headed to is something that we have taken for granted in these methods, but in reality it would be another challenge that would need to be addressed.

6 MODEL-FREE PURSUIT

What happens in the case when we do not have a short term behavior model for the subject of interest, and we do not know which destination it is planning to reach? In this chapter we examine two example scenarios under this category: First, the worst case scenario in which the subject is behaving adversarially and actively trying to escape the pursuer's field of view. Second, a typical convoying scenario in which the pursuer wants to maintain a fixed distance to the subject. For the adversarial case, we present in Appendix A a computational complexity analysis that shows that finding the lowest speed advantage the pursuer needs to have in order to find the subject is NP-hard. For the convoying case we present a vision-based robot convoying system that is demonstrated in underwater scenarios.

6.1 Convoying Pursuit: A Case Study

Vision-based tracking solutions have been applied to robot convoying in a variety of contexts, including terrestrial driving [Schneiderman et al., 1995, Fries and Wuensche, 2014], on-rails maintenance vehicles [Maire, 2007], and unmanned aerial vehicles [Lugo et al., 2013]. Our work demonstrates robust tracking and detection in underwater settings. This is achieved through *tracking-by-detection*, which combines target detection and temporally filtered image-based position estimation. Our solution is built upon several autonomous systems for enabling underwater tasks for a hexapod robot [Sattar et al., 2008a, Sattar and Dudek, 2009b, Sattar and Dudek, 2009a, Girdhar and Dudek, 2014b, Meger et al., 2015], as well as recent advances in real-time deep learning-based object detection frameworks [Redmon and Farhadi, 2016, Ren et al., 2015].

In the underwater realm, convoying tasks face great practical difficulties due to highly varied lighting conditions, and hard-to-model currents on the robot. While previous work in terrestrial and aerial systems used fiducial markers on the targets to aid tracking, we chose a more general tracking-by-detection approach that is trained solely on the natural appearance of the object/robot of interest. While this strategy increases the complexity of the tracking task, it also offers the potential for greater robustness to changing pose variations of the target in which any attached

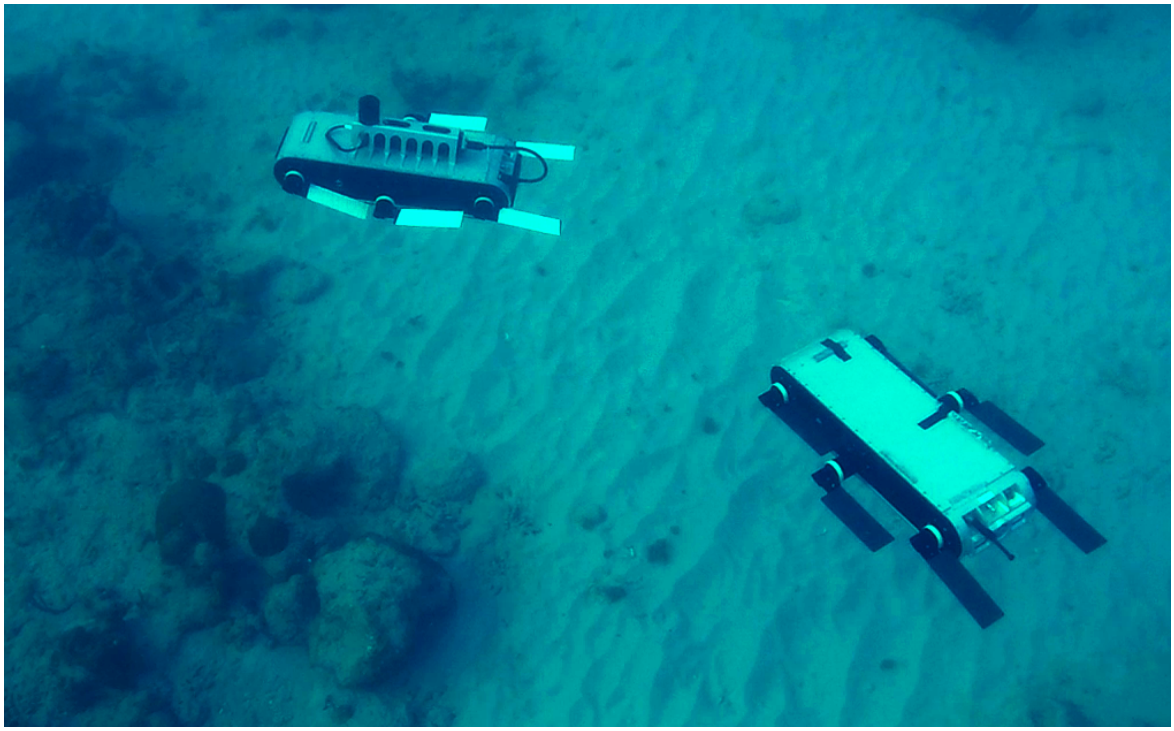


FIGURE 6.1: A sample image from our underwater convoying field trial using Aqua hexapods [Sattar et al., 2008a]. Videos of our field trials, datasets, code, as well as more information about the project are available at http://www.cim.mcgill.ca/~mrl/robot_tracking

markers may not be visible. Other works have demonstrated successful tracking methods using auxiliary devices for underwater localization, including mobile beacons [Chandrasekhar et al., 2006], aerial drones [Erol et al., 2007], or acoustic sampling [Corke et al., 2007]. While these alternative strategies can potentially be deployed for multi-robot convoy tasks, they require additional costly hardware.

Our system learns visual features of the desired target from multiple views, through an annotated dataset of underwater video of the Aqua family of hexapod amphibious robots [Sattar et al., 2008a]. This dataset is collected from both on-board cameras of a trailing robot as well as from diver-collected footage. Inspired by recent general-purpose object detection solutions, such as [Redmon and Farhadi, 2016], [Ren et al., 2015], [Iandola et al., 2016], we propose several efficient neural network architectures for this specific robot tracking task, and compare their performance when applied to underwater sceneries.

In particular, we compare methods using convolutional neural networks (CNNs), recurrent methods stacked on top of CNN-based methods, and frequency-based methods which track the gait frequency of the swimming robot. Furthermore, we demonstrate in an open-water field trial that one of our proposed architectures, based on YOLO [Redmon et al., 2016] and scaled down to run on-board the Aqua

family of robots without GPU acceleration, is both efficient and does not sacrifice performance and robustness in the underwater robot-tracking domain despite motion blur, lighting variations and scale changes.

6.1.1 Related Work

Several vision-based approaches have shown promise for convoying in constrained settings. Some methods employ shared feature tracking to estimate all of the agents' positions along the relative trajectory of the convoy, with map-sharing between the agents. Avanzini *et al.* demonstrate this with a SLAM-based approach [Avanzini et al., 2013]. However, these shared-feature methods require communication between the agents which is difficult without specialized equipment in underwater robots. Using both visual feedback combined with explicit behavior cues to facilitate terrestrial robot convoys has also been considered [Dudek et al., 1995]. Tracking was enhanced by both suitable engineered surface markings combined with action sequences that cue upcoming behaviors. Unlike the present work, that work was restricted to simple 2D motion and hand-crafted visual markings and tracking systems.

Other related works in vision-based convoying often employ template-based methods with fiducial markers placed on the leading agent [Schneiderman et al., 1995, Fries and Wuensche, 2014]. Such methods match the template to the image to calculate the estimated pose of the leading robot. While these methods could be used in our setting, we wish to avoid hand-crafted features or any external fiducial markers due to the possibility that these markers turn out of view of the tracking agent.

An example of a convoying method using visual features of the leading agent without templates or fiducial markers is [Giesbrecht et al., 2009], which uses color-tracking mixed with SIFT features to detect a leading vehicle in a convoy. While we could attempt to employ such a method in an underwater scenario, color-based methods may not work as well due to the variations in lighting and color provided by underwater optics.

6.1.2 Detection Methods: Feedforward CNNs

VGG

The VGG architecture [Simonyan and Zisserman, 2014] has been shown to generalize well to several visual benchmark datasets in localization and classification tasks, so we use it as a starting point for tracking a single object. In particular we started

from the VGG16 architecture, which consists of 16 layers, the first 13 of which are convolutional or max-pooling layers, while the rest are fully connected layers[1], the output of which is the classification or localization prediction of the network.

In our case, we want to output the vector $z = (x, y, w, h, p)$, where (x, y) are the coordinates of the top left corner and (w, h) is the width and height of the predicted bounding box. We normalize these coordinates to lie in $[0, 1]$. p is interpreted as the probability that the robot is present in the image. The error function that we want to minimize combines both the classification error, expressed as binary cross-entropy, and the regression error for localization, which in our case is the mean absolute error for true positives. More formally, the loss function that we used, shown here for a single data point, is:

$$L_n = \mathbb{1}_{\bar{p}=1} \sum_{i=0}^{3} |z_i - \bar{z}_i| - (\bar{p}\log(p) + (1 - \bar{p})\log(1 - p))$$

where symbols with bars denote ground truth annotations.

We evaluated the following variants of this architecture on our dataset:

- *VGG16a*: the first 13 convolutional layers from VGG16, followed by two FC-128 ReLU, and a FC-5 sigmoid layer. We use batch normalization in this variant. The weights of all convolutional layers are kept fixed from pre-training.

- *VGG16b*: the first 13 convolutional layers from VGG16, followed by two FC-128 Parametric ReLU, and a FC-5 sigmoid layer. We use Euclidean weight regularization for the fully connected layers. The weights of all convolutional layers are kept fixed, except the top one.

- *VGG16c*: the first 13 convolutional layers from VGG16, followed by two FC-228 ReLU, and a FC-5 sigmoid layer. The weights of all convolutional layers are fixed, except the top two.

- *VGG15*: the first 12 convolutional layers from VGG16, followed by two FC-128 ReLU, and a FC-5 sigmoid layer. We use batch normalization, as well as Euclidean weight regularization for the fully connected layers. The weights of all convolutional layers are kept fixed.

- *VGG8*: the first 8 convolutional layers from VGG16, followed by two FC-128 ReLU, and a FC-5 sigmoid layer. We use batch normalization in this variant, too. The weights of all convolutional layers are kept fixed.

[1]Specifically, two fully connected layers of width 4096, followed by one fully connected layer of width 1000, denoted FC-4096 and FC-1000 respectively.

In all of our variants, we pre-train the network on the ImageNet dataset as in [Simonyan and Zisserman, 2014] to drastically reduce training time and scale our dataset images to $(224, 224, 3)$ to match the ImageNet scaling.

YOLO

The YOLO detection system [Redmon et al., 2016] frames detection as a regression problem, using a single network optimized end-to-end to predict bounding box coordinates and object classes along with a confidence estimate. It enables faster predictions than most detection systems that are based on sliding window or region proposal approaches, while maintaining a relatively high level of accuracy.

We started with the TinyYOLOv2 architecture [Redmon and Farhadi, 2016], but we found that inference was on our robot's embedded platform (without GPU acceleration) was not efficient enough for fast, closed-loop, vision-based, onboard control. Inspired by lightweight architectures such as [Iandola et al., 2016], we condensed the TinyYOLOv2 architecture as shown in Table 6.1. This enabled inference on embedded robot platforms at reasonable frame rates (13 fps). Following Ning [2] and [Iandola et al., 2016] we:

- replace some of the 3×3 filters with 1×1 filters, and
- decrease the depth of the input volume to 3×3 filters.

Our ReducedYOLO architecture is described in Table 6.2. This architecture keeps the same input resolution and approximately the same number of layers in the network, yet drastically decreases the number of filters for each layer. Since we started with a network which was designed for detection tasks of up to 9000 classes in the case of TinyYOLOv2 [Redmon and Farhadi, 2016], we hypothesize that the reduced capacity of the network would not significantly hurt the tracking performance for a single object class. This is supported by our experimental results. Additionally, we use structures of two 1×1 filters followed by a single 3×3 filter, similar to Squeeze layers in SqueezeNet [Iandola et al., 2016], to compress the inputs to 3×3 filters. Similarly to VGG [Simonyan and Zisserman, 2014] and the original YOLO architecture [Redmon et al., 2016], we double the number of filters after every pooling step.

As in the original TinyYOLOv2 configuration, both models employ batch normalization and leaky rectified linear unit activation functions on all convolutional layers.

[2] 'YOLO CPU Running Time Reduction: Basic Knowledge and Strategies' at `https://goo.gl/xaUWjL`

Type	Filters	Size/Stride	Output
Input			416×416
Convolutional	16	$3 \times 3/1$	416×416
Maxpool		$2 \times 2/2$	208×208
Convolutional	32	$3 \times 3/1$	208×208
Maxpool		$2 \times 2/2$	104×104
Convolutional	64	$3 \times 3/1$	104×104
Maxpool		$2 \times 2/2$	52×52
Convolutional	128	$3 \times 3/1$	52×52
Maxpool		$2 \times 2/2$	26×26
Convolutional	256	$3 \times 3/1$	26×26
Maxpool		$2 \times 2/2$	13×13
Convolutional	512	$3 \times 3/1$	13×13
Maxpool		$2 \times 2/1$	13×13
Convolutional	1024	$3 \times 3/1$	13×13
Convolutional	1024	$3 \times 3/1$	13×13
Convolutional	30	$1 \times 1/1$	13×13
Detection			

TABLE 6.1: TinyYOLOv2 architecture

Type	Filters	Size/Stride	Output
Input			416×416
Convolutional	16	$7 \times 7/2$	208×208
Maxpool		$4 \times 4/4$	52×52
Convolutional	4	$1 \times 1/1$	52×52
Convolutional	4	$1 \times 1/1$	52×52
Convolutional	8	$3 \times 3/1$	52×52
Maxpool		$2 \times 2/2$	26×26
Convolutional	8	$1 \times 1/1$	26×26
Convolutional	8	$1 \times 1/1$	26×26
Convolutional	16	$3 \times 3/1$	26×26
Maxpool		$2 \times 2/2$	13×13
Convolutional	32	$3 \times 3/1$	13×13
Maxpool		$2 \times 2/2$	6×6
Convolutional	64	$3 \times 3/1$	6×6
Convolutional	30	$1 \times 1/1$	6×6
Detection			

TABLE 6.2: Our ReducedYOLO architecture

6.1.3 Detection Methods: Recurrent CNNs

In vision-based convoying, the system may lose sight of the object momentarily due to occlusion or lighting changes, and thus lose track of its leading agent. In an attempt to address this problem, we use recurrent layers stacked on top of our ReducedYOLO architecture, similarly to [Ning et al., 2016]. In their work, Ning et al. use the last layer of features output by the YOLO network for n frames (concatenated with the YOLO bounding box prediction which has the highest IOU with the ground truth) and feed them to single forward Long-Term Short-Term Memory Network (LSTM).

While Ning et al. assume that objects of interest are always in the image (as they test on the OTB-100 tracking dataset), we instead assume that the object may not be in frame. Thus, we make several architectural modifications to improve on their work and make it suitable for our purposes. First, Ning et al. use a simple mean squared error (MSE) loss between the output bounding box coordinates and the ground truth in addition to a penalty which minimizes the MSE between the feature vector output by the recurrent layers and the feature vector output of the YOLO layers. We find that in a scenario where there can be images with no bounding box predicted (as is the case in our system), this makes for an extremely unstable objective function. Therefore we instead use a modified YOLO objective for our single-bounding box single class case. This results in Recurrent ReducedYOLO (RROLO) having the following objective function, shown here for a single data point:

$$\begin{aligned} I((\sqrt{\bar{x}} - \sqrt{x})^2 + (\sqrt{\bar{y}} - \sqrt{y})^2)\alpha_{coord} \\ + I((\sqrt{\bar{w}} - \sqrt{w})^2 + (\sqrt{\bar{h}} - \sqrt{h})^2)\alpha_{coord} \\ + I(\text{IOU} - p)^2\alpha_{obj} \\ + (1 - I)(\text{IOU} - p)^2\alpha_{no\_obj} \end{aligned} \quad (6.1)$$

where $\alpha_{coord}, \alpha_{obj}, \alpha_{no\_obj}$ are tunable hyper-parameters (left at 5, 1, 0.5 respectively based on the original YOLO objective), $\bar{w}, \bar{h}, w, h$ are the width, height, predicted width and predicted height, respectively, $I \in \{0, 1\}$ indicates whether the object exists in the image according to ground truth, p is the confidence value of the prediction and IOU is the Intersection Over Union of the predicted bounding box with the ground truth.

Furthermore, to select which bounding box prediction of YOLO to use as input to our LSTM (in addition to features), we use the highest confidence bounding box rather than the one which overlaps the most with the ground truth. We find that

the latter case is not a fair comparison or even possible for real-world use and thus eliminate this assumption.

In order to drive the final output to a normalized space (ranging from 0 to 1), we add fully connected layers with sigmoidal activation functions on top of the final LSTM output, similarly to YOLOv2 [Redmon and Farhadi, 2016]. Redmon and Farhadi posit that this helps stabilize the training due to the normalization of the gradients at this layer. We choose three fully connected layers with $|\text{YOLO}_{output}|, 256, 32$ hidden units (respectively) and a final output of size 5. We also apply dropout on the final dense layers at training time with a probability of .6 that the weight is kept.

We also include multi-layer LSTMs to our experimental evaluation as well as bidirectional LSTMs which have been shown to perform better on longer sequences of data [Graves, 2012]. A general diagram of our LSTM architecture can be seen in Figure 6.2. Our recurrent detection implementation, based partially on code pro-

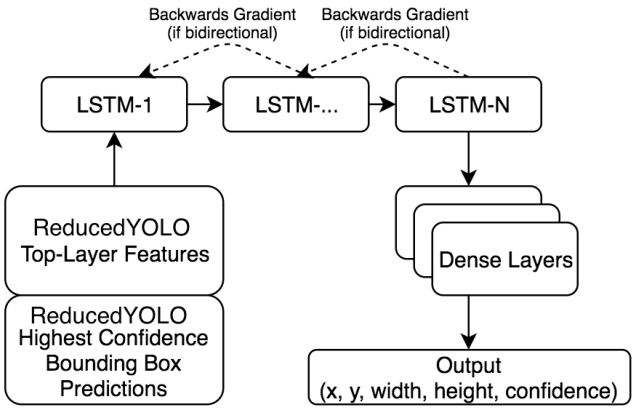


FIGURE 6.2: Overview of our Recurrent ReducedYOLO (RROLO) architecture. The original ROLO work [Ning et al., 2016] did not use bidirectional, dense layers, or multiple LSTM cells in their experiments.

vided by [Ning et al., 2016], is made publicly accessible.[3]

6.1.4 Detection Methods: Based on Frequency-Domain Analysis

Periodic motions have distinct frequency-domain signatures that can be used as reliable and robust features for visual detection and tracking. Such features have been used effectively [Sattar and Dudek, 2009c, Islam and Sattar, 2017] by underwater robots to track scuba divers. Flippers of a human diver typically oscillate at frequencies between 1 and 2 Hz, which produces periodic intensity variations in

[3] http://www.cim.mcgill.ca/~mrl/robot_tracking

the image-space over time. These variations correspond to distinct signatures in the frequency-domain (high-amplitude spectra at 1-2Hz), which can be used for reliable detection. While for convoying purposes, the lead robot's flippers may not have such smoothly periodic oscillations, the frequency of the flippers is a configurable parameter which would be known beforehand.

