Domain Invariant and Semantic Aware Visual Servoing

Fereshteh Sadeghi

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Emanuel Todorov, Chair

Steven M. Seitz

C. Lawrence Zitnick

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Domain Invariant and Semantic Aware Visual Servoing

Fereshteh Sadeghi

Chair of the Supervisory Committee:
Professor Emanuel Todorov
Computer Science and Engineering

Robots should understand both semantics and physics in order to be able to make meaningful interactions in the real world. Vision is one of the primary modalities for us humans to learn and reason about our world. Equipping robots with semantic visual understanding can increase their versatility and realizes adaptable interaction using visual feedback. In this thesis, we investigate how to teach robots learn generalizable skills for diverse real world vision-based tasks that incorporate semantics while human effort and intervention is minimized. To this end, we address major challenging problems brought by real-world constraints for teaching highly generalizable, versatile and semantic aware vision-based robot policies in low cost and safe manner. We propose domain invariant visual servoing for both manipulation and navigation that enables seamless and direct transfer of vision-based robot policies from simulation to the real world. By introducing simulation randomization (domain randomization) we make it possible to collect large volumes of robot data in a low cost and safe fashion. Additionally, we devise techniques that incorporate spatio-temporal semantic visual reasoning on RGB images to train highly generalizable and versatile vision-based robot policies in 3D simulation. More concretely, we investigate several research questions in this thesis: (1) How can learning algorithms be used to enable machines gain visual semantic understanding? (2) How can we learn robotic skills safely and how we can equip robots with visual semantics to make them capable of doing diverse tasks while requiring small amount of robot data? (3) How can we enable our robots/machines to adapt to new unknown environments and situations? I will propose several first steps towards answering these questions in this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

"One only understands the things that one tames," said the fox. "Men have no more time to understand anything. They buy things all ready made at the shops. But there is no shop anywhere where one can buy friendship, and so men have no friends any more. If you want a friend, tame me."

Antoine de Saint-Exupry, *The Little Prince*

There are so many people that I met during my PhD journey from whom I learned a great many things. Just to name a few, I learned to be brave and ambitious in choosing what I am doing, I learned to be incentivized by each and every failure to find ways and solutions to make things work and eventually to get to believe that there is no impossible. I learned to be patient and persistent in all aspects of my research and to be sharp in figuring out the details and discovering the hidden states. And, I got to realize that the secret key to open all closed doors is to have a strong belief in my abilities. While it is not possible to mention all those amazing people here, I'd want to deeply thank *everyone* who taught me or was involved, in one way or another, into me learning new things during the ups and downs of this wonderful, educating and unforgettable journey.

I'd want to sincerely thank and appreciate the chair of my PhD dissertation committee Emanuel Todorov who genuinely supported me and my research without whom this thesis would have never been published. I'd like to deeply thank Emo for his unwavering guidance, patience, continual encouragement as well as for his thought provoking questions and

wise remarks during my defense and dissertation process.

I feel deep gratitude to Larry Zitnick who had been a great mentor and friend for me. When I started my PhD in University of Washington, Larry sparkled my interest in semantic reasoning and high level knowledge inference from visual sensory data. We continued to have fruitful discussions and I have over and over benefited from his insightful comments and advice along my PhD journey.

I am extremely grateful to Steve Seitz who supported me throughout my PhD studies at the University of Washington. I sincerely appreciate all his insightful advice, great support, generosity, and, above all, patience. I had the opportunity to have Steve in my PhD committee and benefit from his clever comments and his insightful feedbacks. Early in my PhD studies days, Steve encouraged me to pursue robotics, without his encouragement and continual appreciations I wouldn't have been in the place I am today; I'd want to express my deep gratitude to Steve because of his profound belief in my work and in my abilities. I'd also want to thank Steve Seitz for providing the greatest educating atmosphere and culture in the GRAIL lab by providing unique opportunities to meet with the giant minds of computer vision, graphics and machine learning though meetings, talks and legendary GRAIL retreats. I have learned many enlightening lessons by participating into GRAIL events and talking to the most elite researchers and receiving their valuable feedback.

Special thanks to Ed Lazowska and Hank Levy whose dedication and wisdom made UW CSE a warm and welcoming second home for me as well as for providing me with the "survival package" for grad school. Being part of the UW CSE family I learned countless invaluable lessons. I am grateful to have been part of UW CSE and will carry the great memories of legendary UW CSE ski days, Holiday skit parties and TGIF socials always with me. Many thanks to Lindsay Michimoto for getting me through the UW CSE, for her practical guidance and for her great help with administrative setups. Also, special thanks go to the "Dr. Computer Science and Engineering" for the most life-saving and out-of-the-box

pieces of advice.

