Overcoming Data Shortages In Robotic Applications

Wei-Di Chang

Masters of Engineering (Thesis)

Department of Electrical and Computer Engineering
McGill University
Montreal, Quebec, Canada

A thesis submitted to McGill University in partial
fulfillment of the requirements of the degree of
Masters of Engineering

Abstract

The abundance of data has played a major role in the recent machine learning boom. The availability of large datasets however, while useful, hasn't directly translated to successful applications in robotics. This thesis attempts to present ways to overcome the lack of data in robotics applications through domain adaptation methods. We propose and study the performance in data-poor scenarios of algorithms designed for two Robotics applications: visual tracking and learning from demonstrations. We first present a robust multi-robot convoying approach that relies on visual detection of the leading agent. Our method is based on the idea of tracking-by-detection, which interleaves efficient object detection with temporal filtering of image-based bounding box estimation. Using a bounding box annotated dataset of images extracted from footage of an underwater robot in ocean settings, we compare multiple tracker variants, including several convolutional neural networks with and without recurrent connections and frequency-based model-free trackers. We investigate the domain adaptation ability of our most applicable architecture through training on synthetic data, generated from a realistic game engine. To demonstrate the practicality of this tracking-by-detection strategy in real-world scenarios, we successfully control a 5-DOF legged underwater robot to follow another robot's independent motion. We then focus on the impact of data shortages when learning from demonstration. Extending the options framework with the notion of reward options, we develop a method for learning joint reward-policy options in the context of generative adversarial inverse RL and show that methods in this context suffer in demonstration data-poor scenarios. We then study the one shot domain adaptation abilities of our approach. That is, given expert demonstrations from a mixture of environments with different dynamics, can the agent learn to properly complete a task in a previously unseen environment with different dynamics. Our results show that our method is able to successfully learn a task in these domain adaptation scenarios, and significantly outperforms inverse RL without options.

Abrégé

L'abondance de données a joué un rôle majeur dans le récent boom de l'apprentissage automatique. Mais, la disponibilité de ces larges ensembles de données, bien qu'utile, ne s'est pas directement traduite à des applications réussies en robotique. Cette thèse présente des moyens de pallier le manque de données dans les applications robotiques grâce à des méthodes d'adaptation de domaine. Nous proposons et étudions la performance d'algorithmes conçus pour deux applications en robotiques dans des situations de manque de données: le suivi visuel et l'apprentissage à partir de démonstrations. Nous présentons d'abord une approche robuste de convoi à robots multiples qui repose sur la détection visuelle de l'agent principal. Notre méthode est basée sur l'idée de suivi par détection, qui couple la détection d'objet avec le filtrage temporel de l'estimation de la boîte englobante de l'objet dans l'image. À l'aide d'un ensemble de données constitué d'images d'un robot sous-marin annotées de boîtes englobantes, nous comparons plusieurs variantes d'algorithmes de suivi visuel, dont plusieurs réseaux neuronaux convolutifs avec et sans connexions récurrentes, ainsi que des algorithmes de suivi visuels par fréquence. Nous étudions la capacité d'adaptation de domaine de notre architecture la plus prometteuse à partir d'un entrainement sur données synthétiques, générées grâce à un moteur de jeu réaliste. Pour démontrer la praticité de cette stratégie de suivi par détection dans des scénarios réels nous contrôlons avec succès un robot sous-marin à nageoires, capable de cinq degrés de liberté, pour suivre le mouvement d'un autre robot indépendant. Nous nous concentrons ensuite sur l'impact des pénuries de données lors de l'apprentissage par démonstration. L'on étend le cadre des options en Apprentissage par Renforcement (AR) avec la notion d'options de récompense, développons une méthode d'apprentissage des options de politiques et récompenses conjointe dans le contexte de l'AR inverse génératif et adversairial, et montrons que les méthodes dans ce contexte souffrent dans les cas de manque de démonstrations. Nous étudions ensuite les capacités d'adaptation de domaine en un coup de notre approche. C'est-à-dire que, étant donné des démonstrations expertes provenant d'un mélange d'environnements avec des dynamiques variées, l'agent peut-il apprendre à ef-

fectuer correctement une tâche dans un environnement inconnu avec des dynamiques différentes. Nos résultats montrent que notre méthode est capable d'apprendre une tâche avec succès dans ces scénarios d'adaptation de domaine et surpasse significativement l'AR inverse sans options.

Contributions

This thesis contributes the analysis of remediation methods to data scarce scenarios in robotics:

- Investigation of the effect of using synthetic data obtained from simulation in the training of deep convolutional neural network-based detection methods for underwater vision-based tracking (Chapter 2)
- Investigation of the effect of data scarcity for adversarial inverse reinforcement learning algorithms, and remediation methods through one shot domain adaptation (Chapter 3)

In addition to these two lines of investigations in the performance of algorithms for robotics in data-poor scenarios, the author is not the sole owner but has contributed in significant ways to the following:

- Reduction of the YOLO detection framework for running onboard the AQUA robot (Chapter 2)
- Inception and training of the Recurrent Reduced YOLO detection framework onboard the AQUA robot (Chapter 2)
- Training of the YOLO, Reduced YOLO, and Recurrent Reduced YOLO detection frameworks for tracking the AQUA robot (Chapter 2)
- Experimental results for YOLO and Recurrent YOLO variants (Chapter 2)
- In-ocean deployments of the AQUA robot for data collection and testing of the formulated tracking-by-detection framework (Chapter 2)

- Extension of the options framework to reward options (Chapter 3)
- Creation of OptionGAN, an optionated version of Generative Adversarial Inverse Reinforcement Learning, for use of extended options framework in inverse RL (Chapter 3)
- Empirical results demonstrating OptionGAN's effectiveness in continuous control tasks and one-shot transfer learning in inverse RL (Chapter 3)

Acknowledgements

I would like to thank all the people who have influenced me during the course of my graduate studies.

I want to thank both my supervisors Michael Rabbat and Gregory Dudek for their support, encouragement and always helpful discussions.

I would also like to thank the members of the Mobile Robotics Laboratory, past and present: Anqi, Florian, Juan, Jimmy, Travis, Sandeep, Johanna, Nikhil, Andrew, Arnold, Victor, Herke, Emma, Karim, Peter, Edward, Scott, Lucas, Yi Tian, Monica, Ian, Isabelle, Nik, Dave and Greg; all of them have always been open to discuss anything and were very helpful, both in the lab and outside. The team the Mobile Robotics Laboratory members forms and the dedication they carry will always be a source of inspiration.

And finally, of course, a special thank you to my family and close friends for their incessant loving support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last decade, impressive results have been shown in fields neighbouring and overlapping robotics, namely artificial intelligence, machine learning and computer vision. In computer vision for instance, Alexnet and many subsequent neural networks have conquered the leaderboards at the Imagenet Large Scale Visual Recognition Competition [53, 34, 26], while in reinforcement learning, AlphaGo [64] beat human champions at the game of Go and Atari-solving agents learned to beat the game at human levels [44]. These successes were achieved in part by advances in computational power, but most importantly, the availability of huge amounts of data: ImageNet database contains approximately 1.2 million labelled images, AlphaGo used more than 38 million positions to train their algorithm to play Go, and [44] used more than 38 days of play to train their agents to win Atari.

While robotics has benefited from this surge of data, it hasn't experienced the same data avalanche that neighbouring fields have. Similarly to many fields where data requires acquisition through physical experiments, robotics faces the real world and the process of obtaining a data point is often expensive, time-consuming, and sometimes even dangerous. Oftentimes, a skilled operator is required to supervise the data collection, and the data collection process can wear down or even break parts of the robot. Furthermore, trying something takes a significant amount of time, and seeing the consequence of each trial might even take longer. As such, requiring a million trials to succeed at a task is infeasible when facing the real-world. While virtual data is abundant and cheap, data is scarce in the physical world.

One solution to the data scarcity problem in robotics and other fields originates from domain adaptation which focuses on learning from a source distribution a model that performs well on a different (but related) target data distribution. Using this method, we can supplement the small amount of available data, or even remove the need for data from the target domain D_T using data obtained from a different but related domain D_S. While the "distance" between the source and target domain has no defined bound, of course the two domains need to share *some* information about their counterpart for it to be useful. [68] for example, uses randomized augmentations of the original data from domain D_T as supplemental data from the 'related' domain D_S. Synthetic data can also be used, from generative models such as in [19], or from complex simulators mimicking D_T [48].

1.1 Outline

In this thesis we present two state of the art algorithms with applications in robotics, analyze their performance in data scarce scenarios, and overcome some of their observed limitations using domain adaptation techniques. Due to the difference in the two application domains (one in perception and the other in control), we include relevant background and related work sections in each chapter.

In chapter 2, we tackle the robot convoying problem in the underwater domain and present the problem in section 2.1. We first present in section 2.3 the small dataset available to us for the task and the data synthesis method used to generate extra data. We then present our own as well as competing approaches and compare them on test datasets in sections 2.5. and 2.4 respectively. The results of training on synthetic data for our algorithm are shown in 2.6.2. Finally, field experiment results are shown and analyzed in section 2.6.5.

In chapter 3, we present OptionGAN, an algorithm for learning joint reward-policy options from demonstrations. We first present the learning from demonstration paradigm in section 3.1 and its applicability to robotics in sections 3.3-3.4. We then present our algorithm and methodology in sections 3.5-3.7. Finally, we present our results in section 3.8, and OptionGAN's robustness to domain adaptation scenarios in section 3.8.5.

Chapter 2

Domain Adaptation in the Visual Domain

2.1 Introduction

Convoying behaviours have a wide range of useful application contexts in robotics. Multi-agent systems for example, where a group of simple robots work in concert towards a goal as opposed to a single larger complex robot, have the potential to be more efficient, economical, and less prone to failure. In modern warehouses, large swarms of robots work together, sometimes adopting convoying behaviours to automate and accelerate the shipping of products to the customer. In a close future, one could easily imagine synchronized convoying behaviour applied to self-driving cars or freight trucks, thus reducing traffic congestion. Robot assistant scenarios, where autonomous mobile robots assist Humans in tasks are another. Numerous applications arise from these scenarios, ranging from nursing home robot assistants to firefighter companions.

Vision-based tracking solutions have been applied to robot convoying in a variety of contexts, including terrestrial driving [58, 20], railroad maintenance vehicles [38], and unmanned aerial vehicles [37].