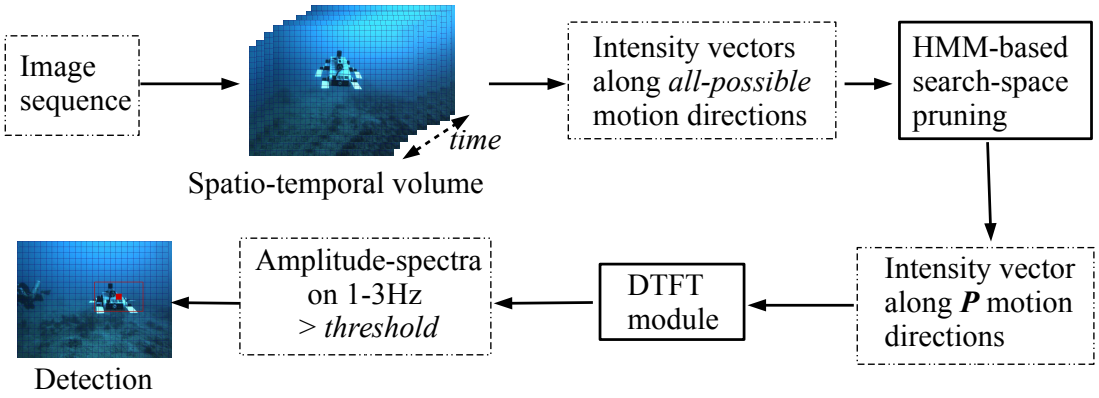


FIGURE 6.3: Outline of mixed-domain periodic motion (MDPM) tracker [Islam and Sattar, 2017]

We implement the mixed-domain motion (MDPM) tracker described by Islam et al [Islam and Sattar, 2017]. An improved version of Sattar et al. [Sattar and Dudek, 2009c], the MDPM works as follows (illustrated in Figure 6.3):

- First, intensity values are captured along arbitrary motion directions; motion directions are modeled as sequences of non-overlapping image sub-windows over time.
- By exploiting the captured intensity values, a Hidden Markov Model (HMM)-based pruning method discards motion directions that are unlikely to be directions where the robot is swimming.
- A Discrete Time Fourier Transform (DTFT) converts the intensity values along P most potential motion directions to frequency-domain amplitude values. High amplitude spectra on 1-3Hz is an indicator of robot motion, which is subsequently used to locate the robot in the image space.

6.1.5 Visual Servoing Controller

The Aqua family of underwater robots allows 5 degrees-of-freedom[4] control, which enables agile and fast motion in 3D. This characteristic makes vehicles of this family

[4]Yaw, pitch, and roll rate, as well as forward and vertical speed

ideal for use in tracking applications that involve following other robots as well as divers [Sattar and Dudek, 2009c]. One desired attribute of controllers in this type of setting is that the robot moves smoothly enough to avoid motion blur, which would degrade the quality of visual feedback. To this end we have opted for an image-based visual servoing controller that avoids explicitly estimating the 3D position of the target robot in the follower's camera coordinates, as this estimate typically suffers from high variance along the optical axis. This is of particular relevance in the underwater domain because performing camera calibration underwater is a time-consuming and error-prone operation. Conversely, our tracking-by-detection method and visual servoing controller do not require camera calibration. Our con-

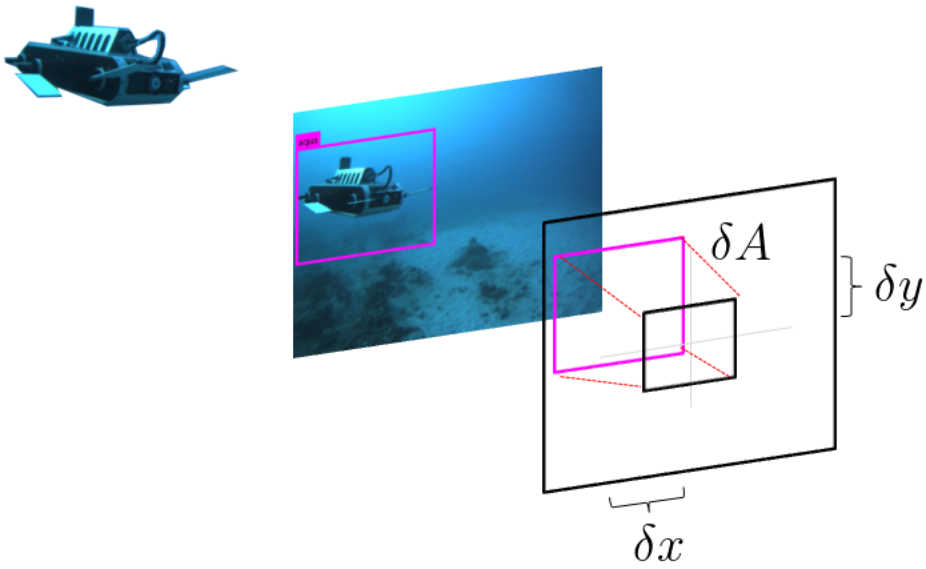


FIGURE 6.4: Errors used by the robot's feedback controller. δx is used for yaw control, δy for depth control, and the error in bounding box area, δA is used for forward speed control.

troller regulates the motion of the vehicle to bring the observed bounding boxes of the target robot on the center of the follower's image, and also to occupy a desired fraction of the total area of the image. It uses a set of three error sources, as shown in Fig. 6.4, namely the 2D translation error from the image center, and the difference between the desired and the observed bounding box area.

The desired roll rate and vertical speed are set to zero and are handled by the robot's 3D autopilot [Meger et al., 2014]. The translation error on the x-axis, δx, is converted to a yaw rate through a PID controller. Similarly, the translation error on the y-axis, δy, is scaled to a desired depth change in 3D space. When the area of the observed bounding box is bigger than desired, the robot's forward velocity is set to zero. We do not do a backup maneuver in this case, even though the robot supports it, because rotating the legs 180^o is not an instantaneous motion. The difference in

area of the observed versus the desired bounding box, namely δA, is scaled to a forward speed. Our controller sends commands at a rate of 10Hz and assumes that a bounding box is detected at least every 2 seconds, otherwise it stops the robot.

6.1.6 Experimental Results

We evaluate each of the implemented methods on the common test dataset using the metrics described below, with n_{images} the total number of test images, n_{TP} the number of true positives, n_{TN} the number of true negatives, n_{FN} the number of false negatives and n_{FP} the number of false positives:

- Accuracy : $\frac{n_{TP}+n_{TN}}{n_{images}}$
- Precision : $\frac{n_{TP}}{n_{TP}+n_{FP}}$ and recall: $\frac{n_{TP}}{n_{TP}+n_{FN}}$
- Average Intersection Over Union (IOU) : Computed from the predicted and ground-truth bounding boxes over all true positive detections (between 0 and 1, with 1 being perfect alignment)
- Localization failure rate (LFR): Percentage of true positive detections having IOU under 0.5 [Cehovin et al., 2015]
- Frames per second (FPS) : Number of images processed/second

Each of the implemented methods outputs its confidence that the target is visible in the image. We chose this threshold for each method by generating a precision-recall curve and choosing the confidence bound which provides the best recall tradeoff for more than 95% precision.

We present the evaluation results in Table 6.3 for each of the algorithms that we considered. The *FPS* metric was measured across five runs on a CPU-only machine with a 2.7GHz Intel i7 processor.

Non-Recurrent Methods

As we can see in Table 6.3, the original TinyYOLOv2 model is the best performing method in terms of IOU, precision, and failure rate. However our results show that the ReducedYOLO model achieves a 14x speedup over TinyYOLOv2, without significantly sacrificing accuracy. This is a noteworthy observation since ReducedYOLO uses 3.5 times fewer parameters compared to TinyYOLOv2. This speedup is crucial for making the system usable on mobile robotic platforms which often lack Graphical Processing Units, and are equipped with low-power processing units. Additionally, note that the precision metric has not suffered while reducing the

Algorithm	ACC	IOU	P	R	FPS	LFR
VGG16a	0.80	**0.47**	0.78	**0.79**	1.68	**47%**
VGG16b	**0.82**	0.39	0.85	0.73	1.68	61%
VGG16c	0.80	0.35	**0.88**	0.64	1.68	70%
VGG15	0.71	0.38	0.65	0.75	1.83	62%
VGG8	0.61	0.35	0.55	0.69	**2.58**	65%
TinyYOLOv2	**0.86**	**0.54**	**0.96**	**0.88**	0.91	**34%**
ReducedYOLO	0.85	0.50	0.95	0.86	**13.3**	40%
RROLO (n=3,z=1)	0.68	0.53	0.95	0.64	**12.69**	15%
RROLO (n=3,z=2)	**0.84**	**0.54**	**0.96**	**0.83**	12.1	**9%**
RROLO (n=6,z=1)	0.81	0.53	0.95	0.80	12.1	11%
RROLO (n=6,z=2)	0.83	**0.54**	**0.96**	0.82	12.03	11%
RROLO (n=9,z=1)	0.81	0.53	0.95	0.80	11.53	**9%**
RROLO (n=9,z=2)	0.80	0.50	**0.96**	0.79	11.36	14%
MDPM Tracker	0.25	**0.16**	0.94	0.26	**142**	**19.3%**
TLD Tracker	**0.57**	0.12	**1.00**	**0.47**	66.04	97%

TABLE 6.3: Comparison of all tracking methods. Precision and recall values based on an optimal confidence threshold.

model, which implies that the model rarely commits false positive errors, an important quality in convoying where a single misdetection could deviate the vehicle off course.

In addition, we found that none of the VGG variants fared as well as the YOLO variants, neither in terms of accuracy nor in terms of efficiency. The localization failure rate of the VGG variants was reduced with the use of batch normalization. Increasing the width of the fully connected layers and imposing regularization penalties on their weights and biases did not lead to an improvement over *VGG16a*. Reducing the total number of convolutional layers, resulting in the *VGG8* model lead to a drastic decrease in both classification and localization performance, which suggests that even when trying to detect a single object, network depth is necessary for VGG-type architectures.

Finally, the TLD tracker [Kalal et al., 2012] performed significantly worse than any of the detection-based methods, mainly due to tracking drift. It is worth noting that we did not reinitialize TLD after the target robot exited the field of view of the follower robot, and TLD could not always recover. This illustrates why model-free trackers are in general less suitable for convoying tasks than detection-based trackers.

Recurrent Methods

All the recurrent methods were trained using features obtained from the ReducedYOLO model, precomputed on our training set. We limit our analysis to the recurrent model using the ReducedYOLO model's features, which we'll refer to as Recurrent ReducedYOLO (RROLO), since this model can run closest to real time on our embedded robot system. Our results on the test set are shown in Table 6.3. For these methods, z denotes the number of LSTM layers, n is the number of frames in a given sequence. Note that the runtime presented here for RROLO methods includes the ReducedYOLO inference time. Finally, while bidirectional recurrent architectures were implemented and tested as well, we exclude results from those models in the table as we found that in our case these architectures resulted in worse performance overall across all experiments.

As can be seen in Table 6.3, we find that the failure rate and predicted confidence can be tuned and improved significantly without impacting precision, recall, accuracy, or IOU. More importantly, we find that the correlation between bounding box IOU (with the ground truth) and the predicted confidence value of our recurrent methods is much greater than any of the other methods, which translates to a more interpretable model with respect to the confidence threshold parameter while also reducing the tracking failure rate. For our best configurations of VGG, YOLO, ReducedYOLO, and RROLO, we take the Pearson correlation r-value and the mean absolute difference between the ground truth IOU and the predicted confidence[5]. We find that RROLO overall is the most correlated and has the least absolute difference between the predicted confidence and ground truth IOU.

Variations in layers and timesteps did not present a significant difference in performance, while yielding a significant reduction in failure rate even with short frame sequences ($n = 3, z = 2$). Furthermore, the frame rate impact is negligible with a single layer LSTM and short time frames, so we posit that it is only beneficial to use a recurrent layer on top of ReducedYOLO. While the best length of the frame sequence to examine may vary based on characteristics of the dataset, in our case $n = 3, z = 2$ provides the best balance of speed, accuracy, IOU, precision and recall, since this model boosts all of the evaluation metrics while retaining IOU with the ground truth and keeping a relatively high FPS value.

Increasing the number of LSTM layers can boost accuracy and recall further, without impacting IOU or precision significantly, at the expense of higher runtime

[5]Pearson correlation r-value: VGG (.70), YOLO (.48), ReducedYOLO (.56), RROLO (.88). Mean absolute difference between confidence predicted and IOU with ground truth: VGG (.37), YOLO (.17), ReducedYOLO (.18), RROLO (.08).

and higher risk of overfitting. We attempted re-balancing and re-weighting the objective in our experiments and found that the presented settings worked best. We suspect that no increase in IOU, precision and recall was observed as there may not be enough information in the fixed last layers of the YOLO output to improve prediction. Future work to improve the recurrent system would target end-to-end experiments on both the convolutional and recurrent layers, along with experiments investigating different objective functions to boost the IOU while making the confidence even more correlated to IOU.

Frequency-Domain Detection

In our implementation of MDPM tracker, non-overlapping sub-windows of size 30×30 pixels over 10 sequential frames are considered to infer periodic motion of the robot. Peaks in the amplitude spectrum in the range 1-3Hz constitute an indicator of the robot's direction of motion. We found that the frequency responses generated by the robot's flippers are not strong and regular. This is due to lack of regularity and periodicity in the robot's flipping pattern (compared to that of human divers), but also due to the small size of the flippers compared to the image size. Consequently, as Table 6.3 suggests, MDPM tracker exhibits poor performance in terms of accuracy, recall, and IOU. The presence of high amplitude spectra at 1-3Hz indicates the robot's motion direction with high precision. However, these responses are not regular enough and therefore the algorithm fails to detect the robot's presence in a significant number of detection cycles. We can see however that the failure rate for this method is one of the lowest among the studied methods, indicating very precise bounding boxes when detections do occur. Additionally, this method does not need training and is the fastest (and least computationally expensive) method, by a significant margin. Therefore given more consistent periodic gait patterns it would perform quite well, as previously demonstrated in [Islam and Sattar, 2017].

Field Trial: Setup

To demonstrate the practicality of our vision-based tracking system, we conducted a set of in-ocean robot convoying field runs by deploying two Aqua robots at 5 meters depth from the sea surface. The appearance of the leading robot was altered compared to images in our training dataset, due to the presence of an additional sensor pack on its top plate. This modification allowed us to verify the general robustness of our tracking-by-detection solution, and specifically to evaluate the possibility of overfitting to our training environments.

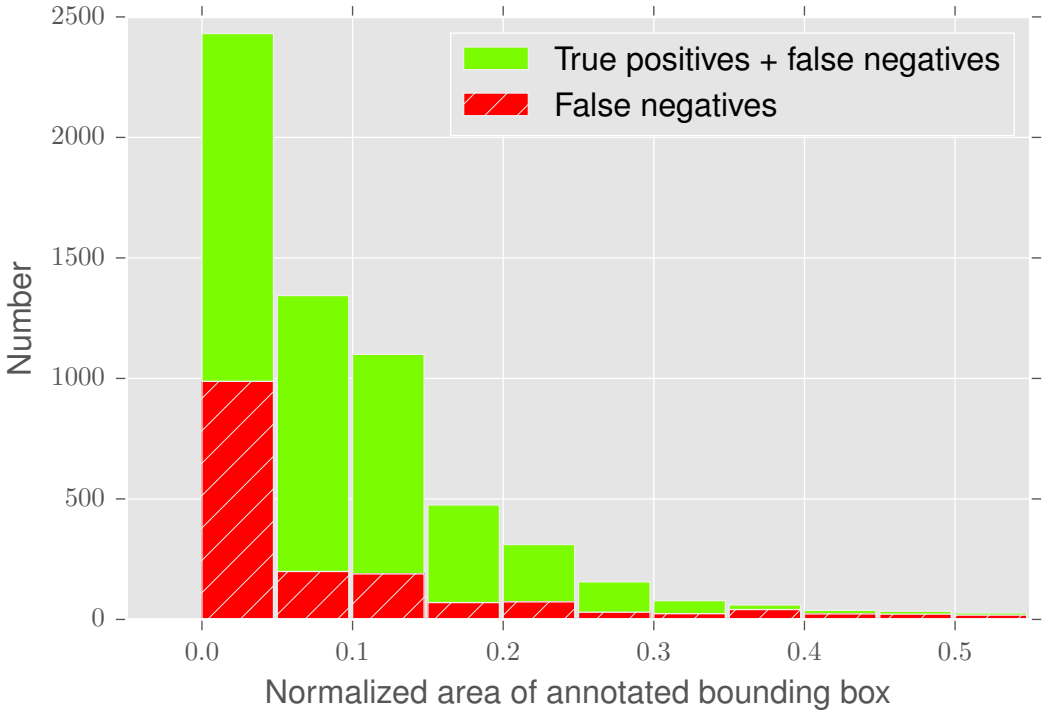


FIGURE 6.5: Histogram of true positive and false negative detections as a function of the area of annotated bounding boxes, as obtained from in-ocean robot tracking runs.

We programmed the target robot to continuously execute a range of scripted trajectories, including maneuvers such as in-place turning, changing depth, and swimming forward at constant speed. We deployed the ReducedYOLO model on the follower robot, which operated at 7 Hz onboard an Intel NUC i3 processor without GPU acceleration or external data tethering. Moreover, the swimming speeds of both robots were set to be identical in each run ($0.5 - 0.7\,m/s$), and they were initialized at approximately two meters away from one another, but due to currents and other factors the distance between them (and the scale of observed bounding boxes) changed throughout the experiment runs.

Field Trial: Results

We configured the follower robot to try to track the leading robot at a fixed nominal distance. This was achieved by setting the desired bounding box area to be 50% of the total image area, as seen in Fig. 6.4.

The ReducedYOLO detector consistently over-estimated the small size of the target. Nevertheless, Fig. 6.6 indicates that the bias error in bounding box centers between detected versus ground truth was consistently low in each frame, regardless of the target size, on average within 10% of the image's width to each other.

This is notable due to frequent occurrences where the robot's size occupied less than 50% of the total area of the image.