I am glad and thankful to have had the opportunity to collaborate with amazing faculty and researchers during my PhD research. I have greatly benefited from collaborating with Ali Farhadi, Yejin Choi, Leonid Sigal and Sergey Levine. Also, thanks to Steve Tanimoto who was a great mentor for me. I benefited from Steve first by getting his supervision for teaching practices when I served as TA and guest lecturer in CSE473 and then from his insightful advice during my PhD.

I would like to thank GRAIL people from whom I learned how to seek for the holy grail in each and every research problem. I am also thankful to have unforgettable memories of GRAIL retreat at the longest of the long beaches, from the fun rainy beach walks and plays, to the sparkling dinner nights at the Depot. I am also deeply thankful of amazing research chats I had with Ross Girshick, Aaron Hertzman, Eli Schetman and Brian Russel, during their visits to UW CSE GRAIL retreats and later Berkeley BAIR retreats.

I deeply admire Ben Taskar and am especially grateful to had the great opportunity of knowing him. I have been extremely lucky to be in his acquaintances in a memorable road trip to Adrift at Long Beach to attend our very first GRAIL retreat. Perhaps, that was one of the most educating road trips I've ever had as Ben filled it with his ingenious ideas and remarks. I will always keep his words of wisdom, his deep insights and his brilliant ideas about computational models, probabilistic inference and deep learning.

I'd like to thank the big BAIR and the novice RAIL for hosting me during my time at UC Berkeley. I specially admire and am thankful of Pieter Abbeel and Jitendra Malik for the insightful discussions we had on vision, robots, control and cognition.

I thank my fellow labmates at UW CSE and UC Berkeley for the stimulating discussions, for the sleepless nights we were working together before deadlines and for all the fun we have had.

I am grateful to have had the opportunity to do an internship at Google Brain Robotics

in Vincent Vanhoucke's group. It has been a great pleasure and educating experience for me to collaborate with Alexander Toshev and to meet and interact with so many amazing people at Google.

I am very thankful to have experienced deep moments of glorious solitude during my time as a PhD student. Nothing could be more refreshing and mind sharpening for me than an afternoon run in beautiful Berkeley hills of the Tilden Regional park. I will never forget how splendid it was to spend many restless research nights at UW CSE614 decorated with deep coffee aroma and then watch the sun rising majestically over the sapphire blue waters of Lake Washington as seen from the Jaech Gallery. I will absolutely miss the joyful moments of spending time with amazing friends having coffee at the elegant Atrium of Paul G. Allen center. I started my PhD as a coffee lover and as an enthusiast in learning delicate latte art, then it happened that I got the most unique opportunity of making my own latte almost every day during my PhD career. Today, I am delighted and happy to have learned this beautiful and joyful latte art through practicing many moments of patience, care and balance. Here is a timelapse video of my progression towards becoming my own barista: https://youtu.be/5rF87UflP7s.

I am whole-heartedly thankful to Hamid Izadinia whose dedication, friendship and love is incomparable to anyone else I have met. With his beautiful mind and his cheerful face, Hamid has been the one who have given color, joy and meaning to each and every success or challenge I have experienced. We kicked off our PhD journey together with the longest of our [road] trips; we tasted, watched and felt new things, we pondered together and we envisioned our goals. Today, with even more and more excitements, I can't wait to explore countless many more adventures with him.

Last but not least, many thanks to my parents and my family, whose unconditional love, belief, patience and wisdom made me who I am today, I am thankful of all the moments of pure joy and deep thoughts I had with my family and I hope I will make them proud.

Chapter 1

INTRODUCTION

"Vision without action is merely a dream.
Action without vision just passes the time.
Vision with action can change the world. "

Joel A. Barker

Perception and motor control are the key capabilities that enable animals and human to perform complex tasks in the real world, such as exploring the environment (e.g. for reaching for food) and interacting with surrounding objects. Such scenarios require the intelligent agent to make high level inference about the semantics and physics of the world to recognize goals, distinguish walkable areas from the obstacles and to make decision for taking efficient actions and interact with objects to achieve the desired goal. Therefore, a stepping stone towards building machines that exhibit intelligent behavior is to investigate how we can teach them visually integrated motor skills. This, of course, has been the visionary goal of computer vision since it was born in the late 1960s as a subfield of Artificial Intelligence (AI).