In the underwater realm, convoying tasks face great practical difficulties due to highly varied lighting conditions, external forces, and hard-to-model currents on the robot. Previous work in terrestrial and aerial systems use fiducial markers on the targets to aid tracking. In this chapter, we present a more general *tracking-by-detection* approach that is trained solely on the natural

2.1 Introduction

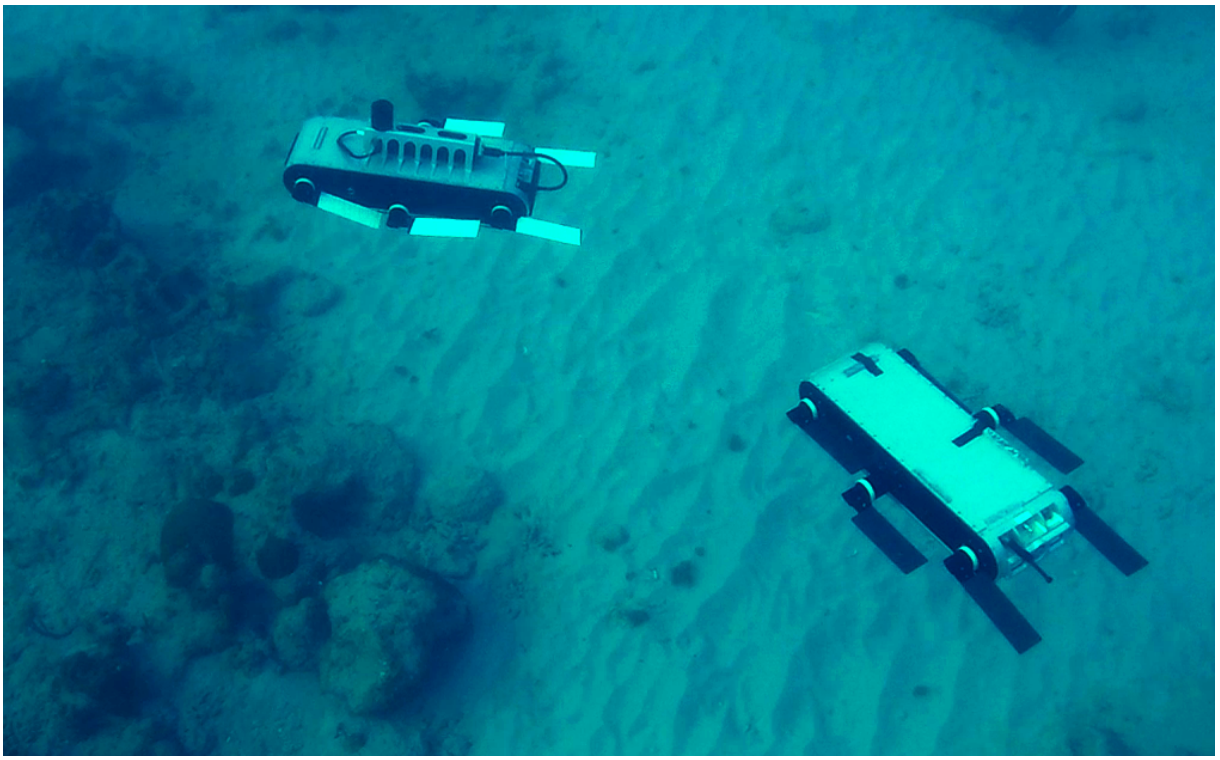

Figure 2.1: A sample image from our underwater convoying field trial using Aqua hexapods [57]. Videos of our field trials, datasets, code, as well as more information about the project are available at http://www.cim.mcgill.ca/~mrl/robot_tracking

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 markers may not be visible. Other works have demonstrated successful tracking methods using auxiliary devices for underwater localization, including mobile beacons [10], aerial drones [18], or acoustic sampling [13]. While these alternative strategies can potentially be deployed for multi-robot convoy tasks, they require additional costly hardware.

This is achieved through tracking-by-detection, which combines a target detection method to first localize the object of interest, and a recurrent approach for temporally filtered image-based position estimation. Our solution is built upon several autonomous systems for enabling underwater tasks for a hexapod robot [57, 55, 54, 22, 40], as well as recent advances in real-time deep learning-based object detection frameworks [51, 52].

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 [57]. 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 [51], [52], [29], 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.

Given the difficulty of collecting underwater imagery, a relatively small dataset is available to us for the training of deep learning methods. We thus generate a dataset of synthetic underwater visual data generated in Unreal Engine, a game engine repurposed as a simulator for our use case, and investigate how well our most promising architecture performs if no real data was available and was trained solely on synthetic data.

Furthermore, we demonstrate in an open-water field trial that one of our proposed architectures, based on YOLO [50] 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.

2.2 Related Work

2.2.1 Vision-Based Convoying

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 [2]. 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 [17]. Tracking was enhanced by both suitable engi-

neered 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 [58, 20]. 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 [21], 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.

2.2.2 Tracking Methods

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 [72], are typically initialized with a bounding box of an arbitrary target, and they adapt to viewpoint changes through semi-supervised learning. The TLD tracker [32], 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 [39], 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. In our work we opted to avoid these methods because of their

susceptibility to errors in terms of occlusion and non-rigid motion.

Works such as [7, 45] use convolutional neural networks and rely on supervised learning to learn a generic set of target representations. Our work herein is more closely related to this body of work, however, we are interested in a single target with known appearance.

2.3 Aqua Dataset

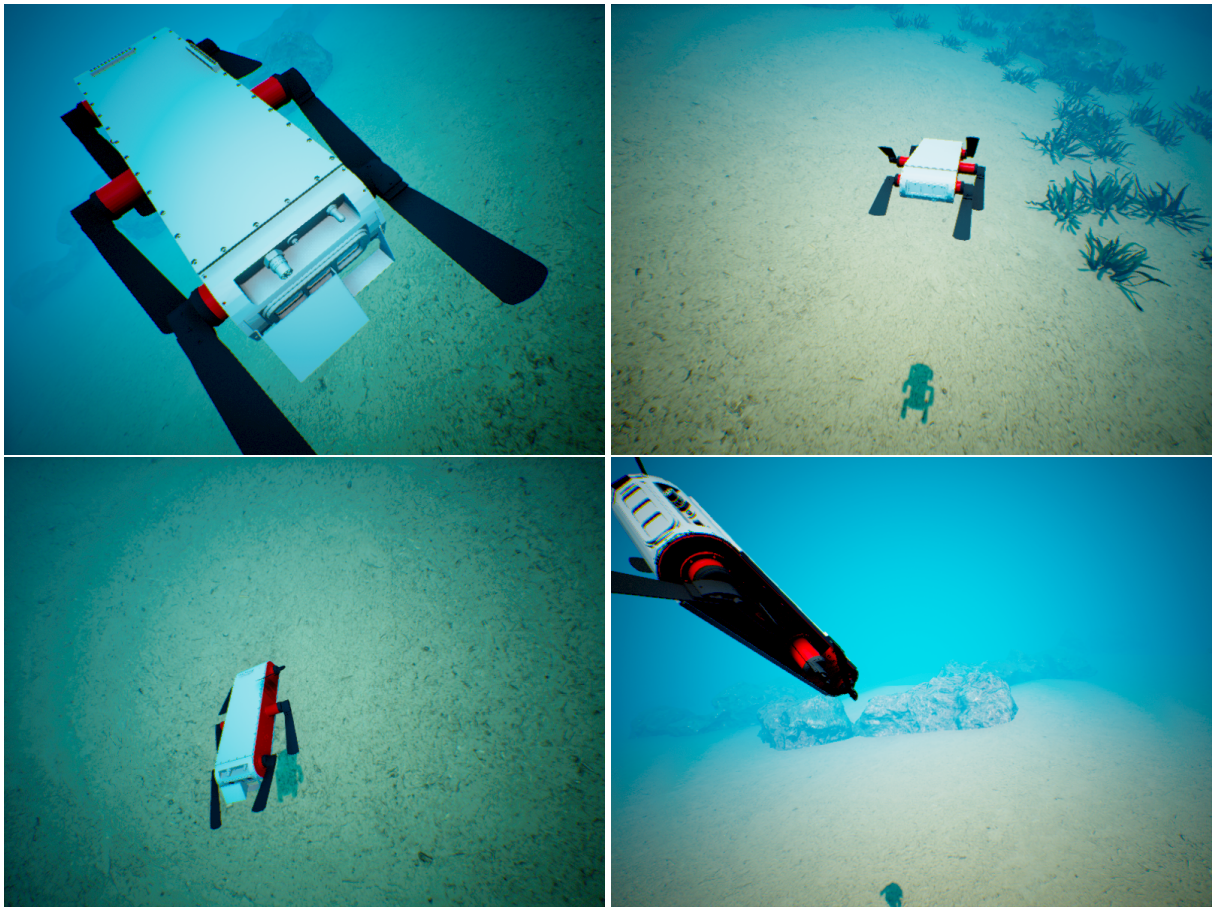

Figure 2.2: Synthetic images generated in Unreal Engine

2.3.1 Dataset Collection

We collected a dataset[1] of video recordings of the Aqua family of hexapods during field trials at McGill University's Bellairs Research Institute in Barbados. This dataset includes approximately 5200 third-person view images extracted from video footage of the robots swimming in various underwater environments recorded from the diver's point of view and nearly 10000 first-person point of view frames extracted from the tracking robot's front-facing camera.

We use the third-person video footage for training and validation, and the first-person video footage as a test set. This separation also highlights the way we envision our system to become widely applicable: the training data can be recorded with a handheld unit without necessitating footage from the robot's camera.

2.3.2 Synthetic Data Generation

Given the relatively small amount of data available for training the deep convolutional neural network-based methods, we generate a training dataset made up of synthetic data. We generate the data in a simulated 3D underwater environment using Unreal Engine 4, with ready-made assets and the CAD model of the AQUA robot. The location of the robot in the environment, joint angles, as well as the lighting, viewpoint, and distance to the camera (within reasonable bounds) are randomized to obtain a wide variety of images. This synthetic dataset contains only single timestep images as opposed to image sequences. In other words, we do not generate synthetic movie datasets for any of the recurrent methods that we examine here. Sample images from the synthetic dataset are shown in Figure 2.2.

2.4 Detection Methods

The initial stage in our tracking pipeline is to localize the object of interest (the AQUA robot in our case) within the image, that is: given an RGB image, output the bounding box pixel coordinates containing the object. We describe in this section the details for each compared method.

[1]The dataset, along with its ground truth annotations, can be found at http://www.cim.mcgill.ca/~mrl/robot_tracking

2.4.1 Non-Recurrent Methods

VGG

The VGG architecture [65] classification performance 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[2], 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

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

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.

In all of our variants, we pre-train the network on the ImageNet dataset as in [65] to drastically reduce training time and scale our dataset images to $(224, 224, 3)$ to match the ImageNet scaling.

YOLO

The YOLO detection system [50] 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 [51], but we found that inference 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 [29], we condensed the TinyYOLOv2 architecture as shown in Table 2.1. This enabled inference on embedded robot platforms at reasonable frame rates (13 fps). Following Ning [3] and [29] 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 2.2. 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 [51], we hypothesize

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

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.

We also use structures of two 1×1 filters followed by a single 3×3 filter, similar to Squeeze layers in SqueezeNet [29], to compress the inputs to 3×3 filters. Similarly to VGG [65] and the original YOLO architecture [50], 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.

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 2.1: TinyYOLOv2 architecture

2.4.2 Recurrent Methods

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 [47]. 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

2.4 Detection Methods

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 2.2: Our ReducedYOLO architecture

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:

$$I((\sqrt{\bar{x}}-\sqrt{x})^2+(\sqrt{\bar{y}}-\sqrt{y})^2)\alpha_{coord}+I((\sqrt{\bar{w}}-\sqrt{w})^2+(\sqrt{\bar{h}}-\sqrt{h})^2)\alpha_{coord} \\ +I(\text{IOU}-p)^2\alpha_{obj}+(1-I)(\text{IOU}-p)^2\alpha_{no\_obj} \quad (2.1)$$

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 [51]. 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 [24]. A general diagram of our LSTM architecture can be seen in Figure 2.3. Our recurrent detection implementation, based partially on code provided by [47], is made publicly accessible.[4]

2.4.3 Methods Based on Frequency-Domain Detection

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 [56, 30] by underwater robots to track scuba divers. Flippers of a human diver typically oscillate at fre-

[4] https://github.com/Breakend/TemporalYolo

2.4 Detection Methods

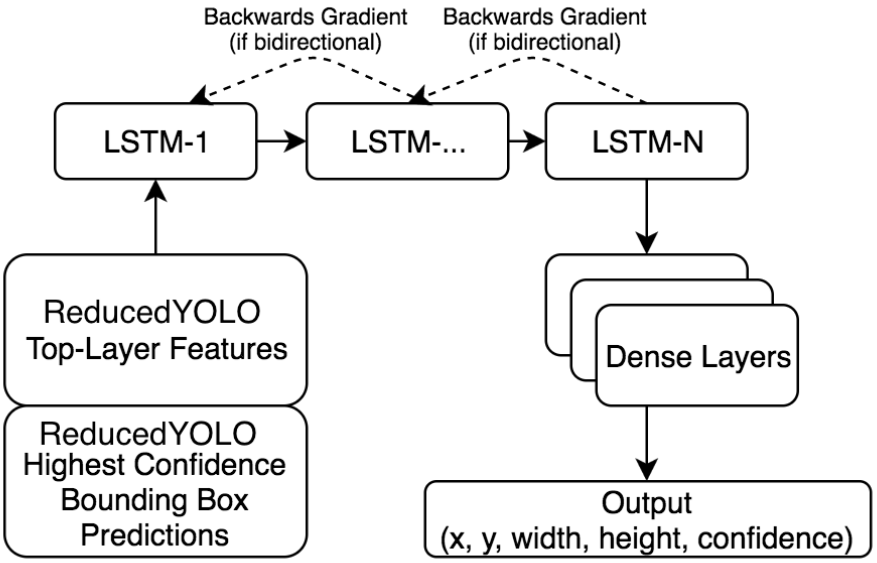

Figure 2.3: Overview of our Recurrent ReducedYOLO (RROLO) architecture. The original ROLO work [47] did not use bidirectional, dense layers, or multiple LSTM cells in their experiments.

quencies between 1 and 2 Hz, which produces periodic intensity variations in 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.