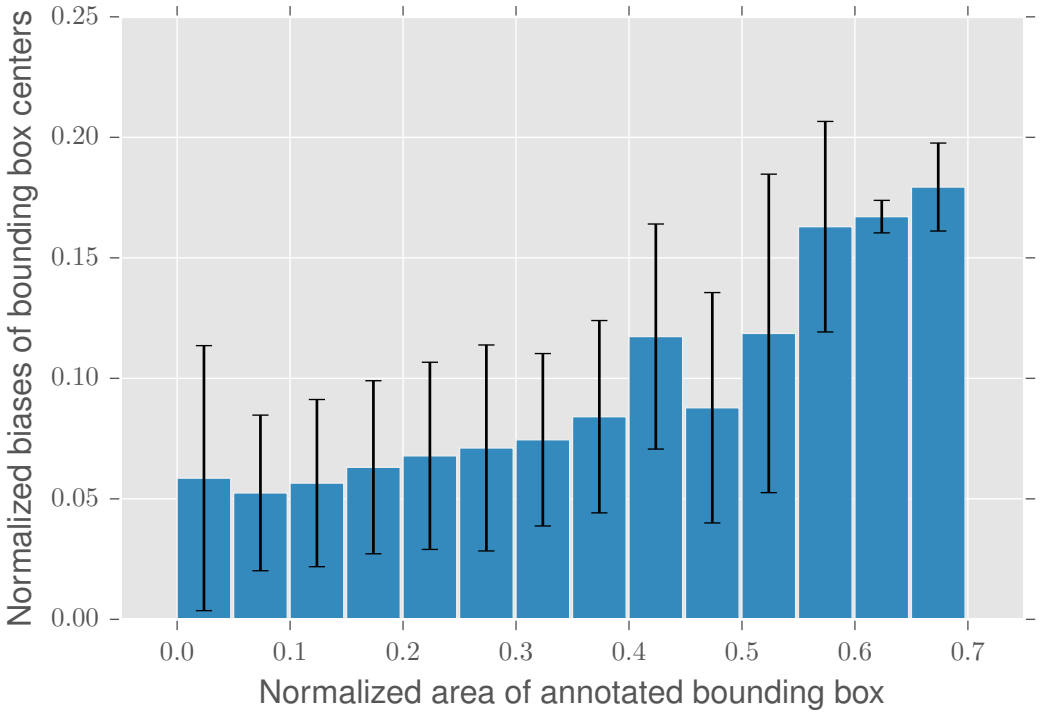


FIGURE 6.6: Histogram of average biases between detected vs. annotated bounding box centers, as obtained from in-ocean robot tracking runs. Bars indicate 1σ error.

We also evaluated the performance of our system in terms of the average "track length", defined as the length of a sequence of true positive detections with a maximum of 3 seconds of interruption. Across all field trial runs, the follower achieved 27 total tracks, with an average duration of 18.2 sec ($\sigma = 21.9$ sec) and a maximum duration of 85 sec. As shown in Fig. 6.7, the vast majority of tracking interruptions were short, specifically less than a second, which did not affect the tracking performance, as the leading robot was re-detected. The majority of these tracking interruptions were due to the fact that the annotated bounding box area of the leading robot was less than 20% of the total area of the follower's image. Sustained visual detection of the target, despite significant visual noise and external forces in unconstrained natural environments, and without the use of engineered markers, reflects successful and robust tracking performance in the field.

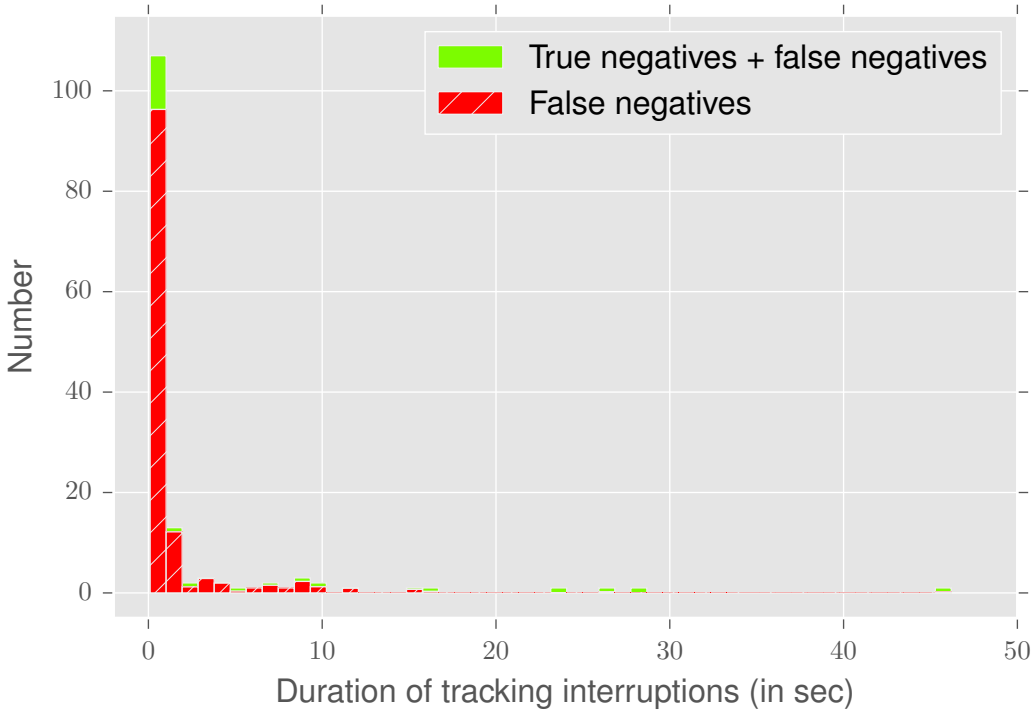


FIGURE 6.7: Histogram of true negative and false negative classifications in terms of their duration for our ReducedYOLO model, as obtained from in-ocean robot tracking runs.

7 VISUAL LOCALIZATION AND MAPPING FOR 3D NAVIGATION

In order to enable the robotic systems presented in the previous chapters navigate in GPS-denied environments we have developed a sensor-based simultaneous localization and mapping (SLAM) system, which estimates the 7DOF pose of the robot and the 3D points of the map. We are interested in metric, true-scale SLAM due to the fact that it can be used in a closed feedback loop for autonomous navigation by most mobile robotic vehicles. This is particularly relevant for robots that operate in GPS-denied environments, or that need higher accuracy than the one provided by normal GPS.

7.1 Scale Drift In Visual Localization And Mapping

Monocular vision-based SLAM systems can provide feedback for position control, however they typically are unable to maintain constant *scale* across time, meaning the distance units change over time due to unobservability of the magnitude of the translation vector. Maintaining constant scale can be difficult for monocular systems, for instance when the camera rotates in-place [Strasdat et al., 2010, Hesch et al., 2014, Jones and Soatto, 2011], with insufficient baseline to triangulate new points in the map. Estimation error in the map feeds back into the localization error, which creates a feedback loop which results in *scale drift*. This error creates unnecessary difficulties for the controllers that have to act on estimated position, resulting in degraded performance that might even expose the vehicle to risk. In addition, another drawback of monocular SLAM systems is that the metric scale of a map estimated with a single camera is going to be different than the true distance units of the world. In other words, not only does the scale drift over time, but the true scale is *unobservable*.

7.2 Inertial And Multi-Camera Localization

We present a system that drastically reduces the risk of the drawbacks discussed above in three ways:

- by using visual correspondences of keypoints from the motion of the second camera in stereo pair, which usually has sufficient baseline to perform correct triangulation of new 3D map points
- by using visual correspondences of keypoints from the synchronized stereo pair, which has a fixed baseline and can provide a steady supply of new 3D map points
- by using sensor measurements from an Inertial Measurement Unit

Our system, shown in Fig. 7.1, tightly integrates measurements obtained from an Inertial Measurement Unit (IMU) with stereo camera ego-motion estimation in a continuous optimization setting. Inertial and visual sensors are ubiquitous in robotics systems, and they work in a wide range of challenging environments, including underwater, in the air, and on the water surface.

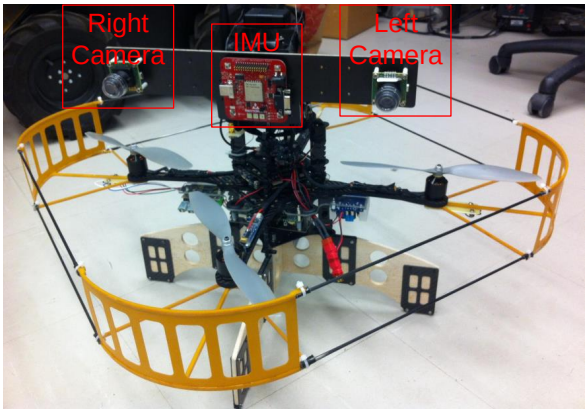


FIGURE 7.1: Pelican quadrotor with mounted sensor module

We show that the combination of these two types of sensors results in more efficient, more robust, and more accurate SLAM than what can otherwise be obtained via any of the individual sensing modalities in this real-time context. It also has the following related advantages:

- it has obvious connections to the human vestibular and visual system, whose combination enables balance and spatial awareness
- it leads to desirable observability properties of the estimated pose and velocity even in cases where monocular camera + IMU SLAM is challenging (i.e. in-place rotations) [Martinelli, 2013, Hesch et al., 2014, Jones and Soatto, 2011]

- there is increased availability of cheap cameras and inertial sensors
- they can operate passively in a diverse set of challenging environments, such as in space or underwater, assuming sufficient visibility but no external supporting equipment

Our system is based on the well-established ORBSLAM [Mur-Artal et al., 2015], which performs SLAM with a single monocular camera and, therefore, suffers from scale drift problems as discussed above. Due to the richer availability of sensory information, we manage to overcome all of these limitations in most scenarios[1]. Our system continuously optimizes the 3D points in the map, as well as the camera's trajectory from which said points were observed. In addition, it maintains uncertainty information about each triangulated 3D map point, in the form of a marginalized covariance matrix, which is updated whenever the point is re-observed. We currently use this information to reject triangulations that are almost ill-posed or have unreasonably high uncertainty in the estimated depth, which improves the accuracy of our estimates and avoids potential instabilities.

Benefits Of IMU Measurements

In addition to these advantages, the inclusion of the IMU in the estimation process provides a few other practical benefits to our system. First, it enables the prediction of keypoint locations in the next image, thus reducing matching time and boosting overall efficiency. Second, it increases the accuracy of the system by involving its measurements tightly in the optimization. This means that the IMU measurements can prevent the system from getting lost during sharp turns when the cameras might suffer from motion blur. Third, and most importantly, the IMU increases the robustness of the system because it enables effective recovery from loss of tracking, whenever that is possible. Whereas the only response of a monocular SLAM system to loss of tracking is to indefinitely wait for an already visited place to be seen again, our system can use the stereo pair to initialize a new sub-map and it can then use the IMU to align the sub-map with the existing map. In fact, our system can do this sub-map initialization from a single frame, without requiring any special motions such as a swiping translation, which is commonly required by monocular systems [Klein and Murray, 2007].

[1] There are still motions that result in unobservable scenarios [Tribou et al., 2015]

Related Work

This increased robustness is entirely due to the rich information offered by the synchronized stereo + IMU sensor module. Our work is similar to other methods proposed such as PTAM [Klein and Murray, 2007], MCPTAM [Harmat et al., 2012], SVO [Forster et al., 2014], and works by Shen [Shen et al., 2015] and others [Leutenegger et al., 2015, Christian Forster, 2015]. All these methods perform feature-based tracking to estimate pose and orientation with respect to a world-fixed frame of reference. PTAM was initially meant for augmented reality applications but has been extended many times. For example in MCPTAM [Harmat et al., 2012], which tries to overcome some limitations by enabling feature tracking in multiple non-overlapping camera views. This method can work well, but does not integrate IMU measurements, and is computationally intensive. SVO [Forster et al., 2014] attempts to reduce the computational requirements by mixing direct (feature based tracking) and semi-direct (photometric based tracking) methodologies to reduce the necessity to perform feature matching at every frame. Works by Shen [Shen et al., 2015] propose methods for tightly coupling an IMU with mono and stereo vision, however BRIEF descriptors are used when performing stereo correspondences, which have been shown to be more computationally expensive than ORB features. To our knowledge, none of these methods perform single-frame submap initialization.

7.2.1 Frame Definitions

We define the following four frames for our estimator: W is the fixed world inertial frame of reference in which all the quantities of interest are expressed, LC and RC are the moving frames of the left and right cameras respectively, and I is the moving IMU frame. The two camera frames and the IMU frame are fixed with respect to each other, and we assume to know in advance the transformations that link them. In fact we have done the calibration using Kalibr [Furgale et al., 2012]. In our particular case, the rotation of W is the local North-East-Down frame. The *localization state vector* that we want to estimate is the following:

$$\mathbf{S} = \begin{bmatrix} {}^{LC}_{W}\mathbf{q} & {}^{LC}\mathbf{p}_W & {}^{W}\mathbf{v}_I & \mathbf{b}_g & \mathbf{b}_a \end{bmatrix} \tag{7.1}$$

where ${}^{LC}_{W}\mathbf{q}$ is a unit quaternion that expresses the rotation from the fixed world frame to the left camera frame, ${}^{LC}\mathbf{p}_W$ is the position of the world frame in the coordinates of the left camera frame, ${}^{W}\mathbf{v}_I$ is the velocity vector of the IMU in world frame, $\mathbf{b}_g$ is the gyroscope bias, and $\mathbf{b}_a$ is the accelerometer bias affecting the IMU

measurements. Upon every incoming pair of stereo images, we create a frame object that has this instantaneous localisation information as its state vector. In our system frames are created at the camera frame rate, which is 15Hz. We also store all the IMU measurements since the last stereo image pair was received, which from an IMU that runs at 100Hz means that on average each frame has about 7 IMU messages that precede it.

Our system currently assumes that both cameras use the pinhole projection model, are in-phase with the IMU measurements using hardware triggering, and also that they share the same exposure times. The *mapping state vector* that we want to estimate is a dynamic-size vector that comprises a list of selected map points and a list of selected keyframes.

Keyframes are snapshots of frame objects along the trajectory that are dispersed in a way that balances sufficient visual overlap to exploit for triangulation, and enough visual differences to lead to potentially sufficient baseline for motion stereo. Each keyframe contains its own copy of the pose of the left and right camera. Keyframes also contain the list of keypoints that were detected in the pair of stereo images, and their correspondences to existing triangulated 3D map points. Ensuring that many of these correspondences from 2D pixel keypoints to 3D map points are made correctly through triangulation of new map points or identification and merging of identical ones is an essential ingredient to the success of the SLAM system.

Each map point contains its 3D position in the world frame, an ORB descriptor that best represents the descriptors of the keypoints that are currently associated with it, an average viewing direction towards the optical centers of the keyframes that observed it, and a minimum and maximum range from which a keyframe can observe it, in order to limit spurious triangulations. In addition to that, we also store a 3×3 marginal covariance matrix that describes our uncertainty in the position of the map point.

Just like in [Mur-Artal et al., 2015] our system simultaneously runs three interacting threads: tracking, local mapping, and loop closing. This is shown in Fig. 7.2. Our system builts upon theirs, and uses the g2o framework [Kümmerle et al., 2011] for graph-based nonlinear optimization throughout the system. In the following sections we describe each of the threads, but it is worth noting at this moment that the tracking thread mainly deals with the correct state estimation of frames and IMU data processing, while the other two threads deal exclusively with estimating the pose of keyframes, without processing any IMU data. That is, we do not model IMU measurements as factors in the connected graph of keyframes, but we rather take a more filtering-type approach, which will be explained below.

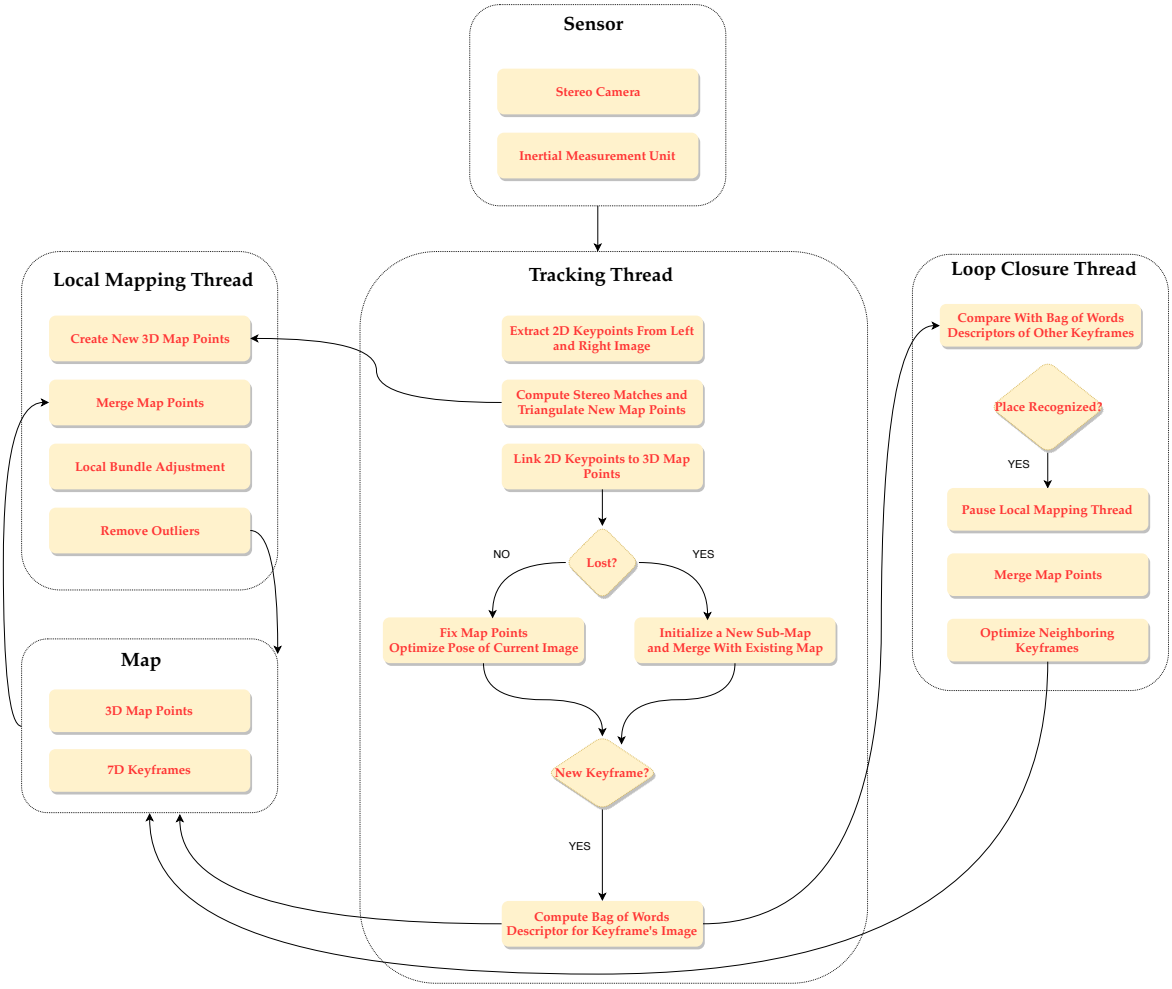


FIGURE 7.2: System Overview

7.2.2 Tracking Thread

The tracking thread is the one that receives each stereo image pair. It is responsible for extracting Harris keypoints [Harris and Stephens, 1988, Shi and Tomasi, 1994] and ORB descriptors [Rublee et al., 2011] on each image, and undistorting those keypoints. This happens in parallel on two separate sub-threads in order to speed up processing time. The tracking thread is also the one receiving the IMU measurements. It initializes stereo frames and populates them with the IMU messages that were received since the previous frame. The tracking thread also computes high-quality stereo matches from the left descriptors to the right at 15Hz. These matches satisfy strict thresholds over distances from the epipolar lines. The stereo keypoint matches are going to be used for initializing the map for the first time, for sub-map initialization, and also in the local mapping thread. These steps are illustrated in Fig. 7.2 and they are going to be described in more detail in sections 7.2.2, 7.2.2 and 7.2.3 respectively.