Visual servoing is considered to be the fusion of computer vision, robotics and control in the classic robotics literature [43, 42]. The earliest works in the visual servoing date back to early 1970s were the more generic term of "visual feedback" was used to describe a robotic system that uses visual information inside a closed loop to position a robot end-effector [200, 175]. The term "visual servoing" was first appeared in 1979 by Hill and Park [81] and the first visual servoing systems were reported in 1980s [167, 199]. Around the same time, in 1986, the first computer vision book was written by Berthold Horn with the title of "Robot Vision" [86].

The development of visual servoing research field evidenced substantial growth during the 90s by the emergence of technological advances as well as photogrammetry techniques brought by

3-D computer vision in camera calibration, projective 3-D reconstruction and bundle adjustment theory [42, 89, 43, 191, 77, 60]. Accordingly, classic visual servoing systems were built upon "structure from motion" (SfM) and extraction of geometric features (e.g. key-points, lines, etc.). Following these advances, studying the relationship between robot controller and a vision system gained interest in the 90's.

More than two decades later, creating visually enabled robot controllers is still of great importance and interest [113, 144, 162, 114]. Integrating robotic controllers with vision is particularly important in low structured or unstructured environments where environment features vary constantly. Visual feedback is crucial in reducing the system sensitivity and improving robustness to calibration errors and perturbations. While computer vision techniques are designed to provide precise means to analyze the environment and describe observation, visual servoing aims to control a robot to manipulate its environment by a closed visual feedback loop.

In this thesis, we will be reconsidering the problem of "visual servoing" in light of the recent advances in deep learning and modern computer vision techniques. While, there have been notable advances in policy learning for goal-oriented agents using recent deep learning algorithms, there are still major challenges brought by real-world constraints for teaching highly generalizable and versatile robot policies in a cost efficient and safe manner. We show that a promising approach to learn highly generalizable robotic policies is to equip robots with semantic visual understanding learned through large volume of robot data. Conventional visual servoing mechanisms aim to acquire the capability to surf in the 3D world but they inherently do not incorporate object semantics. We argue that, if robots can recognize different semantic phenomena occurring within each of their observations, then they can depart from learning specialized policies for individual tasks and move towards learning generalist policies. This makes learned policies to be robust against noise and perturbations that often occurs in real world settings. However, a major challenge towards training policies with large amount of robotic data is that collecting such data is extremely infeasible in the real world. This motivated us to devise techniques for learning robot policies in simulation such that they can directly generalize to the real world.

In this thesis, we first study visual semantic reasoning based on RGB images. We then work to-

wards elaborating semantics and learning domain invariant visual servoing policies. We present our core contributions in the context of robotics navigation and manipulation problems: (a) Collision avoidance and semantic navigation of mobile robots. (b) Self-adaptive robotic manipulation based on visual feedback. In the following we explain an overview of our approach towards tackling the aforementioned problems and then highlight our contributions.

1.1 Self Supervised Visual Semantic Reasoning

Vision is one of the primary modalities for us humans to learn and reason about our world from the phenomena we evident everyday. Images and videos demonstrate scenes and events that implicitly carry a rich source of knowledge about semantic relationships between object. These semantic relations are governed by intrinsic and extrinsic object properties and in very simple words, common sense relations tell us how things are related and what are the compositionality rules between entities and leads us to infer high level information from the raw data. To enabled automatic discovery of high level semantics and common sense rules, we introduced VisKE (Visual Knowledge Extraction) for self supervised visual verification of relation phrases. VisKE assess the validity of a given verb-based relation phrase between common nouns [161]. The validity assessment is done by jointly analyzing over text and images and reasoning about the spatial consistency of the relative configurations of the entities. VisKE is scalable and flexible to be applied in intelligent machines as it does not require explicit human supervision for learning and inferring object relationships. This property which is a form of hypothesis assessment capability is particularity important for intelligent systems that need to make prediction and reasoning in the *absence of complete data*.

1.2 Collision Avoidance and Semantic Navigation for Mobile Robots

Robot navigation is an important yet challenging problem in robotics. On one hand, the mobile robot moves in continuous 3-D space and experiences various viewpoints while interacting with its world. This scenario makes a huge amount of visual data correlated with the continuous action space which is infeasible to be labeled. On the other hand, standard computer vision techniques (such as image classification or object detection) are known to work well with big labeled datasets

of static images seen during the training. To be functional, a mobile robot should have high generalization in recognizing scenes and objects so that it can be deployed in diverse and unstructured real world environments. In addition, the mobile robot should be able to identify the dangerous situations, such as crashing into an obstacle. To survive moving, it should also be capable of making right decisions to mitigate such dangerous situations.