We implement the mixed-domain motion (MDPM) tracker described by Islam *et al* [30]. An improved version of Sattar *et al* [56], the MDPM works as follows (illustrated in Figure 2.4) :

- 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

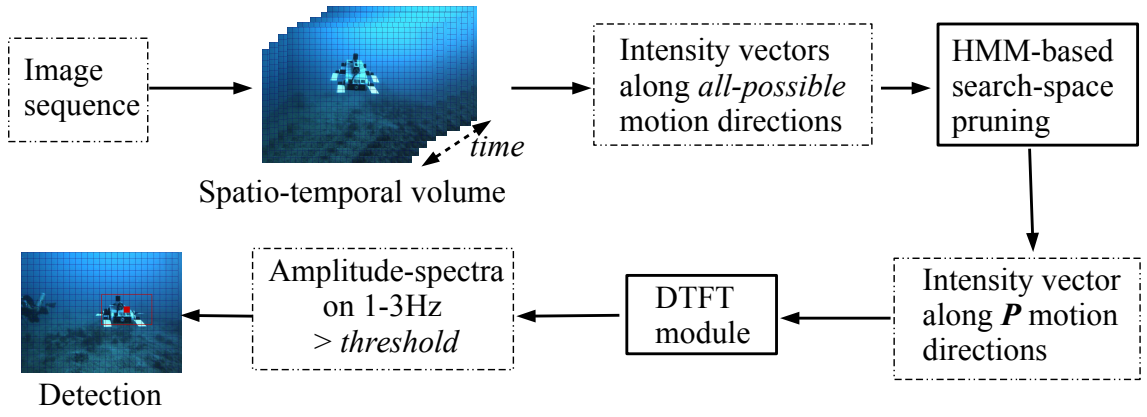

Figure 2.4: Outline of mixed-domain periodic motion (MDPM) tracker [30]

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.

2.5 Visual Servoing Controller

The Aqua family of underwater robots allows 5 degrees-of-freedom[5] control, which enables agile and fast motion in 3D. This characteristic makes vehicles of this family ideal for use in tracking applications that involve following other robots as well as divers [56].

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 controller 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

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

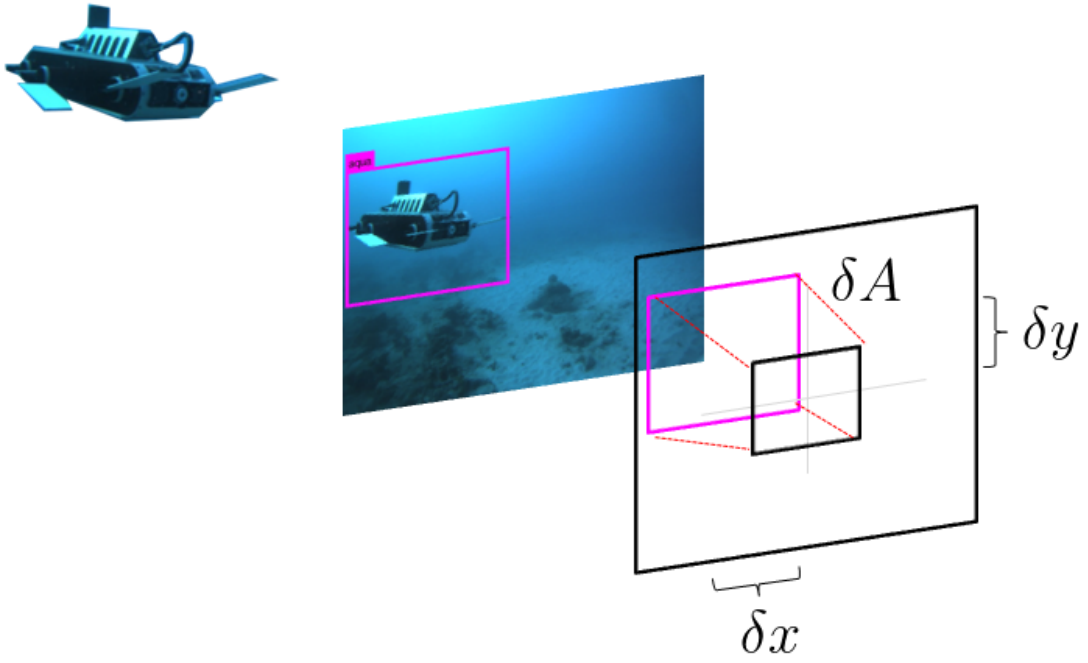

Figure 2.5: Errors used by the robot's feedback controller. δx is used for yaw control, δy for depth control, and the error in bounding box area, δA is used for forward speed control.

sources, as shown in Fig. 2.5, 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 [41]. 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.

2.6 Experimental Results

We evaluate each of the implemented methods on the common test dataset described in Section 2.3 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 [9]
- 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 2.3 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.

2.6.1 Non-Recurrent Methods

As we can see in Table 2.3, 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 a significant loss in 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

2.6 Experimental Results

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

Processing Units, and are equipped with low-power processing units. Additionally, note that the precision metric has not suffered while reducing the 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 [32] 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 leading robot exited the field of view of the following robot, and TLD could not always recover. This illustrates why model-free trackers are in general less suitable for convoying tasks than model-based trackers.

2.6.2 Synthetic Dataset Domain Adaptation

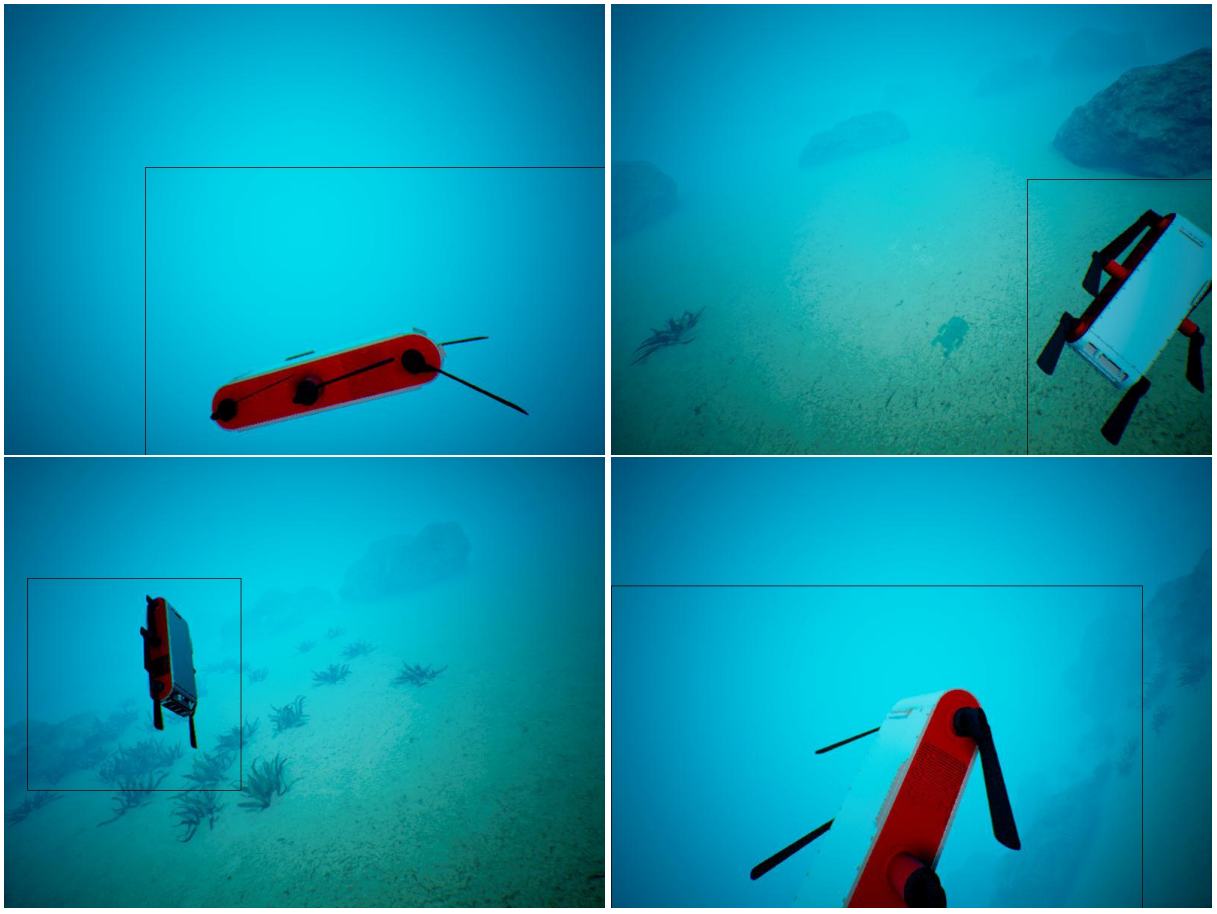

Figure 2.6: Synthetic images and corresponding bounding boxes generated in Unreal Engine. Only the top-right image is well bounded.

We investigate the feasibility of training the ReducedYOLO model on synthetic data only, as it is the closest to running in real time onboard the AQUA robot. We train the model using an increasing number of simulator generated images, going up to 50000 training images, and obtain our results on the common real images test dataset. The Results for this experiment are

N Synthetic Training Images	ACC	IOU	P	R	LFR
1000	0.000	0.000	0.000	0.000	N/A
5000	**0.542**	0.173	0.930	**0.459**	36%
10000	0.443	**0.189**	0.914	0.329	26%
20000	0.528	0.188	0.925	0.442	35%
30000	0.416	0.166	0.931	0.287	23%
40000	0.455	0.171	0.937	0.336	26 %
50000	0.390	**0.189**	**0.943**	0.247	**20%**

Table 2.4: ReducedYOLO Trained on synthetic data only

shown in Table 2.4.

As we can see, training only on synthetic data gives poor results compared to training on the real dataset. Synthetic data seems to make the performance of the algorithm worse as the number of synthetic training images increases. This goes against intuition given the fidelity of the visual data seen in Figure 2.2.

We find out through inspection of the annotations generated by Unreal Engine, that the ground truth bounding boxes output by the simulator are often oversized compared to the robot. We show a few examples of this issue in Figure 2.6. With these examples, the algorithm learns overly loose bounding boxes which include significant amounts of background scenery. Because of this, the resulting trained network naturally suffers in both localization and classification performance. In addition, with the YOLO detection framework randomizing the batches of data seen in training, the amount of well annotated data the network is trained on is unpredictable, thus explaining the poor coherence of the results as a function of the number of synthetic training images.

The poor annotations output by the simulator are due to its 3D nature as well as the pipeline used to output the annotations. We plan to fix this issue to obtain conclusive results.

2.6.3 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),

again, because this model can run closest to real time on our embedded robot system. Our results on the test set are shown in Table 2.3. 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 2.3, 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[6]. 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 do 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 and higher risk of overfitting. We attempted re-balancing and re-weighting the objective in our experiments and found

[6]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).

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.

2.6.4 Method Based on 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 2.3 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 [30].