After initializing the frame objects the tracking thread is responsible for associating 2D keypoints to existing 3D map points. It doesn't do this by brute force matching over the entire map. Instead, it uses the previous frame's state estimate and integration of the IMU measurements that came after it to predict the rough pose of the current frame (see section 7.2.2). It then projects the map points associated with the previous frame to the current frame's image plane, and searches in very small local neighborhoods around those predicted pixels. If this association fails the system transitions into the LOST state, as shown in Fig. 7.2.

The tracking thread refines the predicted state of the current frame, by minimizing the reprojection error of its associated keypoints while also not deviating from the integrated IMU state. This step is the first part of "Current Pose Estimation" in the flowchart of Fig. 7.2 and it is going to be described in more detail in section 7.2.2. The second part does an additional refinement optimization on the pose of the current frame. This optimization considers a limited number of keyframes that share map point observations with the current frame. It then attempts to establish more such associations in the current frame by projecting each keyframe's map points that are visible to the current frame, and refines its camera pose even further.

Finally, the tracking thread checks if the current frame is a good candidate for being cloned into a keyframe. If so, it is dispatched to the local mapping thread for processing, and the thread starts processing the next set of incoming measurements.

Single-Frame Map Initialization via Stereo Triangulation

One of the advantages of relying on a stereo camera is that it allows our system to initalize a map from a single pair of images without requiring translational "swiping" motion of the camera that is required to initialize monocular systems [Klein and Murray, 2007]. As long as the observed scene is textured enough and is near enough to the camera to perform accurate triangulation – that is at most 10-30 times the baseline in practice – this step is going to be successful. Our system performs stereo triangulation in two steps:

Linear Triangulation: In the first step, we use Hartley's linear triangulation method [R. Hartley, 1997], which sets up the following constrained least squares problem: let $\mathbf{X} = [A \ B \ C \ D]^\top$ be the homogeneous coordinates of a 3D point in the world frame W. Using the 3×4 projection matrices $\mathbf{P}_{\text{left}}$ and $\mathbf{P}_{\text{right}}$ of the two cameras we get the equations that describe the projection of rays: $[u \ v \ 1]^\top = \mathbf{P}_{\text{left}}\mathbf{X}$ and $[u' \ v' \ 1]^\top = \mathbf{P}_{\text{right}}\mathbf{X}$. Note, that here we assume the poses of the two cameras in the world frame are known and fixed. From these we get four linear equations that constrain the homogeneous coordinates of the point, and since the equations describe rays, we can constrain the variable artificially by requiring that $\|\mathbf{X}\| = 1$.

Then the problem formulation for linear triangulation becomes

$$\mathbf{QX} = \mathbf{0} \tag{7.2}$$

$$\text{subject to } \|\mathbf{X}\| = 1 \tag{7.3}$$

with

$$\mathbf{Q} = \begin{bmatrix} uP_{\text{left},3}^{\top} - P_{\text{left},1}^{\top} \\ vP_{\text{left},3}^{\top} - P_{\text{left},2}^{\top} \\ u'P_{\text{right},3}^{\top} - P_{\text{right},1}^{\top} \\ v'P_{\text{right},3}^{\top} - P_{\text{right},2}^{\top} \end{bmatrix}_{4\times 4} \tag{7.4}$$

where $P_{\text{left},1}^{\top}$ denotes the first row of the projection matrix of the left camera in the stereo pair. This is a linear least squares problem with norm constraints. It is actually identical to the Rayleigh Quotient problem in physics:

$$\underset{\mathbf{X}}{\text{argmin}} \|\mathbf{QX}\| \tag{7.5}$$

$$\text{subject to } \|\mathbf{X}\| = 1 \tag{7.6}$$

Its solution is given by the eigenvector that corresponds to the smallest eigenvalue of the matrix $Q^{\top}Q$. We use SVD to compute this eigenvector and then we convert from homogeneous coordinates to Euclidean coordinates in $\mathcal{R}^3$, which corresponds to the initial estimate of the map point's position.

Refining Triangulation: Our system further improves on the triangulation estimate above by doing a second round of optimization, which computes the 3D position of the map point in world coordinates by minimizing a weighted-norm reprojection error. This second optimization round weighs ORB keypoints detected at closer scales higher than those detected at farther scales in the scale pyramid. This is because keypoints that are better localized on the image will most likely be better triangulated in 3D, so their error should be penalized more. These weights are encoded by the diagonal matrix $\mathbf{\Sigma}_z$. Then the second optimization for refining the map point position is

$$\underset{{}^{W}\mathbf{f}}{\text{argmin}} \left\| [u\ v\ u'\ v']^{\top} - \mathbf{h}({}^{LC}_{W}\mathbf{q}, {}^{LC}\mathbf{p}_{W}, {}^{RC}_{W}\mathbf{q}, {}^{RC}\mathbf{p}_{W}, {}^{W}\mathbf{f}) \right\|^2_{\mathbf{\Sigma}_z} \tag{7.7}$$

which is an unconstrained nonlinear least squares problem. Its solution is obtained iteratively, and it is initialized by the solution found in the first Rayleigh Quotient optimization, described above. In addition, after this second optimization takes

place we compute the covariance of the nonlinear least squares estimator, which describes the uncertainty of the triangulation, as follows:

$$\text{cov}({}^{W}\mathbf{f}) = (\mathbf{J}_h^\top \boldsymbol{\Sigma}_z^{-1} \mathbf{J}_h)^{-1} \tag{7.8}$$

where $\mathbf{J}_h$ is the Jacobian of the stereo projection function $\mathbf{h}$ evaluated at the map point estimate from the linear triangulation step. We assume a Gaussian distribution of the noise in the estimate of the map point position, and we note that whether this covariance underestimates the real error (making this least squares estimator inconsistent) strongly depends on whether the initial triangulation estimate is close to the optimal solution, which in turn depends on the pixel noise characteristics of the camera and the pixel localisation of the keypoint. We performed a Monte-Carlo simulation which indicates that even for a keypoint localisation error of 3 pixels, the probability that the covariance computed from the linearized least squares problem leads to a consistent estimator quickly drops off as a function of the depth of the real point. This is shown in Fig. 7.3

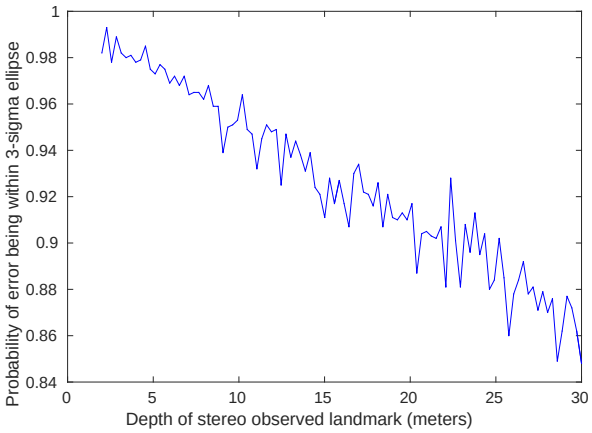


FIGURE 7.3: The probability that the 3-sigma uncertainty ellipse around the map point's least squares estimator includes its true position quickly drops off as depth increases. This is for keypoint localisation error of 3 pixels. This empirical probability drops faster at higher noise levels. This further supports the argument for not trusting triangulation results that are too far away, given the camera's and the keypoint detector's error characteristics, as they lead to *inconsistent* estimators.

Outlier Rejection Criteria: Given these considerations we accept the triangulation as successful if a series of outlier detection tests are passed. The tests include the low-parallax test from [Mur-Artal et al., 2015], χ^2 tests on the reprojection error under the Huber loss, negative estimated depth, excessive depth variance from the position covariance computed above, tests on the conditioning of the least-squares problem itself, and also maximum/minimum cutoffs on the estimated depth. Even

FIGURE 7.4: Sub-map initialization following an occlusion: the green points in the tracking phase correspond to 3D map points that are stable in the map. As the cyclist occludes those points, visual tracking gets lost, but IMU integration maintains spatial continuity. When the occlusion stops, a new map is initialized and merged with the previous one.

though we employ this series of aggressive outlier rejection tests there are still a few spurious points that pass into the pipeline, but each optimization step deals with them separately. The end result of this 2-view/stereo triangulation scheme is that it enables map initialization from a single pair of images, without requiring translational motion, unlike monocular visual SLAM systems.

Sub-Map Initialization

Sub-map initialization is used to recover from events that cause few keypoint associations and the system is unable to accurately estimate the state and is said to be lost. These events include sudden occlusions such as moving objects entering the field of view, motion blur, and very fast abrupt turns which relative to the frame rate. When the system becomes lost, it enters the sub-map initialization process, which simultaneously performs single-frame map initialization as described in the previous section, while at the same time continually integrating the IMU angular and linear velocities and computing a new pose estimate based on the previous pose. Once initialization is successful, the sub-map containing a keyframe and map points is merged with the existing map. This is better illustrated in Fig. 7.4.

Short-Term Feature Prediction Using Inertial Measurements

Upon reception of a new stereo image pair the tracking thread needs to compute a rough estimate of the pose of the new image frame, given the pose of the previous frame, so that it can perform predict where the map points of the previous frame

are going to be projected on the image plane of the current frame. In order to do this our system performs IMU integration on the IMU measurements that have arrived between the previous and the current images. Thus, it computes the relative motion of the IMU from the previous frame, and by extension, the relative motion of the stereo camera. Although this scheme is accurate for the rotational component, it is noisy for the relative translation due to the double integration of the accelerometer. So, during the step of keypoint prediction our system ignores the relative translation of the IMU integration, and instead uses the relative displacement of the two previous frames. This method seems to work quite well even under heavy rotations. In fact our system achieve average keypoint prediction errors in the order of 4-5 pixels.

Given IMU measurements $\boldsymbol{\omega}_m$ and $\boldsymbol{\alpha}_m$ for the angular velocity and linear acceleration of the IMU frame, respectively, we model the errors affecting the measurements as follows, similarly to [Mourikis and Roumeliotis, 2007]:

$$\boldsymbol{\omega}_m = {}^{I}\boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g \tag{7.9}$$

$$\boldsymbol{\alpha}_m = \mathbf{R}({}^{I}_{W}\mathbf{q})({}^{W}\boldsymbol{\alpha} - {}^{W}\mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a \tag{7.10}$$

where ${}^{W}\boldsymbol{\alpha}$ and ${}^{I}\boldsymbol{\omega}$ are the true values, the noise is white Gaussian, and the biases are modeled as Brownian motion processes with $\dot{\mathbf{b}}_g = \mathbf{0}$, $\dot{\mathbf{b}}_a = \mathbf{0}$, and $\mathbf{R}()$ denotes the rotation matrix resulting from the corresponding quaternion. The linear acceleration and angular velocity estimates used in the IMU state integration are $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}_m - \hat{\boldsymbol{b}}_a$ and $\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\boldsymbol{b}}_g$. With that in mind, we perform the Forward Euler integration of the IMU state using the following equations:

$${}^{W}_{I}\hat{\mathbf{q}}_{t_{k+1}} = {}^{W}_{I}\hat{\mathbf{q}}_{t_k} \otimes \begin{bmatrix} \frac{\hat{\boldsymbol{\omega}}}{\|\hat{\boldsymbol{\omega}}\|}\sin(\|\hat{\boldsymbol{\omega}}\|\frac{(t_{k+1}-t_k)}{2}) \\ \cos(\|\hat{\boldsymbol{\omega}}\|\frac{(t_{k+1}-t_k)}{2}) \end{bmatrix}$$

$$\dot{\hat{\mathbf{b}}}_g = \mathbf{0}_{3\times 1}\,, \quad {}^{W}\dot{\hat{\mathbf{v}}}_I = \mathbf{R}({}^{W}_{I}\hat{\mathbf{q}})\hat{\boldsymbol{\alpha}} + {}^{W}\mathbf{g} \tag{7.11}$$

$$\dot{\hat{\mathbf{b}}}_a = \mathbf{0}_{3\times 1}\,, \quad {}^{W}\dot{\hat{\mathbf{p}}}_I = {}^{W}\hat{\mathbf{v}}_I \tag{7.12}$$

Frame Pose Refinement & Tight IMU Coupling

After having assigned an initial estimate to the pose of the newly-created frame, the tracking thread has established correspondences between the 2D keypoints of said image frame and the existing map points of the previous image frame. It then refines that estimate by optimizing over the poses of the two frames in order to minimize two costs: the first is the reprojection error of those map points (denoted by RE), and the second is a relative motion error from the IMU motion (denoted by

RME). During this optimization the map point positions are treated as fixed. Let $\mathbf{S}_0$ denote the localisation state vector of the previous frame, and $\mathbf{S}_1$ denote the localisation vector of the current frame, as explained in Eq. 7.1. Then the optimization problem our system solves at this step is the following:

$$\underset{\mathbf{S}_0,\mathbf{S}_1}{\operatorname{argmin}} \quad \mathrm{RE}(\mathbf{S}_1) + \mathrm{RME}(\mathbf{S}_0, \mathbf{S}_1) \tag{7.13}$$

where the reprojection error is

$$\mathrm{RE}(\mathbf{S}) = \sum_{{}^{W}\mathbf{f}\leftrightarrow\mathbf{z}_f} \left\| \mathbf{z}_f - \mathbf{g}({}^{LC}_{W}\mathbf{q}, {}^{LC}\mathbf{p}_W, \mathbf{f}) \right\|^2_{\mathbf{\Sigma}_z} \tag{7.14}$$

where $\mathbf{g}$ is the function that projects a point from the world frame to a pixel in the camera's image plane. If we denote the relative camera motion that is obtained from the IMU motion, expressed in the previous frame's coordinates, by $\bar{\Delta\mathbf{q}}_1^0, \bar{\Delta\mathbf{p}}_1^0$ and the same quantity obtained from the state vectors of the two frames as $\hat{\Delta\mathbf{q}}_1^0, \hat{\Delta\mathbf{p}}_1^0$. In addition let the integrated IMU velocity starting from the state vector $\mathbf{S}_0$ until the time of the current frame be ${}^{W}\bar{\mathbf{v}}_{I_1}$. Denote also the IMU velocity in the state vector $\mathbf{S}_1$ as ${}^{W}\mathbf{v}_{I_1}$ Then the relative motion error is

$$\mathrm{RME}(\mathbf{S}_0, \mathbf{S}_1) = \left\| \begin{bmatrix} \bar{\Delta\mathbf{q}}_1^0 \otimes \hat{\Delta\mathbf{q}}_0^1 \\ \bar{\Delta\mathbf{p}}_1^0 - \hat{\Delta\mathbf{p}}_1^0 \\ {}^{W}\bar{\mathbf{v}}_{I_1} - {}^{W}\mathbf{v}_{I_1} \\ \mathbf{b}_{g_1} - \mathbf{b}_{g_0} \\ \mathbf{b}_{a_1} - \mathbf{b}_{a_0} \end{bmatrix} \right\|^2_{\Sigma_{\mathrm{IMU}}} \tag{7.15}$$

where Σ_{IMU} expresses the uncertainty of the IMU propagation. This tight integration of the IMU in the tracking and optimization loop allows improved rotation estimates as well as accurate velocity estimates, which will be better shown in Fig. 7.6 in the Experiments Section.

7.2.3 Local Mapping Thread

The local mapping thread receives newly-created and initalized keyframes from the tracking thread and it is responsible for using them in order to triangulate new map points for the system. Therefore, the role of this thread is critical as it needs to supply the other two threads with a sufficient number of well-triangulated map points so that their optimizations can produce meaningful results. This step is illustrated in the system flowchart of Fig. 7.2.

Upon receiving a new keyframe, it establishes links between the new arrival and existing keyframes in the map. These connections signify visual overlap among the keyframes, so in [Mur-Artal et al., 2015] this is called the *Covisibility Graph*. Connections are weighted according to the number of jointly-observed map points [Strasdat et al., 2011]. This enables the system to search in the visual vicinity of a keyframe, without setting positional or time bounds or doing position-based nearest neighbour search.

Once new map points have been created, the local mapping thread tries to merge them with existing map points, to avoid duplicate representations. This is done in the neighbourhood of the new keyframe, as described above, so it is done very efficiently. This step is also depicted in the flowchart of Fig. 7.2. Once duplicate map points have been merged, the system proceeds to its most computationally demanding step: local bundle adjustment, which is going to be described in more detail below.

Creating New Map Points

Monocular SLAM systems only have one option when it comes to triangulating new map points: that is motion stereo, assuming there is sufficient baseline between pairs of keyframes. When the baseline is too short, for instance during in-place rotations, the system is in the risk of scale drift.

Since our system uses stereo images, it has a few more options in such pathological cases. For example, if the left camera is undergoing in-place rotation the right camera is not. We exploit this property to enable as much as possible a steady supply of newly-triangulated map points. We have observed that our system needs at least 200 new map points at each new keyframe in order to perform well. The local mapping thread receives the stereo matches from the left image to the right image, as computed in the tracking thread. It then uses the triangulation technique presented in the map initialization section 7.2.2, to generate 50-200 high quality map points.

In addition to that, our system examines a selection of overlapping keyframes in the visual neighbourhood of the newly-created one. For each neighboring keyframe we do the triangulation of section 7.2.2 between its left image and the current keyframe's left image. This is usually harder than triangulating jointly observed points on stereo images, because in this case pairs of keyframes can have very different viewpoints towards the scene.

When the baseline is small, it does not, which is why our system also matches keypoints from the current keyframe's left image to the neighboring keyframes' right image. It is worth repeating that all these matching processes are done via

projections of map points to the image plane, and not via brute force examination of pairs of keypoints, which makes them very efficient. Thus, our system overcomes the drawback of such pathological cases that plague monocular vision-based SLAM systems.

Merging New Map Points With Existing Map

Once new map points get created they need to be integrated into the map. Therefore duplicates need to be merged. Following [Mur-Artal et al., 2015] our system goes through a selection of the co-visible keyframes of the current one. For each such visual neighbor it projects the map points of the new keyframe to the neighbor, and efficiently identifies matches using Hamming distances on the space of binary ORB descriptors. Vice-versa, it goes through the visual neighbors and projects their Map Points to the current one, also establishing matches.