Some of these challenges has lead into Learning from Demonstration(LfD) approaches with a human in the loop to collect supervised data. The supervised data collected to learn a model can be collected by demonstrating the robot directly or another means while mimicking the desired goal behavior of the robot. For example, in the context of autonomous flight, Abbeel et al. [5, 2] collected training data by an expert human pilot to demonstrate a helicopter with a wide range of complex aerobatic maneuvers. Giusti et al. [71] equipped a hiker with three head mounted cameras to collect data for training a turn classifier for outdoor forest trail following. To collect image data, the hiker then swiftly walks a long trail, by taking care of always looking straight along its direction of motion. However, LfD approaches are constrained by human effort to provide the data which is typically finite. But, deep learning algorithms work best when they are served with plentiful amount of data. As the result, the learned navigation policies may overfit to the limited number of scenarios collected by LfD and the generalization will be limited by how much data could be collected from humans which is a limiting factor. Also, critical and dangerous situations are usually missed from the data collected by LfD which leads into the distribution mismatch problem.

What about letting the agent to autonomously collect data and learn from its own experience? In that case, the agent can continuously improve its policy by learning from its own failures and successes. For example, in the indoor flight scenario, we let the drone fly. It may collide with things, and it will be penalized or it may move forward by which it will be rewarded. We let the drone continue doing this trial-and-error an enormous number of times until it eventually learns how to fly by reinforcement learning. While these seems to be a compelling approach, it is not quite feasible in the real world. Because, the trial and error approach for collecting training data requires the drone to crash many times which can result in damaging the drone itself and its training environment.

In fact, using reinforcement learning (RL) to learn control policies for indoor navigation especially with high-dimensional representations such as deep neural networks, presents a number of major challenges. First, RL tends to be data-intensive, making it difficult to use with safety critical platforms such as aerial vehicles [149], which have limited flight time and require time-consuming battery changes. Second, RL relies on trial-and-error, which means that, in order to learn to avoid collisions, the vehicle must experience at least a limited number of collisions during training. This can be extremely problematic for fragile robots such as quadrotors. Also, for goal driven semantic object navigation, the mobile agent should be able to understand both physics and semantics in order to reach various goal objects without colliding with obstacles. This blows up the number of different situations that can happen resulting in making the problem more complicated and more more data intensive.

Learning in simulation is a promising avenue to address the dangers of trial-and-error as well as data inefficiency for deep RL. In robotics, the use of simulators such as MuJoCo[1] [189] and Gazebo[2] are popular for learning control policies. The use of simulated data for training policies for real world applications has been popular since a long time ago. In the seminal work of Pomerleau in 1989, simulated images were used for training ALVIN which is one the first prototypes for self driving car [146]. However, the darker side of this story is that the models learned in simulation will suffer from a "reality gap". Transfer learning approaches use small amount of real data from the target domain for fine-tuning and joint training to address this problem. To address these challenges, we explore how to learn policies in simulation such that they can be directly applied in the real world.

In [162] we introduce the domain randomization as an orthogonal approach to conventional transfer learning approaches for learning policies in simulation such that they can generalize to real world. As compared to the global image-level photometric distortions that are applied to improve object detection in computer vision tasks, our randomization scheme, produces semantic object level visual diversity. We deployed our proposed simulation randomization technique (domain

[1] https://www.roboti.us

[2] http://gazebosim.org

randomization) for learning collision avoidance policies for indoor flight with a cheap drone called CAD^{2}RL. We developed a new deep reinforcement learning method based on Monte-Carlo return estimates for learning policies. Our policy which is entirely trained in simulation can successfully navigate a drone to open areas and avoid collisions in various real world scenarios.

While the collision avoidance policy can navigate the mobile robot to open spaces, it does not follow a specific goal. To develop a goal driven navigation policy that can generalize to diverse unseen real world situations without requiring real robot data, we propose a new method that incorporates visual semantics into policy learning . While robot platforms provide means for interacting with the physical world, robots cannot autonomously acquire object level semantics(e.g. chair, teddy-bear, etc) without needing human. At the same time, collecting robot trajectory data with dense semantic labels is practically infeasible. How can we teach robots perform real world tasks that incorporate semantics while human effort and intervention is minimized? To answer this question we devised Domain Invariant Visual Servoing (DIViS) method that learns semantics from static visual data while it learns physics correlated with the navigation task in simulation. Our quantitative and qualitative experimental evaluations show that our policy that is entirely trained in simulation can be directly deployed on a real robot and can generalize in novel and unstructured real-world environments.