2.6.5 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 al-

lowed us to verify the general robustness of our tracking-by-detection solution, and specifically to evaluate the possibility of overfitting to our training environments.

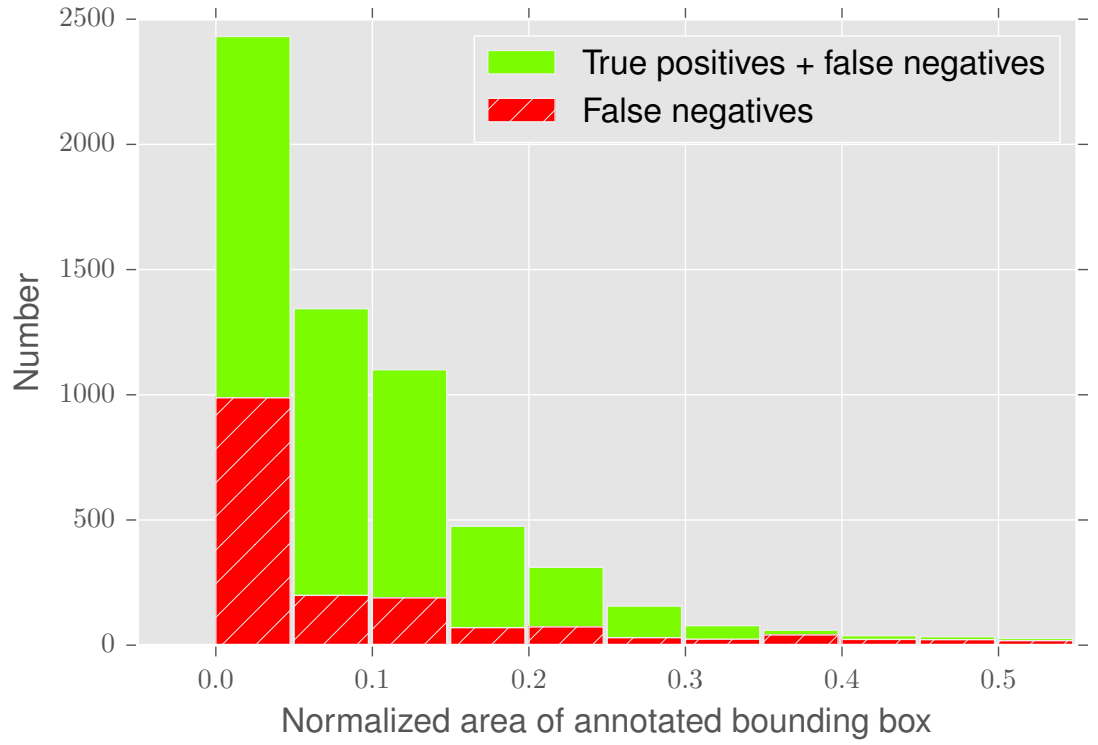

Figure 2.7: Histogram of true positive and false negative detections as a function of the area of annotated bounding boxes, as obtained from in-ocean robot tracking runs.

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.

Results

We configured the follower robot 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. 2.5.

The ReducedYOLO detector consistently over-estimated the small size of the target. Nevertheless, Fig. 2.8 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.

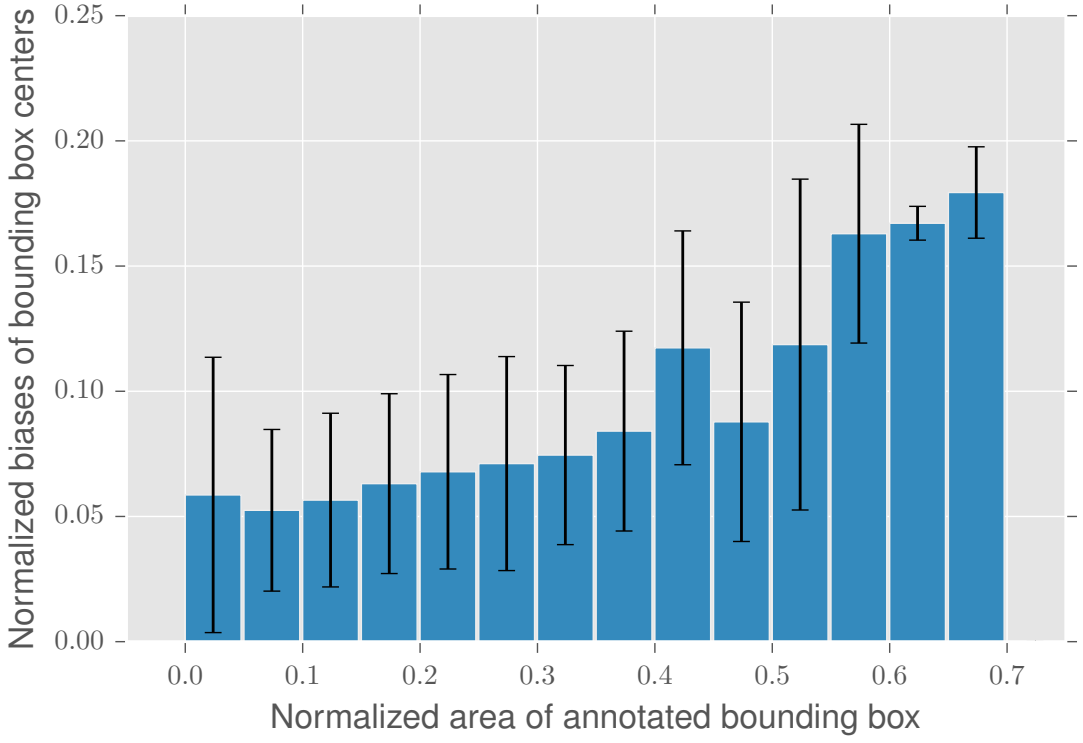

Figure 2.8: Histogram of average biases between detected vs. annotated bounding box centers, as obtained from in-ocean robot tracking runs. Bars indicate 1σ error.

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. 2.9, 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.

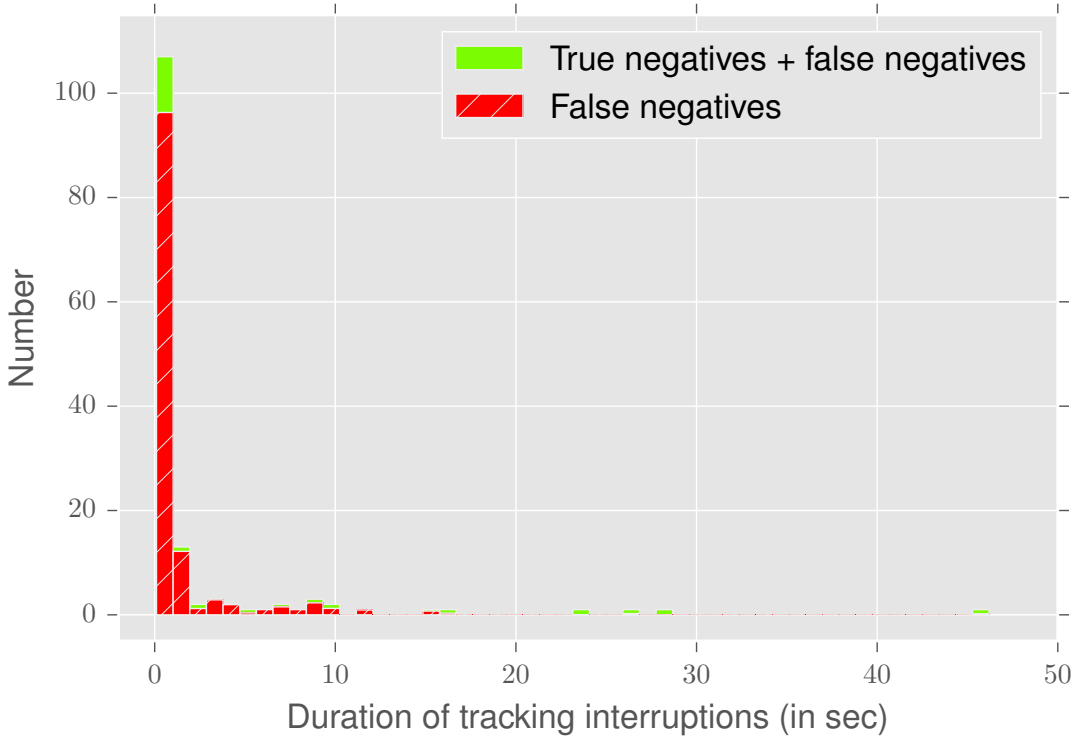

Figure 2.9: Histogram of true negative and false negative classifications in terms of their duration for our ReducedYOLO model, as obtained from in-ocean robot tracking runs.

2.7 Conclusions

We presented a general *tracking-by-detection* approach that relies on visual sensing of the appearance of non-engineered targets, and also capitalizes on recent advances in deep learning for deployment onboard robots with limited computing capabilities. We demonstrated the utility of several lightweight neural network architecture for appearance-based visual convoying, and we showed improvements made possible by recurrent extensions. We successfully performed multi-robot convoying in the open sea in practice, using supervised learning based on limited training data annotated beforehand. Furthermore, we carried out an extensive comparison of various tracker variants with respect to a multitude of desirable attributes for visual convoying tasks and investigated the feasibility of using synthetic data for these approaches by training and evaluating our best candidate architecture using only synthetic data.

Possible improvements through future research include improvement on temporal-based bounding box detection by making the entire architecture trainable end-to-end, the use of a higher fidelity simulator for better synthetic data, extending this work to visual servoing with multiple bounding boxes per frame, as well as robust target tracking, despite interruptions in visual contact by using stronger predictive models.

Chapter 3

Domain Adaptation to Learn from Demonstration

3.1 Introduction

Robot learning from demonstration (Lfd) is a paradigm to enable robots to reproduce a behavior for a task demonstrated by an expert, usually a human. It can therefore be used to develop new robot behaviors without writing code explicitly describing the behavior but instead by "showing" the behavior to the robot. In the context of autonomous cars for example, the "stop at red lights" behavior could be learned from human drivers driving lawfully instead of programming by hand the actions needed for the encountered conditions. This field holds much potential in a future where robots will become increasingly commonplace and mostly interact with users unfamiliar with the inner workings of a robot.

Lfd can be tackled with direct inverse reinforcement learning (also known as imitation learning or policy imitation), where the robot attempts to imitate the policy of the expert by observing its states and actions. In this case, the agent attempts to replicate the actions chosen by the expert policy without reasoning about the reward function the expert was optimizing over. Behavioral cloning is the simplest example of imitation learning, where it is treated as a supervised learning problem: given a sequence of expert states and actions, learn the mapping between each state and the corresponding action. However this approach lacks robustness as any deviation from the

demonstrated paths will lead to states the learner has not encountered and thus unpredictable behaviors. Bagnell's summary [5] of imitation learning presents many reasons why robust policies are not achievable only using supervised learning.

Indirect inverse reinforcement learning on the other hand, usually simply referred to as Inverse Reinforcement Learning (IRL), tackles the Lfd problem by first inferring about the reward function the expert demonstrator was optimized over, and imitates the expert behavior using forward reinforcement learning methods. From an engineering point of view, this line of work is also called *inverse optimal control*. While IRL inherently produces more generalizable policies, it has been determined that IRL is a degenerate problem [46] as there can be many reward functions that explain an observed optimal behavior. Multiplying any reward function by a positive scalar for instance will not change optimal behavior for that MDP.

In this chapter, we present an IRL algorithm applicable to robotics and demonstrate its robustness to data scarce scenarios through domain adaptation.

3.2 Background and Notation

3.2.1 Markov Decision Processes (MDPs)

MDPs consist of states S, actions A, a transition function $P : S \times A \rightarrow (S \rightarrow \mathbb{R})$, and a reward function $r : S \rightarrow \mathbb{R}$. We formulate our methods in the space of continuous control tasks ($A \in \mathbb{R}, S \in \mathbb{R}$) using measure-theoretic assumptions. Thus we define a parameterized *policy* as the probability distribution over actions conditioned on states $\pi_\theta : S \times A \rightarrow [0, 1]$, modeled by a Gaussian $\pi_\theta \sim \mathcal{N}(\mu, \sigma^2)$ where θ are the policy parameters. The value of a policy is defined as $V_\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s]$ and the action-value is $Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a]$, where $\gamma \in [0, 1)$ is the discount factor.