The map points that are merged need to have their mean and covariance updated, so two sources of information need to be combined into one. We could perform the uncertainty update of this operation using covariance intersection techniques, but we have opted for treating this update in the same way one would do it in an Extended Kalman Filter. We regard the act of merging as obtaining a new pixel measurement for an existing map point, and we update its estimate so as to minimize the minimum mean squared error criterion, given a prior estimate. This is akin to maintaining an EKF for each map point, and updating it whenever new observations are established.

More concretely, if the prior Gaussian estimate of the map point is given by $({}^{W}\mathbf{f}_{t-1},\ \mathbf{\Sigma}_{t-1})$, and we merge it with an existing map point with pixel observation z on a neighboring keyframe with pose ${}^{LC}_{W}\mathbf{q}, {}^{LC}\mathbf{p}_{W}$, then the minimum mean squared error update is as follows:

$$\begin{aligned} \mathbf{K} &= \mathbf{\Sigma}_{t-1}\mathbf{J}_g^\top(\mathbf{J}_g\mathbf{\Sigma}_{t-1}\mathbf{J}_g^\top + \mathbf{\Sigma}_z)^{-1} \\ {}^{W}\Delta\mathbf{f} &= \mathbf{K}(\mathbf{z} - \mathbf{g}({}^{LC}_{W}\mathbf{q}, {}^{LC}\mathbf{p}_{W}, {}^{W}\mathbf{f}_{t-1})) \\ \mathbf{\Sigma}_t &= \mathbf{\Sigma}_{t-1} - \mathbf{K}\mathbf{J}_g\mathbf{\Sigma}_{t-1} \end{aligned} \tag{7.16}$$

where $\mathbf{g}$ is the same reprojection function introduced in Eq. 7.14, and $\mathbf{J}_g$ is its Jacobian evaluated at the previous mean estimate ${}^{W}\mathbf{f}_{t-1}$.

Local Bundle Adjustment

Local Bundle Adjustment takes the newly-created keyframe and its co-visible neighbours, as well as all their map points, and it performs a fixed number of reprojection

error minimization steps. This is similar to Eq. 7.14, with the difference that local bundle adjustment optimizes over the keyframe poses as well as over the map point positions. Hence, there is usually a large number of variables involved in this step, in the order of 50 keyframes and 1500 map points. On an i7 computer this step usually takes in the order of 0.2-0.5 seconds, while executed in the background. It is critical for refining the estimates of the localisation and mapping variables, but also for the process of identifying spurious map points and badly-estimated keyframes that need to be pruned out.

7.2.4 Loop Closing Thread

The loop closing thread is responsible for identifying scenes that have been visited before. It does this by comparing the Bag-of-Words descriptor [Galvez-López and Tardos, 2012] that is stored in each keyframe before it is inserted in the map. The system maintains an index from visual words to the keyframes that observed them, so, querying this index for potential place recognition candidates that have observed the words in the current keyframe is very efficient. The same index is used by the tracking thread in order to search for local relocalisation candidate keyframes that match the BoW descriptor of the newly-created frame. The loop closing error is propagated to the visual neighbors using an incrementally updated spanning tree of the co-visibility graph [Mur-Artal et al., 2015].

7.3 Experimental Results

In this section we describe the hardware used and the calibration done, prior to running any of our experiments. In the following sections we describe several experiments that validate the accuracy of our state estimator.

7.3.1 Synchronized Sensor Module

Our custom sensor module (as shown in Fig. 7.1) consists of two IDS Imaging uEye LE cameras along with a VectorNav VN-100 IMU mounted to a rigid platform. The cameras are mounted in a stereo fashion with a baseline of 30cm and capture 752 x 480 resolution images. The IMU runs at 100Hz and provides a synchronous trigger to the cameras at ∼15Hz. A dual core 3.4 GHz Intel NUC5i7RY baseboard, with 16GB of RAM and a 512GB SSD, is used to acquire the images and perform state estimation. The sensor module and computer together weigh ∼450 grams.

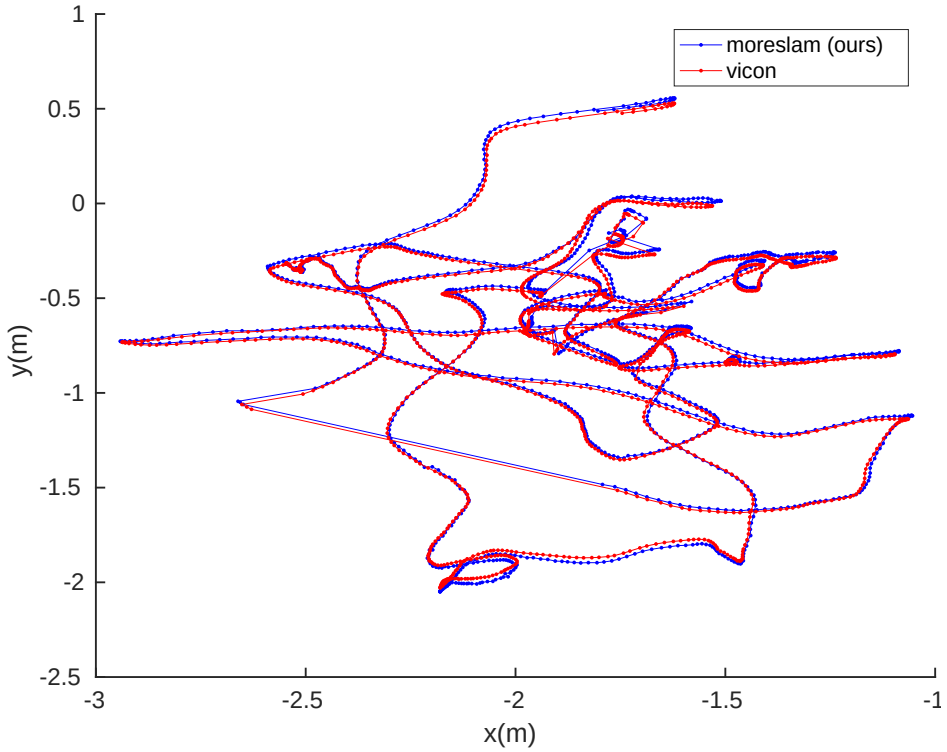


FIGURE 7.5: View of the estimated vs Vicon ground truth path projected on the xy plane.

Our experiments were performed on: (1) a hand-held unit used in experiments described in sections 7.3.2 and 7.2.2, and (2) a submersible hand-held unit used in the reconstruction of an underwater shipwreck described in section 7.3.3.

7.3.2 Validation From Vicon Ground Truth

To validate the accuracy of the state estimates, we used a Vicon motion capture system which provides independent pose and orientation measurements - the plot of the horizontal plane is shown in figure 7.5. Compared to the Vicon over a 120s interval and a 4m x 4m area, the pose estimates of our system have the following errors:

$\tilde{x}$ (mm)	$\tilde{y}$ (mm)	$\tilde{z}$ (mm)	$\tilde{\theta}_x$ (deg)	$\tilde{\theta}_y$ (deg)	$\tilde{\theta}_z$ (deg)
5.8 ±17.8	9.7 ±19.2	10.4 ±9.1	-0.2 ±0.51	-0.07 ±0.50	-0.11 ±1.03

TABLE 7.1: Estimation error compared to Vicon. Reported error is standard deviation.

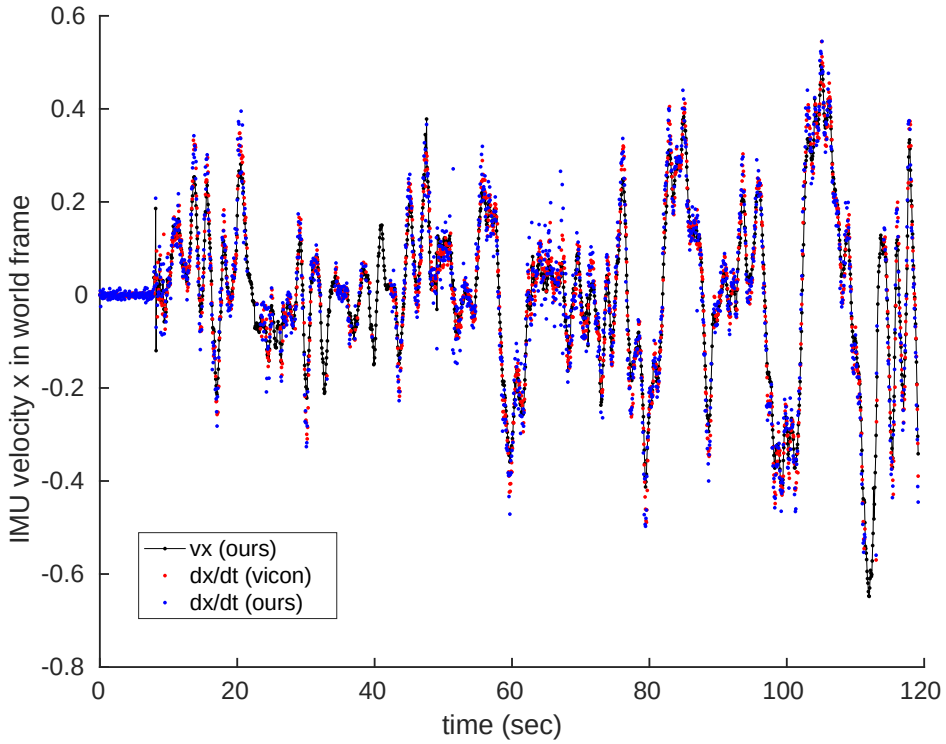


FIGURE 7.6: Estimated velocity in x of the IMU in world coordinates. Our system's velocity state estimate almost coincides with our system's and Vicon's differentiated position estimates.

7.3.3 Reconstruction Of An Underwater Shipwreck

We also tested our system on stereo and IMU data that were collected using a waterproof sensor box handheld by a diver. The camera intrinsics were calibrated underwater, but their fixed transformation was computed on land. The trajectory that the diver followed was able to perform visual coverage of the Helion shipwreck at the coast of Carlisle Bay in Barbados. The coverage was redundant with multiple loops, different viewpoints, and numerous opportunities for loop closure. The reason behind mapping a shipwreck was to test the performance of our system in cases where features are guaranteed to be a few meters away from the camera, in the presence of many moving objects such as fish, and also lack of visibility after 10 meters, which is common in underwater environments and one of the factors that makes them challenging. The dataset includes multiple in-place rotations especially at the bow of the ship and multiple opportunities for losing tracking due to abrupt turns. It also includes transitions from downward-looking coverage to forward-looking. The general shape of the trajectories followed a figure 8, with the intersection point being at the center of the ship. For rough ground truth, we measured the dimensions of the rectangular opening in the hull that dominates the front part of the ship. It was measured underwater to be 4.5m $\times$ 9.7m.

We used this dataset to compare our system with ORB-SLAM1 [Mur-Artal et al., 2015], ORB-SLAM2 [Mur-Artal and Tardós, 2016], and SVO [Forster et al., 2014]. To evaluate the tracking performance, we turned off the relocalisation and loop closing modules, and we ran the four systems 10 times separately on the same single loop at the front of the ship. The results are shown in Figs. 7.7. Of the four algorithms the one that fared the worst was SVO, which did not complete a single loop during these tests, as it got lost every four meters or so. This is shown on Table 7.2. ORB-SLAM2, which was the most robust in terms of not getting lost during any of the 10 rounds, also misestimated the in-place rotation at the bow of the ship and its yaw estimate drifted. Our system performed better than ORB-SLAM1, almost aligning the rectangle at the front of the ship, and estimated the rectangular opening in the hull to be 4.7m $\times$ 10.8m

TABLE 7.2: Number of times the estimator got lost while reconstructing the Helion shipwreck

Ours	ORB-SLAM1	ORB-SLAM2	SVO
2	4	0	10

Relocalisation and loop closure are essential for the life-long operation of this system, and we decided to test it in conjunction with this functionality by visually mapping the entire shipwreck, seen in Fig. 7.8. We validated the end result by comparing against the tape measurements of the front rectangle opening. The final estimates of its dimensions were 4.7m $\times$ 10.9m, and the system was able to process the entire 15-minute trajectory.

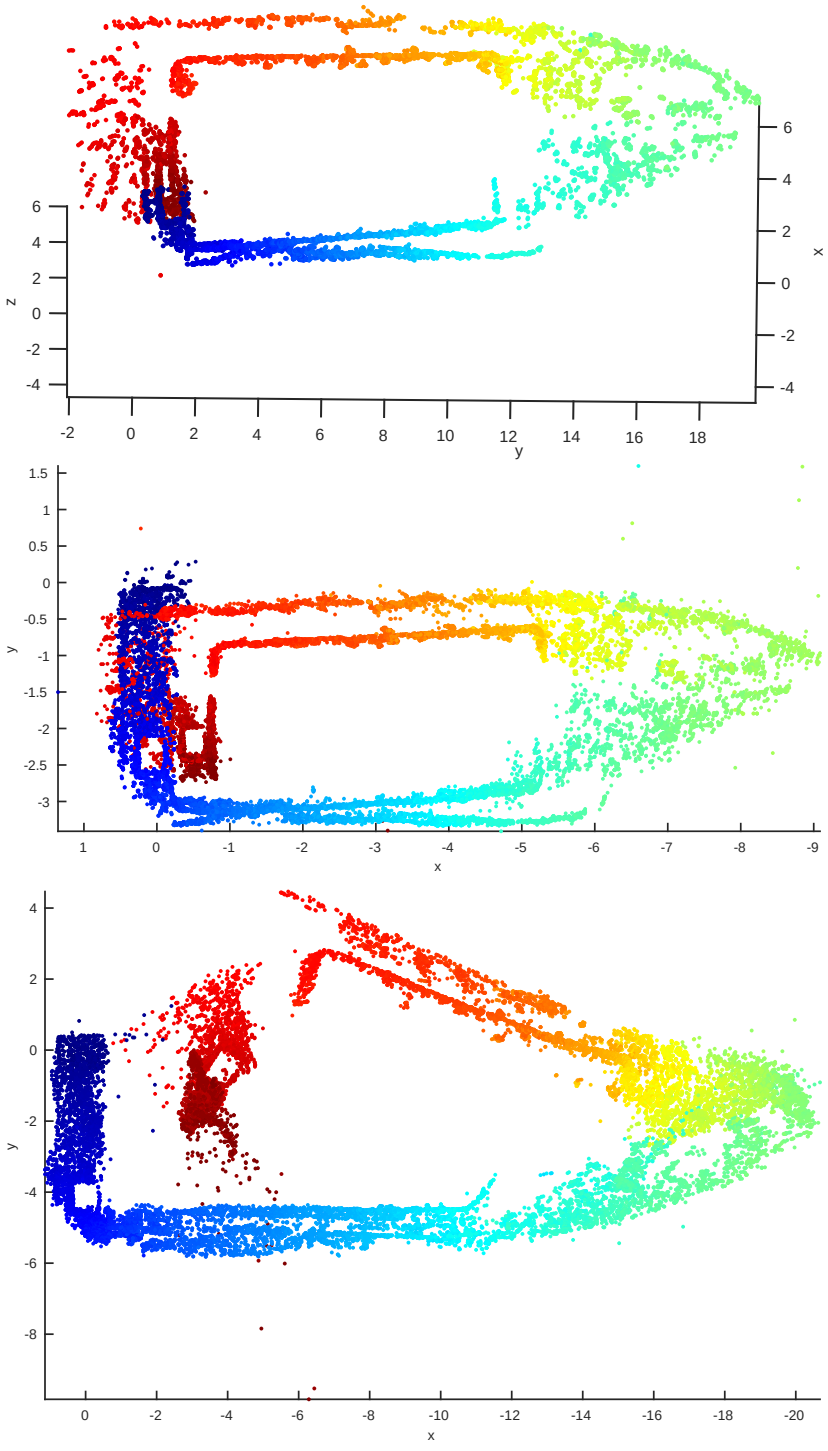


FIGURE 7.7: (a) Our system's map in North-East-Down frame (b) ORB-SLAM1's map in the first camera's frame (c) ORBSLAM2's map in the first camera's frame

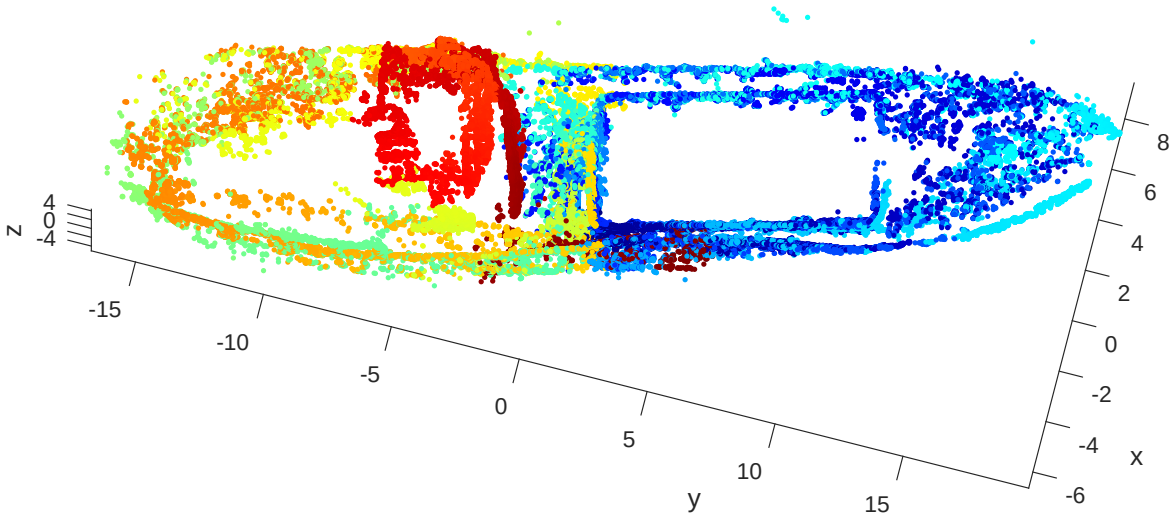


FIGURE 7.8: Map of the Helion shipwreck, estimated by our system with loop closure enabled. Color represents time. Blue points were visited first.

8 FUTURE WORK

This thesis has touched upon multiple aspects of problems related to robot videography based on human preferences and specifications, and how to make the interaction between people who need data from challenging environments and the robots that can get that data more efficient and seamless. That said, there are multiple specific directions that need to be explored in depth:

8.1 Active Semi-Supervised Learning for Visual Rewards

Semi-supervised learning refers to learning from both labeled and unlabeled examples. It has been explored in multiple contexts [Zhu et al., 2003, Zhu, 2006], including deep generative models [Kingma et al., 2014], and it has been shown that learning about the distribution of the data can help inform discriminative models. An implicit assumption in this techniques is that if two data points are "close", according to some definition, then their labels should be "close" as well. To the best of our knowledge this idea has not been examined in depth for structured prediction and semantic segmentation in particular. The main constraint is once again that current methods, such as [Kingma et al., 2014], assume a small number of classes, as is the case for example on MNIST or similar datasets. When the number of possible classes reaches $C^{H \times W}$ then these methods break as they rely on marginalization over all possible classes. Even less explored is the idea of semi-supervised learning that is done in an active way, where the data to be labeled are deliberatively chosen to provide the most information to a learner. Similar research directions are usually found in the *machine teaching* literature [Zhu, 2015].

8.2 Predictive Topological Tracking in 3D

In this thesis we relied on the topology of 2D spaces to compress the possible predictions one can make about future paths and trajectories. The notion of homotopy was sufficient to capture the possible ways of navigating around obstacles. Unfortunately, this is not the case when we go to 3D. For example, imagine a finite-length wall separating A to B, in an otherwise empty space. Paths from A to B in 2D must

choose whether to traverse on the left or the right of the wall, and thus belong to two different classes. In 3D paths can jump over the wall in a continuous way, and therefore the notion of homotopy is not helpful. The notion of *persistent homology* within topological data analysis is promising for this direction [Edelsbrunner and Harer, , Pokorny et al., 2016c, Pokorny et al., 2016a, Pokorny and Kragic, 2015], as a way to describe the shape of a high-dimensional pointcloud. The main idea in this area is that high-dimensional pointclouds can be summarized by low-dimensional descriptors, which are obtained by converting the pointcloud into (loosely speaking) graphs whose neighbourhood relationships are determined by increasing spatial distance[1], leading to a graph description of connectivity at increasing scales. The persistent features in this sequence of graphs determine the persistent homology, from which we can define classes of topologically distinct trajectories. The utility of these concepts for tracking will be if topological information can be combined with generative models of trajectories, so that topologically distinct paths can be sampled efficiently; a direction which at the moment is very much overlooked.

8.3 Combining Intermittent Tracking and Visual Exploration

We demonstrated a system where a robot visually tracked a diver while at the same time directed another camera towards interesting parts of the scene. The long-term vision for this type of system is that the robot should be able to intermittently track the diver, without having to be constantly fixated in their direction. This will allow robots to explore collaboratively alongside humans, without constantly having to see where the human is, but by making informed short-term and long term predictions of their location. After exploring on their own, robots will try to relocate their human collaborators to redirect their attention towards any interesting observations they made in the meantime. In order to achieve this, many of the insights we developed in our work on combining IRL with tracking will apply here.

[1] More formally: filtered simplicial complexes

8.4 Combining Model-Based Visual Attention with Surprise-Based Exploration

In surprise- and curiosity-based exploration the robot is typically drawn to parts of the scene where novel observations can be made [Girdhar and Dudek, 2011, Girdhar and Dudek, 2012, Girdhar and Dudek, 2014a, Paul et al., 2014]. Often however, the user is interested both in surprising observations and in specific types of anticipated observations, and focusing on visual anomalies does not guarantee that the user will find the recorded data useful at the end. We argue that these two types of objectives need to be balanced in a similar way to exploration-exploitation in large-scale MDP planning and reinforcement learning. Our work on active reward learning in this thesis focused exclusively on exploitation, while [Girdhar and Dudek, 2011, Girdhar and Dudek, 2012, Girdhar and Dudek, 2014a, Paul et al., 2014] looked carefully at anomaly detection and exploration. Combining the two objectives in a single visual search and exploration framework is an important direction, where Upper Confidence Bounds for online learning could be a fruitful ingredient [Auer, 2000].

8.5 Interactive, Long-Term Visual Search and Exploration

It is a testament to the fragility, expense of execution, and engineering efforts required, that robotics experiments are typically limited to a few hours in duration and a few hundred meters in spatial cover. Fortunately, this does not encapsulate the entirety of the field, as robotic efforts from large oceanographic institutes span months and kilometers of reach. We argue that as long experiments unfold and data comes in in batches, users' preferences about visual content will dynamically change. It is important to be able to perform Bayesian updates of the visual reward/attention model with as few interactions as possible.

9 CONCLUSIONS

Where should robots look? What type of data should they collect for us? What are the best ways for us to communicate our needs and specifications to them? These are some of the questions that have motivated the work presented in this thesis. We have provided both algorithms and robotic systems that have been demonstrated to work in challenging real-world scenarios, including underwater, on the ground, and in the air, in collaboration with and to the service of human scientists. We have performed some of the most unique and large scale multirobot experiments in the field, and we have contributed several algorithmic advancements to an area (robot videography) that for many justifiable reasons has typically been studied in a compartmentalized way (tracking, human robot interaction, visual search and exploration, field robotics), due its sheer breadth. This thesis has been an attempt to change that, and emphasize the fact that robot videographers are essential for assisting humans to collect the data they need.

This thesis described the following contributions:

- An active supervised learning approach that tries to minimize the model uncertainty of a distribution of pixel-level saliency maps. This method selects batches of the most informative images to present next to the expert user for pixel-level annotation of their regions of interest.
- Robot field trials involving cooperation between aerial, sea surface, and underwater vehicles exploring environments in a semi-autonomous fashion, according to the specifications of a human. These types of multi-robot systems working in concert with humans are unique in the robotics literature and have paved the way for novel environmental monitoring applications.
- A method that combines visual tracking, behavior prediction, planning and control so as to be able to follow a known target with intermittent observations. Unlike many existing trackers, it is able to deal with the case where visual contact with the subject is lost, by making informed predictions about its subject's behavior, which is learned through inverse reinforcement learning.

- A method to topologically constrain the predictions of a pursuit algorithm in order to condense the expanse of possibilities that the planner must consider, which enables target tracking in large environments even in the absence of a behavior model for the target of interest.
- We prove complexity theory results that show the NP-hardness of the pursuit problem even when the follower has speed advantage over the target.
- A localization and mapping method that allows real-time visual navigation based on sensory inputs from a stereo camera and an inertial measurement unit. We have used this method for GPS-free navigation in many of the environments explored and the scenarios examined in the field trials mentioned above. In particular we demonstrate the real-time 3D mapping of an entire shipwreck by a lightweight, handheld sensor pack, which is one of the few deployments of its kind.

We hope that the main messages conveyed through this work:

(A) That visual exploration should actively incorporate a model of the users' preferences, otherwise the data a robot collects through surprise-based navigation is not guaranteed to be useful to them.

(B) That visual tracking should be intermittent and predictive, and combined with exploration and navigation, otherwise robots are not going to be able to robustly assist humans in real environments.

are considered non-controversial and useful to the development of the field. Robotics is destined to become part of our lives. The ways and the extent to which it can ultimately help our fellow humans in their daily work depends on what collectively we adopt as research priorities. We hope that this thesis inspires much needed further work in these two directions.

A Worst Case Computational Complexity of Model-Free Pursuit

We first examine the complexity of computing the minimum speed advantage the pursuer needs to have in order to clear the environment and thus find the evader. To this end we assume a discretized model of the environment, which we represent by a graph $G = (V, E)$. We also discretize the time of the motion and we assume that at the end of each time step both the pursuer and the evader will reside on a node of the graph. In other words, we consider node search, where edges signify transitions between nodes, without making it possible for the agents to reside on an edge at the end of a time step. As such, the evader can traverse one edge in one time step, while the pursuer can traverse $s_p \in \{1, ..., 2|E|\}$ edges [1], which is going to denote his speed. Under these assumptions we let S_n be a star-shaped tree on n nodes.

Definition 1. *Let $CLEAR(S_n, s_p, t, v_0)$ be the decision problem that returns 'yes' iff S_n can be cleared by a single pursuer with initial position v_0 on the tree and speed s_p in time $t \in \mathbb{Q} \cap [0, \infty]$.*

In order to classify the complexity of CLEAR and other related problems, we are going to need the following problem:

Definition 2. *Let $3PARTITION(x_1, x_2, ..., x_{3n})$ be the decision problem that returns 'yes' iff the positive integers $x_1, x_2, ..., x_{3n}$, whose sum is nB where $B \in \mathbb{N}$, can be partitioned into triples that have the same sum, B. 3PARTITION has been shown to be strongly NP-complete [Garey and Johnson, 1979], in the sense that it remains NP-complete even if the input numbers $x_1, x_2, ..., x_{3n}$ are bounded by a polynomial in n.*

Theorem 1. *$CLEAR(S_n, s_p, t, v_0)$ is NP-complete.*

Proof. We are going to construct a polynomial-time Turing reduction from 3PARTITION to CLEAR. Given a set of positive integers $x_1, x_2, ..., x_n$ we construct a star-shaped tree with $3n$ branches, as shown in Fig. A.1. The i^{th} branch contains x_i nodes, including the root. We claim that a 3-partition exists iff this tree can be cleared by a

[1] $2|E|$ allows a tour of the graph in one time step

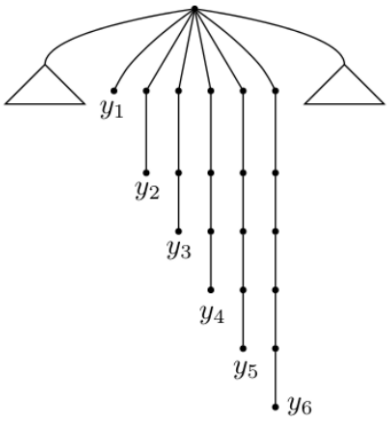


FIGURE A.1: An example of a tree construction used in the reductions. In this example, $(y_1, y_2, y_3) = (2, 3, 4)$ and $(y_4, y_5, y_6) = (5, 6, 7)$ are two triples of positive integers, some of the $3n$ numbers that are the input of a 3PARTITION instance

single pursuer, having speed $s_p = 2(B-3)$, in time $t = n - \max\{x_i - 1\}/(2(B-3))$, starting from the root.

($\Rightarrow$) Assume a 3-partition exists, and consider an arbitrary triple. Its sum is $B = \sum_{i=1}^{3n} x_i/n$. A pursuer with speed $s_p = 2(B-3)$ can start from the root, clear the three branches corresponding to the triple and return to the root in one time unit, while the evader has crossed only one edge. The pursuer can thus clear the remaining $n-1$ triples in time $n-1$. Since the last branch of the last triple does not require going back to the root, we should only take it into account once. To minimize the time it takes to clear the tree the pursuer must leave the longest branch for last. So, the tree can be cleared in time $t = n - \max\{x_i - 1\}/(2(B-3))$.

($\Leftarrow$) Assume a 3-partition does not exist. Then for any partition of $x_1, x_2, ..., x_{3n}$ into triples, we will be able to find two sets $\{y_1, y_2, y_3\}$ and $\{y_4, y_5, y_6\}$ such that without loss of generality: $y_1 + y_2 + y_3 < B$, $y_4 + y_5 + y_6 > B$, none of the y_i is $\max\{x_i\}$, and no repartition of these six numbers into two triples would make both sums B. We want to show that a pursuer with speed $s_p = 2(B-3)$ cannot clear these two branches in time $t <= 2$, while being able to clear the remaining $n-2$ branches in time $n - 2 - \max\{x_i - 1\}/(2(B-3))$. Suppose the pursuer decides to clear y_1, y_2, y_3 first, which will require a portion of the the first time unit. If he spends the remaining of that portion at the root, to avoid possible recontamination by the evader, then he will need more than one time unit to clear y_4, y_5, y_6, so we are done. If he decides to spend the remaining portion of the first unit clearing part of y_4, y_5, y_6, even if he arrives at the root at the end of the second time unit, y_1, y_2, y_3 or some of the other triples will have been recontaminated. So, additional time will be required to re-clear them.

This shows that CLEAR is NP-hard. The trajectory of the pursuer on the tree is a polynomial-time-verifiable certificate for CLEAR(S_n, s_p, t, v_0) which makes it NP-complete. □

It is interesting to compare this result with the problem of computing the minimum number of pursuers, each with speed equal to or less than that of the evader, required to clear a graph. Even though that problem was shown to be NP-complete for general graphs, it is tractable for the case of trees, where a linear-time algorithm for computing the search number is available [Parsons, 1978, Megiddo et al., 1988, Borie et al., 2009]. In our variant of the problem, even the case of star-shaped trees is computationally hard for a single pursuer. This is due to the size of the configuration space of the game, which for the case of star-shaped trees can be denoted by $(c_1, c_2, ..., c_b, p)$. c_i is the number of nodes that are currently cleared on the i^{th} branch, and p is the current node at which the pursuer resides. $c_i \in \Theta(n)$ and $p \in \Theta(n)$ so the size of the configuration space is $\Theta(n^{b+1})$. If the number of branches is also $\Theta(n)$ then the size of the configuration space becomes non-polynomial. The goal of the pursuer is to find a simple path from $(0, 0, ..., 0, \text{root})$ to any of the following configurations $(\max c_1, \max c_2, ..., \max c_b, *)$ in this state space. The edges linking these configurations depend on the speed advantage s_p that we allow the pursuer, which can range from 1 to $2(n-1)$. The more speed we allow, the denser the connectivity of the state space. Therefore, given a certain pursuer speed, a clearing strategy is possible when there is a path connecting the starting state to one of the goal states. When $s_p = n - 1$, for example, there are direct edges connecting them, so the pursuer can clear the star in one time unit. If we restrict the number of branches b to be fixed, the size of the state space becomes polynomial and typical graph-searching algorithms can give us a clearing strategy.

Other variants of CLEAR are also computationally hard, even for the case of stars. For instance, computing the minimum time in which a star is clearable by a pursuer with speed s_p, or computing the minimum speed at which a star is clearable at a given time t. This is shown in the following theorems.

Definition 3. *Let MINTIME-CLEAR$(S_n, s_p, v_0) = t^*$ be the optimization problem of computing the minimum time t^* at which a star-shaped tree S_n is clearable by a single pursuer with speed s_p who starts at node v_0.*

Definition 4. *Let MINSPEED-CLEAR$(S_n, t, v_0) = s_p^*$ be the optimization problem of computing the minimum speed s_p^* with which a single pursuer can clear a star-shaped tree S_n in time t, starting at node v_0.*

Definition 5. *Let RANGE-CLEAR$(S_n, s_p^{max}, t^{max}, v_0)$ be the decision problem that returns 'yes' iff the star S_n can be cleared within time t^{max} by a pursuer who has speed at most s_p^{max}.*

Theorem 2. *MINTIME-CLEAR*$(S_n, s_p, v_0) = t^*$ *and MINSPEED-CLEAR*$(S_n, t, v_0) = s_p^*$ *are NP-hard.*

Proof. The exact same reduction from 3PARTITION as presented in Theorem 1 can be used in this case too. We observe that in that construction $t^* = n - \max\{x_i - 1\}/(2(B-3))$ is the minimum time at which the star can be cleared given the pursuer speed $2(B-3)$, and conversely, the minimum speed at which the star is clearable in the given time $n - \max\{x_i - 1\}/(2(B-3))$ is $s_p^* = 2(B-3)$. □

Theorem 3. *RANGE-CLEAR*$(S_n, s_p^{max}, t^{max}, v_0)$ *is NP-complete.*

Proof. Via a reduction from the decision version of MINSPEED-CLEAR$(S_n, t, v_0) = s_p^*$, which is NP-complete. To answer that decision problem we can apply RANGE-CLEAR$(S_n, s, t, \text{root}(S_n))$ for all possible speeds s. The same polynomial-time-verifiable certificate that was used in Theorem 1 is valid here, too. □

Definition 6. *Let MINSPEED-CLEAR*$(S_n, v_0) = s_p^*$ *be the optimization problem of computing the minimum speed* s_p^* *with which a single pursuer can clear a star-shaped tree* S_n*, starting at node* v_0*.*

Theorem 4. *MINSPEED-CLEAR*$(S_n, v_0) = s_p^*$ *is NP-hard.*

Proof. The reduction presented in Theorem 1 shows that a 3-partition exists iff the star can be cleared with a certain speed at a certain time. It does not guarantee that $2(B-3)$ is the minimum speed at which the star can be cleared. For example, a speed of $2\max x_i$ can also guarantee clearing of the star. We want to modify that reduction so that $s_p = 2(B-3)$ is indeed the minimum speed at which the modified star can be cleared.

Consider the following modification to the star presented in the proof of Theorem 1: we add s_p^2 new branches to that star, each containing $B-2$ nodes, including the root. A pursuer with speed $s_p = 2(B-3)$ can clear each of these new branches, starting and ending at the root, in exactly one time unit. The old portion of the tree can also be cleared with that speed.

We claim that the new star cannot be cleared with speed less than s_p. This is because the first attempt to clear one of the branches of length $B-2$ would allow recontamination of the root. At the next time step $3n + s_p^2 - 1$ branches will have an a recontaminated edge. Re-clearing these edges will take time at least twice the number of those edges, by which time the branch we started with is going to be completely recontaminated. □

Corollary 1. *The complexity of all the above problems remains true when stars are replaced with trees, and general graphs.*

Bibliography

[Aghasadeghi and Bretl, 2011] Aghasadeghi, N. and Bretl, T. (2011). Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1561–1566.

[Ahuja and Orlin, 2001] Ahuja, R. K. and Orlin, J. B. (2001). Inverse optimization. *Operations Research*, 49(5):771–783.

[Amin and Singh, 2016] Amin, K. and Singh, S. P. (2016). Towards resolving unidentifiability in inverse reinforcement learning. *CoRR*, abs/1601.06569.

[Amodei et al., 2016] Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., and Mané, D. (2016). Concrete problems in AI safety. *CoRR*, abs/1606.06565.

[Aström, 1965] Aström, K. J. (1965). Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205.

[Auer, 2000] Auer, P. (2000). Using upper confidence bounds for online learning. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 270–279.

[Avanzini et al., 2013] Avanzini, P., Royer, E., Thuilot, B., and Derutin, J. P. (2013). Using monocular visual SLAM to manually convoy a fleet of automatic urban vehicles. In *IEEE International Conference on Robotics and Automation*, pages 3219–3224.

[Bacchus et al., 1996] Bacchus, F., Boutilier, C., and Grove, A. (1996). Rewarding behaviors. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1160–1167. AAAI Press.

[Bacchus et al., 1997] Bacchus, F., Boutilier, C., and Grove, A. (1997). Structured solution methods for non-markovian decision processes. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, pages 112–117. AAAI Press.

[Badrinarayanan et al., 2015] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561.

[Baker et al., 2009] Baker, C. L., Saxe, R., and Tenenbaum, J. B. (2009). Action understanding as inverse planning. *Cognition*, 113(3):329 – 349.

[Balcan et al., 2007] Balcan, M.-F., Broder, A., and Zhang, T. (2007). Margin based active learning. In *Proceedings of the 20th Annual Conference on Learning Theory*, COLT'07, pages 35–50, Berlin, Heidelberg. Springer-Verlag.

[Basu et al., 2018] Basu, C., Singhal, M., and Dragan, A. D. (2018). Learning from richer human guidance: Augmenting comparison-based learning with feature queries. In *ACM/IEEE International Conference on Human-Robot Interaction, HRI, Chicago, USA*, pages 132–140.

[Bellman, 1957] Bellman, R. (1957). A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684.

[Bertsekas and Tsitsiklis, 1996] Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, 1st edition.