3.2.2 The Options framework

In reinforcement learning, an option ($\omega \in \Omega$) can be defined by a triplet $(I_\omega, \pi_\omega, \beta_\omega)$. In this definition, π_ω is called an intra-policy option, $I_\omega \subseteq S$ is an initiation set, and $\beta_\omega : S \rightarrow [0, 1]$ is a termination function (i.e. the probability that an option ends at a given state) [67]. Furthermore, π_Ω is the policy-over-options. That is, π_Ω determines which option π_ω an agent picks to use

until the termination function β_ω indicates that a new option should be chosen. Other works explicitly formulate *call-and-return* options, but in our case, we simplify to *one-step* options, where $\beta_\omega(s) = 1; \forall \omega \in \Omega, \forall s \in S$. One-step options have long been discussed as an alternative to temporally extended methods and often provide advantages in terms of optimality and value estimation [67, 16, 14].

3.2.3 Mixture-of-Experts

The idea of creating a mixture of experts (MoEs) was initially formalized to improve learning of neural networks by dividing the input space among several networks and then combining their outputs through a soft weighted average [31]. It has since come into prevalence for generating extremely large neural networks [63]. In our formulation of joint reward-policy options, we leverage a correspondence between Mixture-of-Experts and options. In the case of one-step options, the policy-over-options (π_Ω) can be viewed as a specialized gating function over experts (intra-options policies $\pi_\omega(a|s)$): $\sum_\omega \pi_\Omega(\omega|s)\pi_\omega(a|s)$. Several works investigate convergence to a sparse and specialized Mixture-of-Experts [31, 63].

3.2.4 Policy Gradients

Policy gradient (PG) methods [66] formulate a method for optimizing a parameterized policy π_θ through stochastic gradient ascent. In the discounted setting, PG methods optimize $\rho(\theta, s_0) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)|s_0\right]$.

The PG theorem states: $\frac{\delta \rho(\theta, s_0)}{\delta \theta} = \sum_s \mu_{\pi_\theta}(s|s_0) \sum_a \frac{\delta \pi_\theta(a|s)}{\delta \theta} Q_{\pi_\theta}(s, a)$, where $\mu_{\pi_\theta}(s|s_0) = \sum_{t=0}^{\infty} \gamma^t P(s_t = s|s_0)$ [66]. In Trust Region Policy Optimization (TRPO) [59] and Proximal Policy Optimization (PPO) [61] this update is constrained and transformed into the advantage estimation view such that the above becomes a constrained optimization: $\max_\theta \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t(s_t, a_t)\right]$ subject to $\mathbb{E}_t\left[\mathrm{KL}\left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right]\right] \leqslant \delta$ where $A_t(s_t, a_t)$ is the generalized advantage function according to [60]. In TRPO, this is solved as a constrained conjugate gradient descent problem, while in PPO the constraint is transformed into a penalty term or clipping objective.

3.2.5 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) was first formulated in the context of MDPs by [46]. In later work, a parametrization of the reward function is learned as a linear combination of the state feature expectation so that the hyperdistance between the expert and the novice's feature expectation is minimized [1]. It has also been shown that a solution can be formulated using the maximum entropy principle, with the goal of matching feature expectation as well [73]. Generative adversarial imitation learning (GAIL) makes use of adversarial techniques from [23] to perform a similar feature expectation matching [28]. In this case, a discriminator uses rollouts (state-action pair sequences) from the expert demonstrations and novice rollouts to learn a binary classification probability distribution. The probability that a state belongs to an expert demonstration can then be used as the reward for a policy optimization step. In other words, how well the novice "fools" the discriminator by tending closer to the expert behavior determines the reward it receives for a rollout.

3.3 Applying Lfd to Robotics

One goal in robotics research is to create a system which learns how to accomplish complex tasks simply from observing an expert's actions, such as videos of humans performing actions for instance. While IRL has been instrumental in working towards this goal, it has become clear that fitting a single reward function which generalizes across many domains is difficult.

To this end, several works investigate decomposing the underlying reward functions of expert demonstrations and environments in both IRL and RL [33, 62, 11, 3, 70]. For example, in [33], reward functions are decomposed into a set of subtasks based on segmenting expert demonstration transitions (known state-action pairs) by analyzing the changes in "local linearity with respect to a kernel function". Similarly, in [62], techniques in video editing based on information-similarity are adopted to divide a video demonstration into distinct sections which can then be recombined into a differentiable reward function.

However, simply decomposing the reward function may not be enough, the policy must also be able to adapt to different tasks. Several works have investigated learning a latent dimension along with the policy for such a purpose [25, 71, 35]. This latent dimension allows multiple

tasks to be learned by one policy and elicited via the latent variable. All of these hierarchical methods require however the attempted task to be specified explicitly by an oracle, through the latent variable.

3.4 One Shot Domain Adaptation

For IRL to be applicable to robotics applications, it would ideally be able to learn underlying reward functions and policies solely from human video demonstrations. We call such a case, where the demonstrations come from various domains and the task must be performed in a novel unseen environment, one-shot domain adaptation. For example, given only demonstrations of a human walking on earth, can an agent learn to walk on the moon?

Algorithms able to handle this case would have a wide range of applications in robotics, as it would allow the training of the agent without requiring any data from the final environment the robot will perform in. Training a policy using only expert demonstrations collected in a simulator to perform a task in the real world would reduce the need for human supervision, robot hardware wear, as well as experiment time.

However, such demonstrations would undoubtedly come from a wide range of settings and environments and may not conform to a single reward function. This proves detrimental to current methods which might over-generalize and cause poor performance. In forward RL, decomposing a policy into smaller specialized policy options has been shown to improve results for exactly such cases [67, 4]. Thus, we extend the options framework to IRL and decompose both the reward function and policy, allowing our method to learn deep policies which can specialize to the set of best-fitting experts. As opposed to the previous body of work mentioned in Section 3.3, in our formulation the latent structure is implicitly encoded in an unsupervised manner. This is inherently accomplished while learning to solve a task composed of a wide range of underlying reward functions and policies in a single framework. Overall, this work contains parallels to all of the aforementioned and other works emphasizing hierarchical policies [14, 16, 42], but specifically focuses on leveraging MoEs and reward decompositions to fit into the *options* framework for efficient one-shot domain adaptation in IRL.

To accomplish this, we build upon the Generative Adversarial Imitation Learning (GAIL) framework [28]. Unlike GAIL however, we do not assume knowledge of the expert actions as

we are interested in the Inverse Reinforcement Learning setting rather than Imitation Learning. We therefore rely solely on observations in the discriminator problem, and refer to our baseline approach as Generative Adversarial Inverse Reinforcement Learning (IRLGAN) as opposed to imitation learning. It is important to note that IRLGAN is GAIL without known actions, and we adopt a different naming scheme to highlight this difference. As such, our adversarial game optimizes:

$$\max_{\pi_\Theta} \min_{R_{\hat{\Theta}}} -[\mathbb{E}_{\pi_\Theta}[\log R_{\hat{\Theta}}(s)] + \mathbb{E}_{\pi_E}[\log(1 - R_{\hat{\Theta}}(s))]] \tag{3.1}$$

where π_Θ and π_E are the policy of the novice and expert parameterized by Θ and E, respectively, and $R_{\hat{\Theta}}$ is the discriminator probability that a sample state belongs to an expert demonstration (parameterized by $\hat{\Theta}$). We use this notation since in this case the discriminator approximates a reward function. Similarly to GAIL, we use TRPO during the policy optimization step for simple tasks. However, for complex tasks we adopt PPO. Figure 3.1 and Algorithm 1 show an outline for the general IRLGAN process.

We formulate using our method a way to learn joint reward-policy options with adversarial methods in IRL. As such, we call our method OptionGAN. This method can implicitly learn divisions in the demonstration state space and accordingly learn policy and reward options. Leveraging a correspondence between Mixture-of-Experts (MoE) and one-step options, we learn a decomposition of rewards and the policy-over-options in an end-to-end fashion. This decomposition is able to capture simple problems and learn any of the underlying rewards structure in one shot. This gives flexibility and benefits for a variety of future applications (both in reinforcement learning and standard machine learning).

We evaluate OptionGAN in the context of continuous control locomotion tasks, considering both simulated MuJoCo locomotion OpenAI Gym environments [8], modifications of these environments for task transfer [27], and a more complex Roboschool task [61]. We show that the final policies learned using joint reward-policy options outperform a single reward approximator and policy network in most cases, and particularly excel at one-shot domain adaptation.

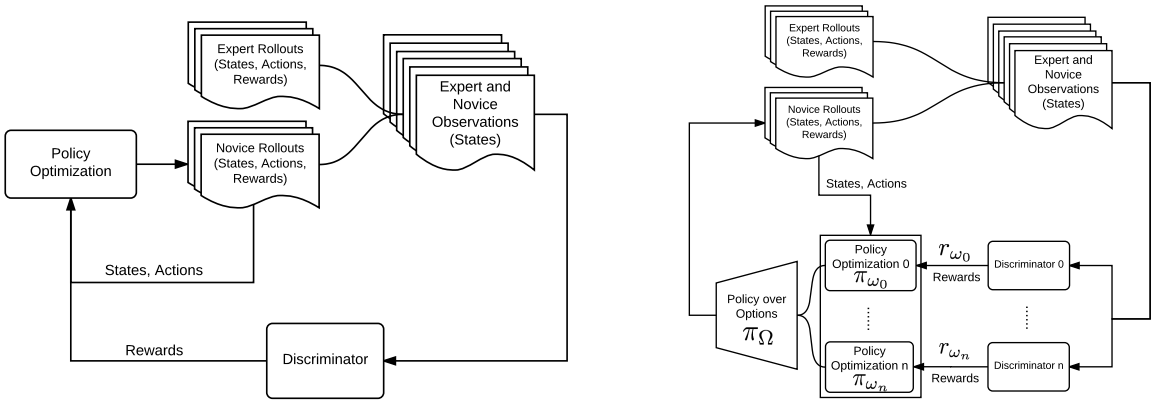

Figure 3.1: Generative Adversarial Inverse Reinforcement Learning (left) and OptionGAN (right) Architectures

3.5 Reward-Policy Options Framework

Based on the need to infer a decomposition of underlying reward functions from a wide range of expert demonstrations in one-shot domain adaptation, we extend the options framework for decomposing rewards as well as policies. In this way, intra-option policies, decomposed rewards, and the policy-over-options can all be learned in concert in a cohesive framework.

In this case, an option is formulated by a tuple: $(I_\omega, \pi_\omega, \beta_\omega, r_\omega)$. Here, r_ω is a reward option from which a corresponding intra-option policy π_ω is derived. That is, each policy option is optimized with respect to its own local reward option. The policy-over-options not only chooses the intra-option policy, but the reward option as well: $\pi_\Omega \rightarrow (r_\omega, \pi_\omega)$. For simplicity, we refer to the policy-over-reward-options as r_Ω (in our formulation, $r_\Omega = \pi_\Omega$). There is a parallel to be drawn from this framework to Feudal RL [15], but here the intrinsic reward function is statically bound to each worker (policy option), whereas in that framework the worker dynamically receives a new intrinsic reward from the manager.

To learn joint reward-policy options, we make use of the IRLGAN framework. We reformulate the discriminator as a Mixture-Of-Experts and re-use the gating function when learning a set of policy options. We show that by properly formulating the discriminator loss function, the Mixture-Of-Experts converges to one-step options. In addition, this formulation allows us

Input : Expert trajectories $\tau_E \sim \pi_E$.
Initialize $\Theta, \hat{\Theta}$
for $i = 0, 1, 2, \ldots$ **do**
  Sample trajectories $\tau_N \sim \pi_{\Theta_i}$
  Update discriminator parameters ($\hat{\Theta}$) according to:

$$L_{\hat{\Theta}} = \mathbb{E}_{s\sim\tau_N}[\log R_{\hat{\Theta}}(s)] + \mathbb{E}_{s\sim\tau_E}[\log(1 - R_{\hat{\Theta}}(s))]$$

  Update policy (with constrained update step and parameters θ) according to:

$$\mathbb{E}_{\tau_N}[\nabla_\Theta \log \pi_{\Theta_i}(a|s)\mathbb{E}_{\tau_N}[\log(R_{\hat{\Theta}_{i+1}}(s))|s_0 = \bar{s}]]$$

end

Algorithm 1: IRLGAN

to use regularizers which encourage distribution of information, diversity, and sparsity in both the reward and policy options.

3.6 Learning Joint Reward-Policy Options

The use of one-step options allows us to learn a policy-over-options in an end-to-end fashion as a Mixture-of-Experts formulation. In the one-step case, selecting an option ($\pi_{\omega,\theta}$) using the policy-over-options ($\pi_{\Omega,\zeta}$) can be viewed as a mixture of completely specialized experts such that: $\pi_\Theta(a|s) = \sum_\omega \pi_{\Omega,\zeta}(\omega|s)\pi_{\omega,\theta}(a|s)$.

The reward for a given state is composed as: $R_{\Omega,\hat{\Theta}}(s) = \sum_\omega \pi_{\Omega,\zeta}(\omega|s)r_{\omega,\hat{\theta}}(s)$, where $\zeta, \theta \in \Theta, \hat{\theta} \in \hat{\Theta}$ are the parameters of the policy-over-options, policy options, and reward options, respectively. Thus, we reformulate our discriminator loss as a weighted mixture of completely specialized experts in Eq. 3.2. This allows us to update the parameters of the policy-over-options and reward options together during the discriminator update.

$$L_\Omega = \mathbb{E}_\omega \left[\pi_{\Omega,\zeta}(\omega|s)L_{\hat{\theta},\omega}\right] + L_{reg} \tag{3.2}$$

Here, $L_{\hat{\theta},\omega}$ is the sigmoid cross-entropy loss of the reward options (discriminators). L_{reg}, as

will be discussed later on, is a penalty or set of penalties which can encourage certain properties of the policy-over-options or the overall reward signal. As can be seen in Algorithm 2 and Figure 3.1, this loss function can fit directly into the IRLGAN framework.

Input : Expert trajectories $\tau_E \sim \pi_E$.
Initialize $\theta, \hat{\theta}$
for $i = 0, 1, 2, \ldots$ **do**
  Sample trajectories $\tau_N \sim \pi_{\Theta_i}$
  Update discriminator options parameters $\hat{\theta}, \omega$ and policy-over-options parameters ζ, to minimize:

$$L_\Omega = \mathbb{E}_\omega \left[\pi_{\Omega,\zeta}(\omega|s) L_{\hat{\theta},\omega} \right] + L_{reg}$$

$$L_{\hat{\theta},\omega} = \mathbb{E}_{\tau_N}[\log r_{\hat{\theta},\omega}(s)] + \mathbb{E}_{\tau_E}[\log(1 - r_{\hat{\theta},\omega}(s))]$$

  Update policy options (with constrained update step and parameters $\theta_\omega \in \Theta_\Omega$) according to:

$$\mathbb{E}_{\tau_N}[\nabla_\theta \log \pi_\Theta(a|s) \mathbb{E}_{\tau_N}[\log(R_{\Omega,\hat{\Theta}}(s))|s_0 = \bar{s}]]$$

end

Algorithm 2: OptionGAN

Having updated the parameters of the policy-over-options and reward options, standard PG methods can be used to optimize the parameters of the intra-option policies. This can be done by weighting the average of the intra-option policy actions with the policy-over-options $\pi_{\Omega,\zeta}$. While it is possible to update each intra-option policy separately as in [4], this Mixture-of-Experts formulation is equivalent, as discussed in the next section. Once the gating function specializes over the options, all gradients except for those related to the intra-option policy selected would be weighted by zero. We find that this end-to-end parameter update formulation leads to easier implementation and smoother learning with constraint-based methods.

3.7 Mixture-of-Experts as Options

To ensure that our MoE formulation converges to options in the optimal case, we must properly formulate our loss function such that the gating function specializes over experts. While it may be possible to force a sparse selection of options through a top-k choice as in [63], we find that this leads to instability since for $k = 1$ the top-k function is not differentiable. As is specified in [31], a loss function of the form $L = (y - \frac{1}{||\Omega||} \sum_\omega \pi_\Omega(\omega|s) y_\omega(s))^2$ draws cooperation be-

tween experts, but a reformulation of the loss, $L = \frac{1}{||\Omega||} \sum_{\omega} \pi_{\Omega}(\omega|s)(y - y_{\omega}(s))^2$, encourages specialization.

If we view our policy-over-options as a softmax (i.e. $\pi_{\Omega}(\omega|s) = \frac{\exp(z_{\omega}(s))}{\sum_i \exp(z_i(s))}$), then the derivative of the loss function with respect to the gating function becomes:

$$\frac{dL}{dz_{\omega}} = \frac{1}{||\Omega||} \pi_{\Omega}(\omega|s) \left((y - y_{\omega}(s))^2 - L \right) \tag{3.3}$$

This can intuitively be interpreted as encouraging the gating function to increase the likelihood of choosing an expert when its loss is less than the average loss of all the experts. The gating function will thus move toward deterministic selection of experts.

As we can see in Eq. 3.2, we formulate our discriminator loss in the same way, using each reward option and the policy-over-options as the experts and gating function respectively. This ensures that the policy-over-options specializes over the state space and converges to a deterministic selection of experts. Hence, we can assume that in the optimal case, our formulation of an MoE-style policy-over-options is equivalent to one-step options. Our characterization of this notion of MoE-as-options is further backed by experimental results. Empirically, we still find temporal coherence across option activation despite not explicitly formulating call-and-return options as in [4].

3.7.1 Regularization Penalties

Due to our formulation of Mixture-of-Experts as options, we can learn our policy-over-options in an end-to-end manner. This allows us to add additional terms to our loss function to encourage the appearance of certain target properties.

Sparsity and Variance Regularization

To ensure an even distribution of activation across the options, we look to conditional computation techniques that encourage sparsity and diversity in hidden layer activations and apply these to our policy-over-options [6]. We borrow three penalty terms L_b, L_e, L_v (adopting a similar notation). In the minibatch setting, these are formulated as:

3.7 Mixture-of-Experts as Options

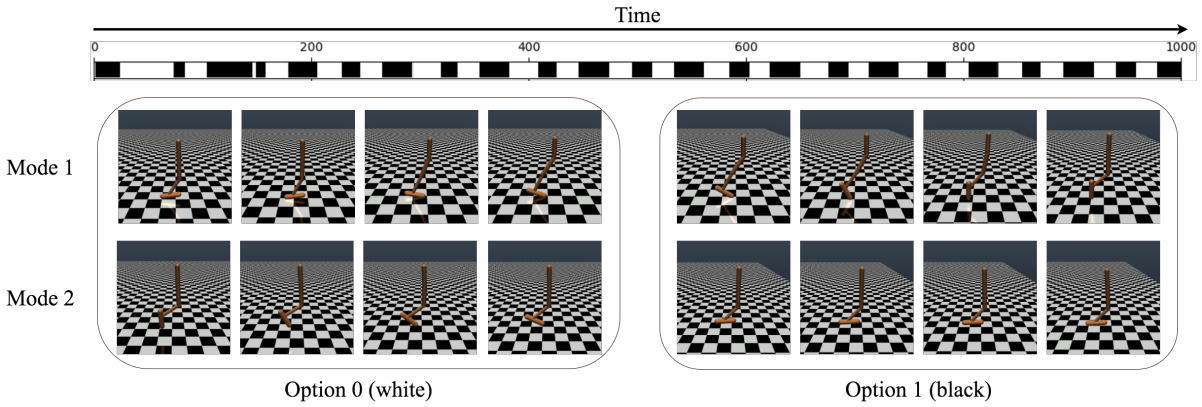

Figure 3.2: The policy-over-options elicits two interpretable behaviour modes per option, but temporal cohesion and specialization is seen between these behaviour modes across time within a sample rollout trajectory.

$$L_b = \sum_{\omega} ||\mathbb{E}_s[\pi_\Omega(\omega|s)] - \tau||_2 \tag{3.4}$$

$$L_e = \mathbb{E}_s\left[|| \left(\frac{1}{||\Omega||} \sum_{\omega} \pi_\Omega(\omega|s) \right) - \tau||_2 \right] \tag{3.5}$$

$$L_v = -\sum_{\omega} var_\omega\{\pi_\Omega(\omega|s)\} \tag{3.6}$$

where τ is the target sparsity rate (which we set to $.5$ for all cases). Here, L_b encourages the activation of the policy-over-options with target sparsity τ "in expectation over the data" [6]. Essentially, L_b encourages a uniform distribution of options over the data while L_e drives toward a target sparsity of activations per example (doubly encouraging our mixtures to be sparse). L_v also encourages varied π_Ω activations while discouraging uniform selection.

Mutual Information Penalty

To ensure the specialization of each option to a specific partition of the state space, a mutual information (MI) penalty is added.[1] We thus minimize mutual information pairwise between option distributions, similarly to [36]:

$$I(F_i; F_j) = -\frac{1}{2}\log(1 - \rho_{ij}^2), \tag{3.7}$$

where F_i and F_j are the outputs of reward options i and j respectively, and ρ_{ij} the correlation coefficient of F_i and F_j, defined as $\rho_{ij} = \frac{E[(F_i - E[F_i])(F_j - E[F_j])]}{\sigma_i^2 \sigma_j^2}$.

The resulting loss term is thus computed as:

$$L_{\text{MI}} = \sum_{\omega \in \Omega} \sum_{\hat{\omega} \in \Omega, \omega \neq \hat{\omega}} I(\pi_\omega, \pi_{\hat{\omega}}). \tag{3.8}$$

Thus the overall regularization term becomes:

$$\mathcal{L}_{reg} = \lambda_b L_b + \lambda_e L_e + \lambda_v L_v + \lambda_{\text{MI}} L_{\text{MI}}. \tag{3.9}$$

3.8 Experiments

Task	Expert	IRLGAN	OptionGAN (2ops)	OptionGAN (4ops)
Hopper-v1	3778.8 ± 0.3	**3736.3 ± 152.4**	*3641.2 ± 105.9*	*3715.5 ± 17.6*
HalfCheetah-v1	4156.9 ± 8.7	3212.9 ± 69.9	**3714.7 ± 87.5**	*3616.1 ± 127.3*
Walker2d-v1	5528.5 ± 7.3	*4158.7 ± 247.3*	*3858.5 ± 504.9*	**4239.3 ± 314.2**
HopperSimpleWall-v0	3218.2 ± 315.7	*2897.5 ± 753.5*	*3140.3 ± 674.3*	**3272.3 ± 569.0**
RoboschoolHumanoidFlagrun-v1	2822.1 ± 531.1	*1455.2 ± 567.6*	*1868.9 ± 723.7*	**2113.6 ± 862.9**

Table 3.1: True Average Return for simple and complex experiments, with the standard error across 10 trials on the 25 final evaluation rollouts using the final policy.

To evaluate our method of learning joint reward-policy options, we first investigate continuous control tasks. We divide our experiments into 3 settings: simple locomotion tasks, one-shot

[1] While it may be simpler to use an entropy regularizer, we found that in practice it performs worse. Entropy regularization encourages exploration [43]. In the OptionGAN setting, this results in unstable learning, while the mutual information term encourages diversity in the options while providing stable learning.

domain adaptation, and complex tasks. We compare OptionGAN against IRLGAN in all scenarios, investigating whether dividing the reward and policy into options improves performance against the single approximator case.[2] Table 3.1 shows the overall results of our evaluations and we highlight a subset of learning curves in Figure 3.3. We find that in nearly every setting, the final optionated policy learned by OptionGAN outperforms the single approximator case.