[Bhattacharya and Hutchinson, 2009] Bhattacharya, S. and Hutchinson, S. (2009). On the Existence of Nash Equilibrium for a Two-player Pursuit–Evasion Game with Visibility Constraints. *The International Journal of Robotics Research*, 29(7):831–839.

[Bhattacharya et al., 2010a] Bhattacharya, S., Kumar, V., and Likhachev, M. (2010a). Search-based path planning with homotopy class constraints. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 1230–1237. AAAI Press.

[Bhattacharya et al., 2010b] Bhattacharya, S., Kumar, V., and Likhachev, M. (2010b). Search-Based Path Planning with Homotopy Class Constraints. In *AAAI*, pages 1230–1237.

[Bhattacharya et al., 2011a] Bhattacharya, S., Likhachev, M., and Kumar, V. (2011a). Identification and representation of homotopy classes of trajectories for search-based path planning in 3d. In *Proceedings of the Robotics: Science and Systems Conference (RSS 2011), (Best Paper Award)*.

[Bhattacharya et al., 2011b] Bhattacharya, S., Likhachev, M., and Kumar, V. (2011b). Identification and Representation of Homotopy Classes of Trajectories for Search-based Path Planning in 3D. In *Robotics: Science and Systems (RSS)*.

[Blum et al., 2003] Blum, A., Jackson, J. C., Sandholm, T., and Zinkevich, M. (2003). Preference elicitation and query learning. In Schölkopf, B. and Warmuth, M. K., editors, *Learning Theory and Kernel Machines*, pages 13–25.

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1613–1622. JMLR.org.

[Bolme et al., 2010] Bolme, D. S., Beveridge, J. R., Draper, B. A., and Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550.

[Borie et al., 2009] Borie, R., Tovey, C., and Koenig, S. (2009). Algorithms and complexity results for pursuit-evasion problems. In *International Joint Conference on Artifical Intelligence*, pages 59–66. Morgan Kaufmann Publishers Inc.

[Boularias et al., 2011] Boularias, A., Kober, J., and Peters, J. (2011). Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 182–189.

[Boutilier, 2002] Boutilier, C. (2002). A pomdp formulation of preference elicitation problems. In *Eighteenth National Conference on Artificial Intelligence*, pages 239–246.

[Braziunas, 2006] Braziunas, D. (2006). Computational approaches to preference elicitation. Technical report, University of Toronto.

[Braziunas and Boutilier, 2010] Braziunas, D. and Boutilier, C. (2010). Assessing regret-based preference elicitation with the utpref recommendation system. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, pages 219–228.

[Breiman, 1996] Breiman, L. (1996). Bagging Predictors. *Journal of Machine Learning*, 24(2):123–140.

[Brown and Niekum, 2017] Brown, D. S. and Niekum, S. (2017). Efficient probabilistic performance bounds for inverse reinforcement learning. *CoRR*, abs/1707.00724.

[Bruce and Tsotsos, 2008] Bruce, N. D. and Tsotsos, J. K. (2008). Attention in cognitive systems. theories and systems from an interdisciplinary viewpoint. chapter An Information Theoretic Model of Saliency and Visual Search, pages 171–183. Springer-Verlag, Berlin, Heidelberg.

[Bruce et al., 2015] Bruce, N. D., Wloka, C., Frosst, N., Rahman, S., and Tsotsos, J. K. (2015). On computational modeling of visual saliency: Examining what's right, and what's left. *Vision Research*, 116:95 – 112. Computational Models of Visual Attention.

[Burgard et al., 2000] Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S. (2000). Collaborative multi-robot exploration.

[Cehovin et al., 2015] Cehovin, L., Leonardis, A., and Kristan, M. (2015). Visual object tracking performance measures revisited. *CoRR*, abs/1502.05803.

[Chandrasekhar et al., 2006] Chandrasekhar, V., Seah, W. K., Choo, Y. S., and Ee, H. V. (2006). Localization in underwater sensor networks: survey and challenges. In *1st ACM international workshop on Underwater networks*, pages 33–40.

[Chen et al.,] Chen, M., Frazzoli, E., Hsu, D., and Lee, W. S. POMDP-lite for Robust Robot Planning under Uncertainty, journal = CoRR, volume = abs/1602.04875, year = 2016, url = http://arxiv.org/abs/1602.04875,.

[Chen and Ziebart, 2015] Chen, X. and Ziebart, B. (2015). Predictive Inverse Optimal Control for Linear-Quadratic-Gaussian Systems. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 165–173, San Diego, California, USA. PMLR.

[Choi and eung Kim, 2012] Choi, J. and eung Kim, K. (2012). Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In *Advances in Neural Information Processing Systems 25*, pages 305–313.

[Choi and Kim, 2011] Choi, J. and Kim, K.-E. (2011). Inverse reinforcement learning in partially observable environments. *Journal of Machine Learning Research*, 12:691–730.

[Choset, 1997] Choset, H. (1997). Incremental construction of the generalized voronoi diagram, the generalized voronoi graph, and the hierarchical generalized voronoi graph. In *CGC Workshop on Computational Geometry*.

[Choset and Burdick, 2000] Choset, H. and Burdick, J. (2000). Sensor-Based Exploration: The Hierarchical Generalized Voronoi Graph. *The International Journal of Robotics Research*, 19(2):96–125.

[Christian Forster, 2015] Christian Forster, Luca Carlone, F. D. D. S. (2015). Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics Science and Systems (RSS)*.

[Christiano, 2018] Christiano, P. (2018). AI alignment blog. https://ai-alignment.com/. Accessed: 2018-3-19.

[Cohn et al., 1996] Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *J. Artif. Int. Res.*, 4(1):129–145.

[Comaniciu et al., 2003] Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575.

[Corke et al., 2007] Corke, P., Detweiler, C., Dunbabin, M., Hamilton, M., Rus, D., and Vasilescu, I. (2007). Experiments with underwater robot localization and tracking. In *IEEE International Conference on Robotics and Automation*, pages 4556–4561.

[Dantam and Stilman, 2012] Dantam, N. and Stilman, M. (2012). *The Motion Grammar: Linguistic Perception, Planning, and Control*, pages 49–56.

[Doerr et al., 2015] Doerr, A., Ratliff, N., Bohg, J., Toussaint, M., and Schaal, S. (2015). Direct loss minimization inverse optimal control. In *Proceedings of Robotics: Science and Systems*, Rome, Italy.

[Doherty et al., 2018] Doherty, K., Flaspohler, G., Roy, N., and Girdhar, Y. (2018). Approximate distributed spatiotemporal topic models for multi-robot terrain characterization. In *Intelligent Robots and Systems (IROS)*.

[Doshi-Velez and Konidaris, 2016] Doshi-Velez, F. and Konidaris, G. (2016). Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *International Joint Conference on Artificial Intelligence*, pages 1432–1440. AAAI Press.

[Dudek et al., 1995] Dudek, G., Jenkin, M., Milios, E., and Wilkes, D. (1995). Experiments in sensing and communication for robot convoy navigation. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 268–273.

[Edelsbrunner and Harer,] Edelsbrunner, H. and Harer, J. Persistent homology – a survey.

[Edelsbrunner and Harer, 2010] Edelsbrunner, H. and Harer, J. (2010). *Computational topology : an introduction*. Providence, Rhode Island : American Mathematical Society.

[Efrat et al., 2006] Efrat, A., Kobourov, S. G., and Lubiw, A. (2006). Computing homotopic shortest paths efficiently. *Computational Geometry Theory and Applications*, 35(3):162–172.

[Efron and Stein, 1981] Efron, B. and Stein, C. (1981). The jackknife estimate of variance. *Annals of Statistics*, 9(3):586–596.

[Emerson, 1990] Emerson, E. A. (1990). Handbook of theoretical computer science (vol. b). chapter Temporal and Modal Logic, pages 995–1072. MIT Press, Cambridge, MA, USA.

[Englert et al., 2017] Englert, P., Vien, N. A., and Toussaint, M. (2017). Inverse kkt — learning cost functions of manipulation tasks from demonstrations. *International Journal of Robotics Research*, 36(13-14):1474–1488.

[Erol et al., 2007] Erol, M., Vieira, L. F. M., and Gerla, M. (2007). AUV-aided localization for underwater sensor networks. In *IEEE International Conference on Wireless Algorithms, Systems and Applications*, pages 44–54.

[Evans et al., 2016] Evans, O., Stuhlmüller, A., and Goodman, N. D. (2016). Learning the preferences of ignorant, inconsistent agents. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 323–329.

[Evans et al., 2017] Evans, O., Stuhlmüller, A., Salvatier, J., and Filan, D. (2017). Modeling Agents with Probabilistic Programs. `http://agentmodels.org`. Accessed: 2018-3-19.

[Eysenbach et al., 2018] Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. (2018). Diversity is all you need: Learning skills without a reward function. *CoRR*, abs/1802.06070.

[Finn et al., 2016a] Finn, C., Christiano, P. F., Abbeel, P., and Levine, S. (2016a). A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *CoRR*, abs/1611.03852.

[Finn et al., 2016b] Finn, C., Levine, S., and Abbeel, P. (2016b). Guided cost learning: Deep inverse optimal control via policy optimization. *CoRR*, abs/1603.00448.

[Forster et al., 2014] Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). In *Proc. IEEE Intl. Conf. on Robotics ...*.

[Fries and Wuensche, 2014] Fries, C. and Wuensche, H.-J. (2014). Monocular template-based vehicle tracking for autonomous convoy driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2727–2732.

[Frintrop et al., 2010] Frintrop, S., Rome, E., and Christensen, H. I. (2010). Computational visual attention systems and their cognitive foundations: A survey. *ACM Transactions Applied Perception*, 7(1).

[Furgale et al., 2012] Furgale, P., Barfoot, T., and Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2088–2095.

[Furnkranz and Hullermeier, 2010] Furnkranz, J. and Hullermeier, E. (2010). *Preference Learning*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition.

[Gal and Ghahramani, 2016a] Gal, Y. and Ghahramani, Z. (2016a). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA. PMLR.

[Gal and Ghahramani, 2016b] Gal, Y. and Ghahramani, Z. (2016b). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, pages 1050–1059.

[Gal et al., 2017] Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image data. *CoRR*, abs/1703.02910.

[Galvez-López and Tardos, 2012] Galvez-López, D. and Tardos, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197.

[Gao and Vasconcelos, 2005] Gao, D. and Vasconcelos, N. (2005). Discriminant saliency for visual recognition from cluttered scenes. In *Advances in Neural Information Processing Systems 17*, pages 481–488. MIT Press.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

[Geifman and El-Yaniv, 2017] Geifman, Y. and El-Yaniv, R. (2017). Deep active learning over the long tail. *CoRR*, abs/1711.00941.

[Giesbrecht et al., 2009] Giesbrecht, J. L., Goi, H. K., Barfoot, T. D., and Francis, B. A. (2009). A vision-based robotic follower vehicle. In *SPIE Defense, Security, and Sensing*, page 73321O. International Society for Optics and Photonics.

[Giguere et al., 2006] Giguere, P., Dudek, G., and Prahacs, C. (2006). Characterization and modeling of rotational responses for an oscillating foil underwater robot. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3000–3005.

[Girdhar and Dudek, 2011] Girdhar, Y. and Dudek, G. (2011). *A surprising problem in navigation*, chapter 11, pages 228–252.

[Girdhar and Dudek, 2012] Girdhar, Y. and Dudek, G. (2012). Efficient on-line data summarization using extremum summaries. In *2012 IEEE International Conference on Robotics and Automation*, pages 3490–3496. IEEE.

[Girdhar and Dudek, 2014a] Girdhar, Y. and Dudek, G. (2014a). Exploring underwater environments with curiosity. In *Canadian Conference on Computer and Robot Vision*, pages 104–110. IEEE.

[Girdhar and Dudek, 2014b] Girdhar, Y. and Dudek, G. (2014b). Exploring underwater environments with curiosity. In *Computer and Robot Vision (CRV), 2014 Canadian Conference on*, pages 104–110. IEEE.

[Girdhar et al., 2011] Girdhar, Y., Xu, A., Dey, B. B., Meghjani, M., Shkurti, F., Rekleitis, I., and Dudek, G. (2011). MARE: Marine Autonomous Robotic Explorer. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 5048–5053, San Francisco, USA.

[Gottlieb et al., 2013] Gottlieb, J., Oudeyer, P.-Y., Lopes, M., and Baranes, A. (2013). Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in Cognitive Sciences*, 17(11):585 – 593.

[Graves, 2012] Graves, A. (2012). Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 5–13. Springer.

[Guestrin et al., 2011] Guestrin, C., Koller, D., Parr, R., and Venkataraman, S. (2011). Efficient solution algorithms for factored mdps. abs/1106.1822.

[Hadfield-Menell et al., 2016] Hadfield-Menell, D., Dragan, A. D., Abbeel, P., and Russell, S. J. (2016). Cooperative inverse reinforcement learning. *CoRR*, abs/1606.03137.

[Hadfield-Menell et al., 2017] Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S. J., and Dragan, A. (2017). Inverse reward design. In *Advances in Neural Information Processing Systems 30*, pages 6765–6774.

[Harmat et al., 2012] Harmat, A., Sharf, I., and Trentini, M. (2012). Parallel tracking and mapping with multiple cameras on an unmanned aerial vehicle. *Intelligent Robotics and Applications*, pages 421–432.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151.

[Hatcher, 2000] Hatcher, A. (2000). *Algebraic topology*. Cambridge Univ. Press, Cambridge.

[Heess et al., 2015] Heess, N., Hunt, J. J., Lillicrap, T. P., and Silver, D. (2015). Memory-based control with recurrent neural networks. *CoRR*, abs/1512.04455.

[Hesch et al., 2014] Hesch, J. A., Kottas, D. G., Bowman, S. L., and Roumeliotis, S. I. (2014). Camera-imu-based localization: Observability analysis and consistency improvement. *International Journal of Robotics Research*, 33(1):182–201.

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

[Ho and Ermon, 2016] Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. *CoRR*, abs/1606.03476.

[Hodgson and Liebeler, 2002] Hodgson, G. and Liebeler, J. (2002). *The Global Coral Reef Crisis: Trends and Solutions*. Reef Check Foundation.

[Houlsby et al., 2011] Houlsby, N., Huszar, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *CoRR*, abs/1112.5745.

[Houthooft et al., 2016] Houthooft, R., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P. (2016). Curiosity-driven exploration in deep reinforcement learning via bayesian neural networks. *CoRR*, abs/1605.09674.

[Hsu et al., 2008] Hsu, D., Lee, W. S., and Rong, N. (2008). A point-based POMDP planner for target tracking. In *IEEE International Conference on Robotics and Automation*, pages 2644–2650.

[Iandola et al., 2016] Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., and Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *CoRR*, abs/1602.07360.

[Igl et al., 2018] Igl, M., Zintgraf, L., Le, T. A., Wood, F., and Whiteson, S. (2018). Deep variational reinforcement learning for POMDPs. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 2117–2126. PMLR.

[Isaacs, 1969] Isaacs, R. (1969). Differential games: Their scope, nature, and future. *Journal of Optimization Theory and Applications*, 3(5):283–295.

[Islam and Sattar, 2017] Islam, M. J. and Sattar, J. (2017). Mixed-domain biological motion tracking for underwater human-robot interaction. In *IEEE International Conference on Robotics and Automation*.

[Isler et al., 2005] Isler, V., Kannan, S., and Khanna, S. (2005). Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 21(5):875–884.

[Jones and Soatto, 2011] Jones, E. S. and Soatto, S. (2011). Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research*, 30(4):407–430.

[Kalal et al., 2012] Kalal, Z., Mikolajczyk, K., and Matas, J. (2012). Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422.

[Kalman, 1964] Kalman, R. (1964). When Is a Linear Control System Optimal? In *Journal of Basic Engineering*.

[Karasev et al., 2016] Karasev, V., Ayvaci, A., Heisele, B., and Soatto, S. (2016). Intent-aware long-term prediction of pedestrian motion. *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

[Kingma et al., 2014] Kingma, D. P., Mohamed, S., Jimenez Rezende, D., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems 27*, pages 3581–3589.

[Kitani et al., 2012] Kitani, K. M., Ziebart, B. D., Bagnell, J. A., and Hebert, M. (2012). *Activity Forecasting*, pages 201–214. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Klein and Murray, 2007] Klein, G. and Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10.

[Kocsis and Szepesvári, 2006] Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293.

[Kocsis and Szepesvári, 2006] Kocsis, L. and Szepesvári, C. (2006). *Bandit Based Monte-Carlo Planning*, pages 282–293. Springer Berlin Heidelberg.

[Krishnan et al., 2016] Krishnan, S., Garg, A., Liaw, R., Thananjeyan, B., Miller, L., Pokorny, F. T., and Goldberg, K. (2016). Swirl: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. In *WAFR*.

[Krishnan et al., 2017] Krishnan, S., Garg, A., Patil, S., Lea, C., Hager, G., Abbeel, P., and Goldberg, K. (2017). Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning. *The International Journal of Robotics Research*, 36(13-14):1595–1618.

[Kümmerle et al., 2011] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g2o: A General Framework for Graph Optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China.

[Lakshminarayanan et al., 2017] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc.

[LaValle, 2011] LaValle, S. M. (2011). Motion planning. *IEEE Robotics Automation Magazine*, 18(1):79–89.

[Lavalle et al., 1997] Lavalle, S. M., Gonzalez-Banos, H., Becker, C., and Latombe, J.-C. (1997). Motion Strategies for Maintaining Visibility of a Moving Target. In *IEEE ICRA*, pages 731–736.

[LeCun et al., 2006] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006). A tutorial on energy-based learning. MIT Press.

[Leutenegger et al., 2015] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual–inertial odometry using nonlinear optimization. *IJRR*, 34(3):314–334.

[Levine and Koltun, 2012] Levine, S. and Koltun, V. (2012). Continuous inverse optimal control with locally optimal examples. In *ICML '12: Proceedings of the 29th International Conference on Machine Learning*.

[Levine et al., 2011] Levine, S., Popovic, Z., and Koltun, V. (2011). Nonlinear inverse reinforcement learning with gaussian processes. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 19–27.

[Lewis and Gale, 1994] Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 3–12, New York, NY, USA. Springer-Verlag New York, Inc.

[Lipton et al., 2016] Lipton, Z. C., Gao, J., Li, L., Li, X., Ahmed, F., and Deng, L. (2016). Efficient exploration for dialog policy learning with deep BBQ networks \& replay buffer spiking. *CoRR*, abs/1608.05081.

[Lopes et al., 2009] Lopes, M., Melo, F., and Montesano, L. (2009). Active learning for reward estimation in inverse reinforcement learning. In Buntine, W., Grobelnik, M., Mladenić, D., and Shawe-Taylor, J., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 31–46.

[Lugo et al., 2013] Lugo, J. J., Masselli, A., and Zell, A. (2013). Following a quadrotor with another quadrotor using onboard vision. In *IEEE European Conference on Mobile Robots (ECMR)*, pages 26–31.