3.8.1 Experimental Setup

For all evaluation runs, hold all shared hyperparameters constant for both IRLGAN and OptionGAN, and average evaluations across 10 trials, each using a different random seed. We use the average return of the true reward function across 25 sample rollouts as the evaluation metric.

Multilayer perceptrons are used for all approximators as in [28]. For the OptionGAN intra-option policy and reward networks, we use shared hidden layers to ensure a fair comparison against a single network of the same number of hidden layers. That is, $r_\omega, \forall \omega \in \Omega$, and $\pi_\omega, \forall \omega \in \Omega$ each share hidden layers among themselves. The policy-over-options π_Ω use separate parameters however.

For simple settings, all hidden layers are of size $(64, 64)$, while for complex experiments we use $(128, 128)$ hidden layers. In the 2-options case we set $\lambda_e = 10.0, \lambda_b = 10.0, \lambda_v = 1.0$ based on a simple hyperparameter search and reported results from [6]. Finally, in the 4-options case we relax the regularizer that encourages a uniform distribution of options (L_b), setting $\lambda_b = .01$.

3.8.2 Simple Tasks

First, we investigate simple settings for a set of benchmark locomotion tasks provided in OpenAI Gym [8] using the MuJoCo simulator [69]. We use the Hopper-v1, HalfCheetah-v1, and Walker2d-v1 locomotion environments. The results of this experiment are shown in Table 3.1 and sample learning curves for Hopper and HalfCheetah can be found in Figure 3.3. For this setting, we use as demonstrations 10 expert rollouts from a policy trained using TRPO for 500 iterations.

In these simple tasks, OptionGAN converges to policies which perform as well or better than

[2] Extended experimental details and results can be found in the supplemental. Code is located at: https://github.com/Breakend/OptionGAN.

the single approximator setting. Importantly, even in these simple settings, the options which our policy selects have a notion of temporal coherence and interpretability despite not explicitly enforcing this in the form of a termination function. This can be seen in the two option version of the Hopper-v1 task in Figure 3.2. We find that generally each option takes on two behaviour modes. The first option handles: (1) the rolling of the foot during hopper landing; (2) the folding in of the foot in preparation for floating. The second option handles: (1) the last part of take-off where the foot is hyper-extended and body flexed; (2) the part of air travel without any movement.

3.8.3 Complex Tasks

Next, we investigate slightly more complex tasks. We utilize the HopperSimpleWall-v0 environment provided by the gym-extensions framework [27] and the RoboschoolHumanoidFlagrun-v1 environment used in [61]. In the first, a wall is placed randomly in the path of the Hopper-v1 agent and simplified sensor readouts are added to the observations as in [71]. In the latter, the goal is to run and reach a frequently changing target. This is an especially complex task with a highly varied state space. In both cases we use an expert trained with TRPO and PPO respectively, to generate 40 expert rollouts. For the Roboschool environment, we find that TRPO does not allow enough exploration to perform adequately, and thus we switch our policy optimization method to the clipping-objective version of PPO.

Task	IRLGAN	OptionGAN (2ops)	OptionGAN (4ops)
Hopper-v1 (1 demo)	3636.55±72.13	3595.38±127.67	3687.03±23.49
Hopper-v1 (3 demos)	3192.96±354.50	**3757.19±16.17**	3428.16±211.81
Hopper-v1 (5 demos)	3299.126±269.06	3697.98±30.43	3608.63±34.63
Hopper-v1 (10 demos)	**3736.3 ± 152.4**	**3641.2 ± 105.9**	**3715.5 ± 17.6**
HalfCheetah-v1 (1 demo)	2547.32±252.00	2360.68±350.80	528.43±308.92
HalfCheetah-v1 (3 demos)	2755.71±328.53	2396.25±327.09	658.57±409.37
HalfCheetah-v1 (5 demos)	2876.18±146.33	2380.69±270.60	298.13±422.18
HalfCheetah-v1 (10 demos)	**3212.9 ± 69.9**	**3714.7 ± 87.5**	**3616.1 ± 127.3**
Walker2d-v1 (1 demo)	3808.98±341.89	3050.25±314.44	2878.54±204.77
Walker2d-v1 (3 demos)	4067.27±195.46	3547.44±181.29	2673.93±140.80
Walker2d-v1 (5 demos)	3860.87±188.67	3789.80±259.32	2911.82±184.98
Walker2d-v1 (10 demos)	**4158.7 ± 247.3**	**3858.5 ± 504.9**	**4239.3 ± 314.2**

Table 3.2: True Average Return for expert demonstration ablation experiments, with the standard error across 10 trials on the 25 final evaluation rollouts using the final policy.

3.8.4 Simple Tasks with Few Demonstrations

Next we investigate the effect of having fewer expert demonstrations on both IRLGAN and OptionGAN by cutting down the amount of demonstrations available to the algorithms, on the same locomotion tasks. We evaluate each algorithm in the case of 1, 3, and 5 provided expert rollouts. The results of this data shortage scenario are shown in Table 3.2.

We find that both algorithms suffer in performance, to varying degrees, with fewer provided expert demonstrations. Intuitively, if only few demonstrations are given, the discriminator can 'memorize' more easily the state space the expert occupies and obtain a higher classification accuracy early on. However in this adversarial context, the samples from the learning agent getting classified as expert samples by the discriminator act as the reward signal and therefore, as desirable states to move towards. With the classification task being more clear cut with less expert samples, the learning agent is left with less of a guiding signal and therefore suffers from poorer performance.

3.8.5 One-Shot Domain Adaptation

Task	Expert	IRLGAN	OptionGAN (2ops)	OptionGAN (4ops)
Hopper (One-Shot)	3657.7 $\pm$ 25.4	2775.1 $\pm$ 203.3	*3409.4 $\pm$ 80.8*	**3464.0 $\pm$ 67.8**
HalfCheetah (One-Shot)	4156.9 $\pm$ 51.3	1296.3 $\pm$ 177.8	1679.0 $\pm$ 284.2	**2219.4 $\pm$ 231.8**
Walker (One-Shot)	4218.1 $\pm$ 43.1	3229.8 $\pm$ 145.3	**3925.3 $\pm$ 138.9**	*3769.40 $\pm$ 170.4*

Table 3.3: True Average Return for one-shot domain adaptation experiments, with the standard error across 10 trials on the 25 final evaluation rollouts using the final policy.

Finally, we investigate one-shot domain adaptation scenarios. In these cases, the novice is trained on a target environment, while expert demonstrations come from a similar task, but from a set of source environments with altered dynamics (i.e. one-shot transfer from varied expert demonstrations to a new environment). To demonstrate the effectiveness of OptionGAN in this domain adaptation task, we use expert demonstrations from environments with varying gravity conditions as seen in [27, 12]. For each environment, we vary the gravity (0.5, 0.75, 1.25, 1.5 $\times$ Earth's gravity) and train experts in each gravity variation using TRPO. We gather 10 expert trajectories from each gravity variation, for a total of 40 expert rollouts, to train a novice agent on the normal Earth gravity environment (the unmodified environment as provided in OpenAI Gym), and repeat this for Hopper-v1, HalfCheetah-v1, and Walker2D-v1.

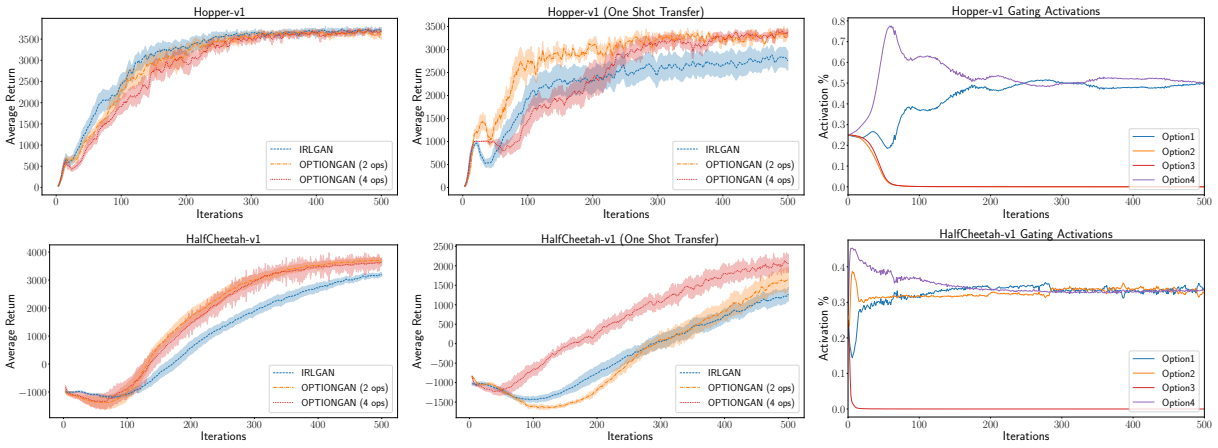

Figure 3.3: Left Column: Simple locomotion curves. Error bars indicate standard error of average returns across 10 trials on 25 evaluation rollouts. Middle Column: One-shot transfer experiments with 40 expert demonstrations from varied gravity environments without any demonstrations on the novice training environment training on demonstrations from .5G, .75G, 1.25G, 1.5G gravity variations. Right Column: Activations of policy-over-options over time with 4 options on training samples in the one-shot transfer setting with $\lambda_b = .01$.

These gravity tasks are selected due to the demonstration in [27] that learning sequentially on these varied gravity environments causes catastrophic forgetting of the policy on environments seen earlier in training, which suggests that the dynamics are varied enough that trajectories are difficult to generalize across, yet still share some state representations and task goals.

As seen in Figure 3.3, using options can cause significant performance increases in this area, but performance gains can vary across the number of options and the regularization penalty as seen in the regular experiments (Table 3.1).

3.9 Ablation Investigations

Convergence of Mixtures to Options

To show that our formulation of Mixture-of-Experts decomposes to options in the optimal case, we investigate the distributions of our policy-over-options. We find that across 40 trials, 100% of activations fell within a reasonable error bound of deterministic selection across 1M samples. That is, in 40 total trials across 4 environments (Hopper-v1, HalfCheetah-v1, Walker2d-v1,

3.9 Ablation Investigations

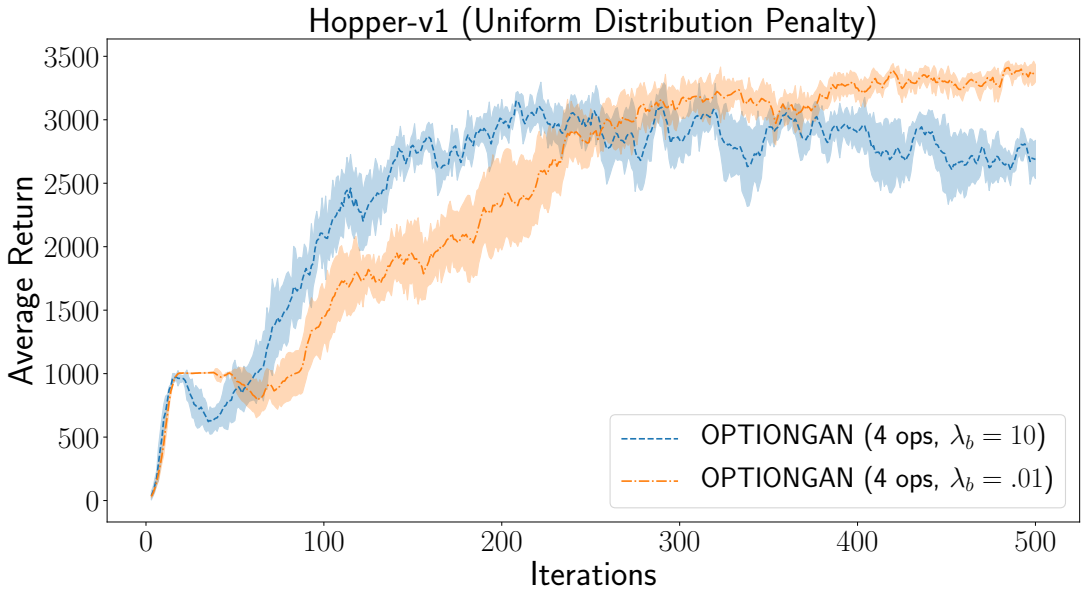

Figure 3.4: Effect of uniform distribution regularizer. Average π_Ω across final sample novice rollouts: $\lambda_b = 10.0$, $[.27, .21, .25, .25]$; $\lambda_b = .01$, $[0., 0., .62, .38]$.

RoboschoolHumanoidFlagrun-v1), policies were trained for 500 iterations (or 5k iterations in the case of RoboschoolHumanoidFlagrun-v1). We collected 25k samples at the end of each trial. Among the gating activations across the samples, we recorded the number of gating activations within the range $\{0 + \epsilon, 1 - \epsilon\}$ for $\epsilon = 0.1$. 100% fell within this range. 98.72% fell within range $\epsilon = 1^{-3}$. Thus at convergence, both intuitively and empirically we can refer to our gating function over experts as the policy-over-options and each of the experts as options.

Effect of Uniform Distribution Regularizer

We find that forcing a uniform distribution over options can potentially be harmful. This can be seen in the experiment in Figure 3.4, where we evaluate the 4 option case with $\lambda_b = \{0.1, 10\}$. However, relaxing the uniform constraint results in rapid performance increases, particularly in the HalfCheetah-v1 where we see increases in learning speed with 4 options.

There is an intuitive explanation for this. In the 4-option case, with a relaxed uniform distribution penalty, we allow options to drop out during training. In the case of Hopper and Walker tasks, generally 2 options drop out slowly over time, but in HalfCheetah, only one option drops

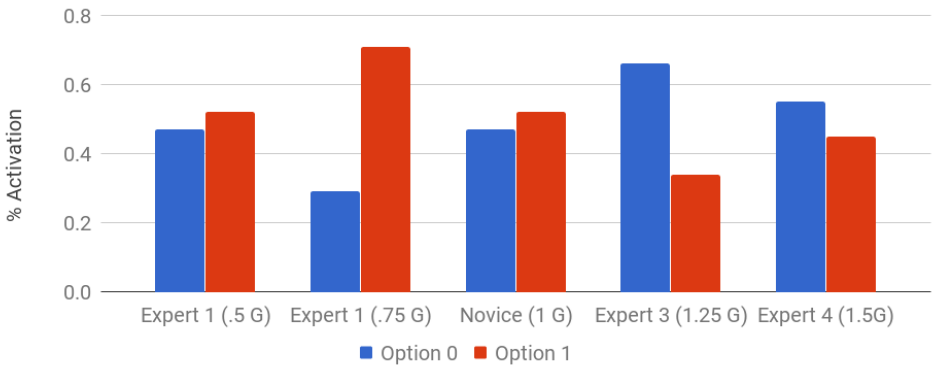

Figure 3.5: Probability distribution of π_Ω over options on expert demonstrations. Inherent structure is found in the underlying demonstrations. The .75G demonstration state spaces are significantly assigned to Option 1 and similarly, the 1.25G state spaces to Option 0.

out in the first 20 iterations with a uniform distribution remaining across the remaining options as seen in Figure 3.3. We posit that in the case of HalfCheetah there is enough mutually exclusive information in the environment state space to divide across 3 options, quickly causing a rapid gain in performance, while the Hopper tasks do not settle as quickly and thus do not see that large gain in performance.

Latent Structure in Expert Demonstrations

Another benefit of using options in the IRL transfer setting is that the underlying latent division of the original expert environments is learned by the policy-over-options. As seen in Figure 3.5, the expert demonstrations have a clear separation among options. We suspect that options further away from the target gravity are not as specialized due to the fact that their state spaces are covered significantly by a mixture of the closer options (see supplemental material for supporting projected state space mappings). This indicates that the policy-over-options specializes over the experts and is thus inherently beneficial for use in one-shot domain adaptation.

3.10 Conclusion

We propose a direct extension of the options framework by adding joint reward-policy options. We learn these options in the context of generative adversarial inverse reinforcement learning

and show that this method outperforms the single policy case in a variety of tasks – particularly in domain adaptation settings. Furthermore, the learned options demonstrate temporal and interpretable cohesion without specifying a call-and-return termination function.

Our formulation of joint reward-policy options as a Mixture Of Experts allows for: potential upscaling to extremely large networks as in [63], reward shaping in forward RL, and using similarly specialized MoEs in generative adversarial networks. Our optionated networks form an effective and extendable framework. They capture the problem structure effectively, which allows strong generalization in one-shot domain adaptation scenarios. Moreover, as adversarial methods are now commonly used across a myriad of communities, the embedding of options within this methodology is an excellent delivery mechanism to exploit the benefits of hierarchical RL in many new fields.

Chapter 4

Conclusion & Future Work

4.1 Summary

In this thesis, we presented two algorithms with applications in robotic, both of which we show are negatively affected by data shortages. Using domain adaptation methods, we tried to alleviate these data shortage scenarios, using synthetic data in the case of underwater visual tracking and data originating from various other domains in the case of agents learning from demonstrations.

In Chapter 2, we presented a general *tracking-by-detection* approach that reliably tracks targets without engineered markers, making use of recent advances in deep learning for deployment onboard robots with limited computing capabilities. We compared the applicability of multiple lightweight neural network architecture variants, and showed improvements made possible by recurrent extensions of these networks, using several desirable metrics for visual convoying tasks. We also investigated the performance of our most promising architecture in data scarce settings, using realistic simulated visual data generated in Unreal Engine. In addition, we successfully performed open sea multi-robot underwater convoying using a limited amount of annotated training data.

In Chapter 3, we proposed a direct extension of the options framework by adding joint reward-policy options. By making use of multiple discriminators in the context of generative adversarial inverse reinforcement learning, we learn the formulated reward-policy options and

show that our method outperforms the single discriminator and policy case in most attempted tasks. We show that the options learned using our algorithm demonstrate temporal and interpretable cohesion without having explicitly specified an option termination function. Furthermore, through expert demonstration ablation experiments, we show that the studied algorithms suffer in data poor scenarios and demonstrate the advantageous domain adaptation capabilities of our method which leverages joint reward-policy options.

4.2 Future Work

In the visual tracking domain, the temporal-based bounding box detection pipeline could be improved by making the entire architecture trainable end-to-end. Additionally, to make this approach robust to contexts with more than a pair of robots, it could be extended to visual servoing with multiple bounding boxes per frame. Additionally, by using stronger predictive models, tracking could be made more robust to tracking interruptions. Using our developed Unreal Engine-based robot simulator, the 3D pose data of the robot can easily be extracted, and extensions to 6D tracking-by-detection (pose and position) of the robot are thus possible. Extensions of our framework to underwater gesture-based visual communication for convoying behaviours is also possible, by specifying robot poses as communication tokens.

In learning from demonstrations, our optionated policy-reward formulation using MoEs could be upscaled to extremely large networks as in [63], reward shaping in forward RL, or ported for use in model-based RL. Furthermore, while adversarial methods are now used across an increasing number of fields within machine learning, the use of multiple discriminators is still rare, and the use of similarly specialized MoEs in generative adversarial networks could prove useful to such fields. Our optionated networks capture the problem structure effectively, which allows strong generalization in one-shot domain adaptation scenarios. Using these networks in other domain adaptation scenarios, with data from "further" domains, such as Sim2Real experiments for instance could prove beneficial.

Finally, recent work in Sim2Real scenarios have used domain randomization [68, 49], where the environment characteristics for a task are stochastically changed as the agent learns the task, as a viable strategy to improve the domain adaptation abilities of the agent. This method is general enough to be applicable to both the visual tracking and learning from demonstration

domains, and as such, would be a good candidate to improve our results overall.

List of Publications

Published:

- Peter Henderson, **Wei-Di Chang**, Pierre-Luc Bacon, David Meger, Joelle Pineau, and Doina Precup. "OptionGAN: Learning Joint Reward-Policy Options using Generative Adversarial Inverse Reinforcement Learning." In *AAAI*. 2018.
- Peter Henderson, **Wei-Di Chang**, Florian Shkurti, Johanna Hansen, David Meger, and Gregory Dudek. "Benchmark Environments for Multitask Learning in Continuous Domains." In *ICML Lifelong Learning: A Reinforcement Learning Approach Workshop*. 2017.
- Florian Shkurti, **Wei-Di Chang**, Peter Henderson, Md Jahidul Islam, Juan Camilo Gamboa Higuera, Jimmy Li, Travis Manderson, Anqi Xu, Gregory Dudek, and Junaed Sattar. "Underwater multi-robot convoying using visual tracking by detection." In *IROS*. 2017.

Bibliography

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, pages 1–, New York, NY, USA, 2004. ACM.

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

[3] M. Babes, V. Marivate, K. Subramanian, and M. L. Littman. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 897–904, 2011.

[4] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *AAAI*, pages 1726–1734, 2017.

[5] J. A. Bagnell. An invitation to imitation. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 2015.

[6] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

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

[8] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym, 2016.

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

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

[11] J. Choi and K. eung Kim. Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 305–313. Curran Associates, Inc., 2012.

[12] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

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

[14] C. Daniel, G. Neumann, and J. R. Peters. Hierarchical relative entropy policy search. In *International Conference on Artificial Intelligence and Statistics*, pages 273–281, 2012.

[15] P. Dayan and G. E. Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–278, 1993.

[16] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

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

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

[19] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[20] C. Fries and H.-J. Wuensche. Monocular template-based vehicle tracking for autonomous convoy driving. In *Intelligent Robots and Systems*, pages 2727–2732. IEEE, 2014.

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

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

[23] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

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

[25] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. *arXiv preprint arXiv:1705.10479*, 2017.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[27] P. Henderson, W.-D. Chang, F. Shkurti, J. Hansen, D. Meger, and G. Dudek. Benchmark environments for multitask learning in continuous domains. *ICML Lifelong Learning: A Reinforcement Learning Approach Workshop*, 2017.

[28] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

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

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

[31] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

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

[33] S. Krishnan, A. Garg, R. Liaw, L. Miller, F. T. Pokorny, and K. Goldberg. Hirl: Hierarchical inverse reinforcement learning for long-horizon tasks with delayed rewards. *arXiv preprint arXiv:1604.06508*, 2016.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[35] Y. Li, J. Song, and S. Ermon. InfoGAIL: Interpretable imitation learning from visual demonstrations. *arXiv preprint arXiv:1703.08840*, 2017.

[36] Y. Liu and X. Yao. Learning and evolution by minimization of mutual information. In *International Conference on Parallel Problem Solving from Nature*, pages 495–504. Springer, 2002.

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

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

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

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

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

[42] J. Merel, Y. Tassa, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

[43] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[44] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

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

[46] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

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

[48] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv:1710.06537*, 2017.

[49] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

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

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

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

[53] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

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

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

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

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

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

[59] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015.

[60] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[61] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[62] P. Sermanet, K. Xu, and S. Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016.

[63] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[64] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

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

[66] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[67] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[68] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017.

[69] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 5026–5033, 2012.

[70] H. van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang. Hybrid reward architecture for reinforcement learning. *arXiv preprint arXiv:1706.04208*, 2017.

[71] Z. Wang, J. Merel, S. Reed, G. Wayne, N. de Freitas, and N. Heess. Robust imitation of diverse behaviors. *arXiv preprint arXiv:1707.02747*, 2017.

[72] Q. Yu, T. B. Dinh, and G. Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers. In *10th European Conference on Computer Vision: Part II*, pages 678–691, 2008.

[73] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, pages 1433–1438. AAAI Press, 2008.