[Luo et al., 2013] Luo, W., Schwing, A., and Urtasun, R. (2013). Latent structured active learning. In *Advances in Neural Information Processing Systems 26*, pages 728–736. Curran Associates, Inc.

[Macgregor and Ormerod, 1996] Macgregor, J. N. and Ormerod, T. (1996). Human performance on the traveling salesman problem. *Perception & Psychophysics*, 58(4):527–539.

[MacKay, 1992] MacKay, D. J. C. (1992). Information-based objective functions for active data selection. *Neural Comput.*, 4(4):590–604.

[Maire, 2007] Maire, F. (2007). Vision based anti-collision system for rail track maintenance vehicles. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 170–175.

[Manderson et al., 2016] Manderson, T., Shkurti, F., and Dudek, G. (2016). Texture-aware slam using stereo imagery and inertial information. In *Conference on Computer and Robot Vision (CRV)*, pages 465–463. IEEE Computer Society.

[Manz et al., 2011] Manz, M., Luettel, T., von Hundelshausen, F., and Wuensche, H. J. (2011). Monocular model-based 3D vehicle tracking for autonomous vehicles in unstructured environment. In *IEEE International Conference on Robotics and Automation*, pages 2465–2471.

[Martinelli, 2013] Martinelli, A. (2013). Visual-inertial structure from motion: Observability and resolvability. pages 4235–4242.

[Meger et al., 2015] Meger, D., Higuera, J. C. G., Xu, A., Giguere, P., and Dudek, G. (2015). Learning legged swimming gaits from experience. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2332–2338. IEEE.

[Meger et al., 2014] Meger, D., Shkurti, F., Poza, D. C., Giguère, P., and Dudek, G. (2014). 3d trajectory synthesis and control for a legged swimming robot. In *IEEE International Conference on Robotics and Intelligent Systems*.

[Megiddo et al., 1988] Megiddo, N., Hakimi, S. L., Garey, M. R., Johnson, D. S., and Papadimitriou, C. H. (1988). The complexity of searching a graph. *J. ACM*, 35(1):18–44.

[Metelli et al., 2017] Metelli, A. M., Pirotta, M., and Restelli, M. (2017). Compatible reward inverse reinforcement learning. In *Advances in Neural Information Processing Systems 30*, pages 2050–2059.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. (2013). Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

[Mourikis and Roumeliotis, 2007] Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint Kalman filter for vision-aided inertial navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3565–3572, Rome, Italy.

[Mulhall, 2007] Mulhall, M. (2007). Saving the rainforests of the sea: An analysis of international efforts to conserve coral reefs. *Duke Environmental Law and Policy Forum*, 19:321–351.

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). ORB-SLAM: a Versatile and Accurate Monocular SLAM System. page 15.

[Mur-Artal and Tardós, 2016] Mur-Artal, R. and Tardós, J. D. (2016). ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *CoRR*, abs/1610.06475.

[Murrieta-Cid et al., 2005] Murrieta-Cid, R., Tovar, B., and Hutchinson, S. (2005). A Sampling-Based Motion Planning Approach to Maintain Visibility of Unpredictable Targets. *Autonomous Robots*, 19(3):285–300.

[Mussmann and Liang, 2018] Mussmann, S. and Liang, P. (2018). On the relationship between data efficiency and error for uncertainty sampling. *CoRR*, abs/1806.06123.

[Nam and Han, 2016] Nam, H. and Han, B. (2016). Learning multi-domain convolutional neural networks for visual tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Narayanan et al., 2013] Narayanan, V., Vernaza, P., Likhachev, M., and LaValle, S. M. (2013). Planning under topological constraints using beam-graphs. In *IEEE International Conference on Robotics and Automation*, pages 431–437.

[Ng and Jordan, 2000] Ng, A. Y. and Jordan, M. (2000). PEGASUS: A Policy Search Method for Large MDPs and POMDPs. In *Proceedings of 16th Conference on Uncertainty in Artificial Intelligence*, pages 406–415.

[Ng and Russell, 2000] Ng, A. Y. and Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, ICML '00, pages 663–670.

[Niekum et al., 2013] Niekum, S., Chitta, S., Barto, A., Marthi, B., and Osentoski, S. (2013). Incremental semantically grounded learning from demonstration. Berlin, Germany.

[Niekum et al., 2015] Niekum, S., Osentoski, S., Konidaris, G., Chitta, S., Marthi, B., and Barto, A. G. (2015). Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157.

[Ning et al., 2016] Ning, G., Zhang, Z., Huang, C., He, Z., Ren, X., and Wang, H. (2016). Spatially supervised recurrent convolutional neural networks for visual object tracking. *arXiv:1607.05781*.

[Oliva and Torralba, 2001] Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175.

[Oliva et al., 2003] Oliva, A., Torralba, A., Castelhano, M. S., and Henderson, J. M. (2003). Top-down control of visual attention in object detection. In *Proceedings 2003 International Conference on Image Processing*, volume 1.

[O'Rourke, 1987] O'Rourke, J. (1987). *Art Gallery Theorems and Algorithms*. Oxford University Press.

[Osband et al., 2016] Osband, I., Blundell, C., Pritzel, A., and Roy, B. V. (2016). Deep exploration via bootstrapped DQN. *CoRR*, abs/1602.04621.

[Parsons, 1978] Parsons, T. D. (1978). Pursuit-evasion in a graph. In *Theory and Applications of Graphs*, pages 426–441. Springer Berlin Heidelberg.

[Paul et al., 2014] Paul, R., Feldman, D., Rus, D., and Newman, P. (2014). Visual precis generation using coresets. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1304–1311.

[Pokorny et al., 2016a] Pokorny, F. T., Goldberg, K., and Kragic, D. (2016a). Topological trajectory clustering with relative persistent homology. In *IEEE ICRA*.

[Pokorny et al., 2016b] Pokorny, F. T., Hawasly, M., and Ramamoorthy, S. (2016b). Topological trajectory classification with filtrations of simplicial complexes and persistent homology. *The International Journal of Robotics Research*, 35(1-3):204–223.

[Pokorny and Kragic, 2015] Pokorny, F. T. and Kragic, D. (2015). Data-driven topological motion planning with persistent cohomology. In *Proceedings of Robotics: Science and Systems*, Rome, Italy.

[Pokorny et al., 2016c] Pokorny, F. T., Kragic, D., Kavraki, L. E., and Goldberg, K. (2016c). High-dimensional winding-augmented motion planning with 2d topological task projections and persistent homology. In *IEEE ICRA*.

[Puterman, 1994] Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition.

[Puydupin-Jamin et al., 2012] Puydupin-Jamin, A. S., Johnson, M., and Bretl, T. (2012). A convex approach to inverse optimal control and its application to modeling human locomotion. In *IEEE International Conference on Robotics and Automation*, pages 531–536.

[R. Hartley, 1997] R. Hartley, P. S. (1997). Triangulation. In *CVIU*, pages 146–157.

[Ramachandran and Amir, 2007] Ramachandran, D. and Amir, E. (2007). Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2586–2591.

[Ratliff et al., 2006] Ratliff, N., Bagnell, J. A. D., and Zinkevich, M. (2006). Maximum margin planning. In *International Conference on Machine Learning*.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788.

[Redmon and Farhadi, 2016] Redmon, J. and Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. *arXiv preprint arXiv:1612.08242*.

[Reed and de Freitas, 2015] Reed, S. E. and de Freitas, N. (2015). Neural programmer-interpreters. *CoRR*, abs/1511.06279.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*.

[Rogers et al., 1994] Rogers, C., Garrison, G., Grober, R., Hillis, Z., and Frankie, M. (1994). Coral reef monitoring manual for the caribbean and western atlantic. *Virgin Islands National Park, 110 p. Ilus.*

[Ross et al., 2014] Ross, S., Pineau, J., Paquet, S., and Chaib-draa, B. (2014). Online planning algorithms for pomdps. *CoRR*, abs/1401.3436.

[Rothkopf and Dimitrakakis, 2011] Rothkopf, C. A. and Dimitrakakis, C. (2011). Preference elicitation and inverse reinforcement learning. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III*, ECML PKDD'11, pages 34–48.

[Roy, 2017] Roy, D. (2017). Probabilistic Programming. http://www.probabilistic-programming.org. Accessed: 2018-3-19.

[Roy and McCallum, 2001] Roy, N. and McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 441–448, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, pages 2564–2571.

[Sadigh et al., 2017] Sadigh, D., Dragan, A. D., Sastry, S., and Seshia, S. A. (2017). Active preference-based learning of reward functions. In *Robotics: Science and Systems XIII*.

[Sandholm and Boutilier, 2006] Sandholm, T. and Boutilier, C. (2006). Preference elicitation in combinatorial auctions. In *Combinatorial Auctions*, pages 233–264. MIT Press.

[Sattar and Dudek, 2009a] Sattar, J. and Dudek, G. (2009a). *A Boosting Approach to Visual Servo-Control of an Underwater Robot*, pages 417–428. Springer Berlin Heidelberg.

[Sattar and Dudek, 2009b] Sattar, J. and Dudek, G. (2009b). Robust servo-control for underwater robots using banks of visual filters. In *IEEE International Conference on Robotics and Automation*, pages 3583–3588.

[Sattar and Dudek, 2009c] Sattar, J. and Dudek, G. (2009c). Underwater human-robot interaction via biological motion identification. In *Robotics: Science and Systems*.

[Sattar et al., 2008a] Sattar, J., Dudek, G., Chiu, O., Rekleitis, I., Giguere, P., Mills, A., Plamondon, N., Prahacs, C., Girdhar, Y., Nahon, M., et al. (2008a). Enabling autonomous capabilities in underwater robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3628–3634.

[Sattar et al., 2008b] Sattar, J., Dudek, G., Chiu, O., Rekleitis, I., Giguère, P., Mills, A., Plamondon, N., Prahacs, C., Girdhar, Y., Nahon, M., and Lobos, J.-P. (2008b). Enabling Autonomous Capabilities in Underwater Robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS)*, pages 3628–3634, Nice, France.

[Sattar et al., 2009] Sattar, J., Giguere, P., and Dudek, G. (2009). Sensor-based behavior control for an autonomous underwater vehicle. *Int. J. Robotic Research*, 28(6):701–713.

[Schein and Ungar, 2007] Schein, A. I. and Ungar, L. H. (2007). Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265.

[Schneiderman et al., 1995] Schneiderman, H., Nashman, M., Wavering, A. J., and Lumia, R. (1995). Vision-based robotic convoy driving. *Machine Vision and Applications*, 8(6):359–364.

[Sermanet et al., 2017] Sermanet, P., Xu, K., and Levine, S. (2017). Unsupervised perceptual rewards for imitation learning. In *Proceedings of Robotics: Science and Systems*.

[Settles, 2010] Settles, B. (2010). Active learning literature survey. Technical report.

[Seung et al., 1992] Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 287–294, New York, NY, USA. ACM.

[Shani et al., 2013a] Shani, G., Pineau, J., and Kaplow, R. (2013a). A survey of point-based POMDP solvers. *AAMAS*, 27(1):1–51.

[Shani et al., 2013b] Shani, G., Pineau, J., and Kaplow, R. (2013b). A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1).

[Shen et al., 2015] Shen, S., Michael, N., and Kumar, V. (2015). Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5303–5310. IEEE.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600.

[Shkurti et al., 2017] Shkurti, F., Chang, W., Henderson, P., Islam, M., Gamboa Higuera, J., Li, J., Manderson, T., Xu, A., Dudek, G., and Sattar, J. (2017). Underwater multi-robot convoying using visual tracking by detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4189–4196, Vancouver, Canada.

[Shkurti and Dudek, 2013] Shkurti, F. and Dudek, G. (2013). On the complexity of searching for an evader with a faster pursuer. In *IEEE ICRA*, pages 4047–4052.

[Shkurti and Dudek, 2017] Shkurti, F. and Dudek, G. (2017). Topologically distinct trajectory predictions for probabilistic pursuit. *IEEE International Conference on Intelligent Robots and Systems*.

[Shkurti et al., 2018] Shkurti, F., Kakodkar, N., and Dudek, G. (2018). Model-Based Probabilistic Pursuit via Inverse Reinforcement Learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia.

[Shkurti et al., 2012] Shkurti, F., Xu, A., Meghjani, M., Gamboa Higuera, J., Girdhar, Y., Giguere, P., Dey, B., Li, J., Kalmbach, A., Prahacs, C., Turgeon, K., Rekleitis, I., and Dudek, G. (2012). Multi-Domain Monitoring of Marine Environments Using a Heterogeneous Robot Team. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1747–1753, Algarve, Portugal.

[Silver and Veness, 2010] Silver, D. and Veness, J. (2010). Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems 23*, pages 2164–2172.

[Sim et al., 2004] Sim, R., Dudek, G., and Roy, N. (2004). Online control policy optimization for minimizing map uncertainty during exploration. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 2, pages 1758–1763.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

[Sondik, 1978] Sondik, E. J. (1978). The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304.

[Sourabh Bhattacharya, 2008] Sourabh Bhattacharya, S. H. (2008). Approximation Schemes for Two-Player Pursuit Evasion Games with Visibility Constraints. In *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

[Steinhardt, 2018] Steinhardt, J. (2018). Model Mis-specification and Inverse Reinforcement Learning. https://jsteinhardt.wordpress.com/2017/02/07/model-mis-specification-and-inverse-reinforcement-learning/. Accessed: 2018-3-19.

[Stiffler and O'Kane, 2012] Stiffler, N. M. and O'Kane, J. M. (2012). Shortest paths for visibility-based pursuit-evasion. *IEEE ICRA*, pages 3997–4002.

[Strasdat et al., 2011] Strasdat, H., Davison, A. J., Montiel, J. M. M., and Konolige, K. (2011). Double window optimisation for constant time visual slam. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2352–2359.

[Strasdat et al., 2010] Strasdat, H., Montiel, J. M. M., and Davison, A. J. (2010). Scale drift-aware large scale monocular slam. In *In Proceedings of Robotics: Science and Systems*.

[Sun et al., 2019] Sun, S., Zhang, G., Shi, J., and Grosse, R. (2019). Functional variational bayesian neural networks. *CoRR*, abs/1903.05779.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

[Sutton et al., 1999] Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA. MIT Press.

[Szepesvari, 2010] Szepesvari, C. (2010). *Algorithms for Reinforcement Learning*. Morgan and Claypool Publishers.

[Tamar et al., 2016] Tamar, A., Levine, S., and Abbeel, P. (2016). Value iteration networks. *CoRR*, abs/1602.02867.

[Theodorou et al., 2010] Theodorou, E., Buchli, J., and Schaal, S. (2010). A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11:3137–3181.

[Thiébaux et al., 2006] Thiébaux, S., Gretton, C., Slaney, J., Price, D., and Kabanza, F. (2006). Decision-theoretic planning with non-markovian rewards. *Journal of Artificial Intelligence Research*, 25(1):17–74.

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

[Tong and Koller, 2002] Tong, S. and Koller, D. (2002). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66.

[Torralba et al., 2006] Torralba, A., Castelhano, M. S., Oliva, A., and Henderson, J. M. (2006). Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search. *Psychological Review*, 113:2006.

[Tribou et al., 2015] Tribou, M. J., Wang, D. W. L., and Waslander, S. L. (2015). Degenerate motions in multicamera cluster SLAM with non-overlapping fields of view. *CoRR*, abs/1506.07597.

[Tsotsos, 1995] Tsotsos, J. (1995). Modeling visual attention via selective tuning. *Artificial Intelligence*, 78(1):507 – 545.

[Tsotsos, 2011] Tsotsos, J. K. (2011). *A Computational Perspective on Visual Attention*. The MIT Press, 1st edition.

[Underwood et al., 2006] Underwood, G., Foulsham, T., van Loon, E., Humphreys, L., and Bloyce, J. (2006). Eye movements during scene inspection: A test of the saliency map hypothesis. *European Journal of Cognitive Psychology*, 18(3):321–342.

[van den Oord et al., 2016a] van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016a). Pixel recurrent neural networks. *CoRR*, abs/1601.06759.

[van den Oord et al., 2016b] van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016b). Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328.

[Vidal et al., 2002] Vidal, R., Shakernia, O., Kim, H. J., Shim, D. H., and Sastry, S. (2002). Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669.

[Wierstra et al., 2010] Wierstra, D., Förster, A., Peters, J., and Schmidhuber, J. (2010). Recurrent policy gradients. *Logic Journal of the IGPL*, 18:620–634.

[Wolfe, 1994] Wolfe, J. M. (1994). Guided search 2.0 a revised model of visual search. *Psychonomic Bulletin & Review*, 1(2):202–238.

[Wulfmeier et al., 2015] Wulfmeier, M., Ondruska, P., and Posner, I. (2015). Deep inverse reinforcement learning. *CoRR*, abs/1507.04888.

[Wulfmeier et al., 2017] Wulfmeier, M., Rao, D., Wang, D. Z., Ondruska, P., and Posner, I. (2017). Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087.

[Xu et al., 2017] Xu, D., Nair, S., Zhu, Y., Gao, J., Garg, A., Fei-Fei, L., and Savarese, S. (2017). Neural task programming: Learning to generalize across hierarchical tasks. *CoRR*, abs/1710.01813.

[Yamauchi, 1997] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151.

[Yang et al., 2009] Yang, M., Wu, Y., and Hua, G. (2009). Context-aware visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1195–1209.

[Yang and Loog, 2018] Yang, Y. and Loog, M. (2018). A benchmark and comparison of active learning for logistic regression. *Pattern Recognition*, 83:401 – 415.

[Yen, 1971] Yen, J. Y. (1971). Finding the K-Shortest Loopless Paths in a Network. *Management Science*, 17(11):712–716.

[Yu et al., 2018] Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., and Darrell, T. (2018). BDD100K: A diverse driving video database with scalable annotation tooling. abs/1805.04687.

[Yu et al., 2008] Yu, Q., Dinh, T. B., and Medioni, G. (2008). Online tracking and reacquisition using co-trained generative and discriminative trackers. In *10th European Conference on Computer Vision: Part II*, pages 678–691.

[Zhang et al., 2017] Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. (2017). Noisy natural gradient as variational inference. *arXiv preprint arXiv:1712.02390*.

[Zhaoping, 2014] Zhaoping, L. (2014). *Understanding Vision: Theory, Models, and Data*. Oxford Press, 1st edition.

[Zhou et al., 2016] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. (2016). Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*.

[Zhou et al., 2017] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. (2017). Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[Zhu, 2006] Zhu, X. (2006). Semi-supervised learning literature survey.

[Zhu, 2015] Zhu, X. (2015). Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*.

[Zhu et al., 2003] Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the*

Twentieth International Conference on International Conference on Machine Learning, ICML'03, pages 912–919. AAAI Press.

[Ziebart et al., 2008] Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *23rd National Conference on Artificial Intelligence - Volume 3*, pages 1433–1438. AAAI.

[Ziebart et al., 2009] Ziebart, B. D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J. A., Hebert, M., Dey, A. K., and Srinivasa, S. (2009). Planning-based prediction for pedestrians. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936.