1.3 Visual Self-Adaptive Robotic Manipulation

Another challenging problem in learning generalizable robot policies is to make robots capable of adapting to new and unknown settings specially when vision-based control is needed. Humans are remarkably proficient at rapidly adapting their behavior to novel situations. They can figure out how to work with different tools and adjust their body motion in response to dynamic changes. For example, people can instantly start digging a hole with shovels of various sizes or manipulate objects without needing to set their viewpoint to a fixed or specific pose. This capability, referred to as visual motor integration, is learned during childhood from manipulating objects in various situations, and is governed by a self-adaptation and mistake correction mechanism that uses rich sensory cues and vision as feedback.

The ability to quickly acquire visual motor control skills under large variations that require self-adaptive capability have substantial implications for autonomous robotic systems for example, this capability would be particularly desirable for robots that can help rescue efforts in emergency or disaster zones. However, current policy learning methods work well in test situations that approximately match those seen during training. To be able to tackle this challenging problem, we had to address the following essential questions: How can we make it feasible to provide the right amount of experience for the robot to learn the self-adaptation behavior based on pure visual observations that simulate a lifelong learning paradigm? How can we design a model that integrates robust perception and self-adaptive control such that it can quickly transfer to unseen environments?

For learning self-adaptation the agent can discover the action dependent transformations and apply the desired action to perform the task. To do this, we proposed to learn self-adaptation by an idea stemming from analogical reasoning. Analogies are extensively explored in cognitive science and explained by several theories: shared structure, shared abstraction, identity of relation, hidden deduction, etc. The common two components among most theories are the discovery of a form of relation or mapping in the source and extension of the relation to the target. In [165], we explore solving visual analogy questions where three images I_a, I_b, and I_c are provided as input and a fourth image I_d must be selected as output. We learned an embedding space from a set of training analogies to generalize for unseen analogies across various categories and attributes using a siamese Convolutional Neural Network (CNN). Using this network, analogical reasoning can be done with simple vector transformations. We extend this idea in [164] to longer sequence of frames to discover the action dependent transformations for the problem of viewpoint invariant goal reaching with a robotic arm. Based on this idea, we devised a recurrent controller that uses past memory of observations and actions to learn a self adaptive policy via a combination of reinforcement objective and learning from synthetic demonstrations. Using diverse simulated data consisting of demonstrated trajectories and reinforcement learning objectives, our visually-adaptive network is able to control a robotic arm to reach a diverse set of visually-indicated goals, from various viewpoints and independent of camera calibration.

1.4 Contributions

The contributions of this thesis are as follows:

Chapter 2: In this chapter we consider the problem of semantic visual reasoning. We present a self-supervise approach for verification of relational phrases using only visual data in the form of images crawled from Internet. We show that object presence cues in 2D RGB images can be used for semantic reasoning about object relationships. This work is published in [161].

Chapter 3: In this chapter we present our proposed simulation randomization technique (also called domain randomization). Simulation randomization in an approach for learning vision-based robot policies with high generalization such that they can be directly deployed on a real robot and work well in diverse real world situations. We demonstrate the effectiveness and applicability of simulation randomization in the context of indoor flight with a cheap drone that only uses RGB images as input and develop CAD^{2}RL method that is Collision Avoidance via Deep Reinforcement Learning algorithm in simulated CAD environment. This work is published in [162].

Chapter 4: In this chapter we present an extension our CAD^{2}RL [162] work for learning semantic aware and domain invariant visual servoing policies for semantic goal reaching by mobile robots. This new approach empowers simulation randomization technique to do visual semantic reasoning by decoupling semantics and physics. We have called this method as Domain Invariant Visual Servoing (DIViS) and it enables direct transfer of semantic aware policies from simulation to the real world. DIViS is presented in [160].

Chapter 5: In this chapter we present a new method for learning analogical reasoning using only visual information from a sequence of three RGB images. Visual analogical reasoning learns a high dimensional embedding space to infer the transformation between pairs of images. This work is called VISALOGY and is published in [165]. Our work on learning self-adaptive behavior based on pure visual feedback which will be explained in Chapter 6 is inspired from visual analogical reasoning.

Chapter 6: In this chapter, we propose a new method for learning self-adaptive behavior based on visual feedback. We study a new problem of viewpoint invariant visual servoing for goal reaching

with a robotic arm. Inspired from ideas on making visual analogical reasoning, we devised a recurrent controller that uses past memory of observations and actions to learn a self adaptive policy via a combination of reinforcement objective and learning from synthetic demonstrations collected in a domain randomized 3D physics simulator. Additionally, we show that disentanglement between perception and control eases transfer to unseen environments, and makes the model both flexible and efficient since each of its parts (i.e. "perception" or "control") can be independently adapted to new environments with small amounts of data. We also devised a deep neural network that combines perception and control trained end-to-end simultaneously, while also allowing each to be learned independently if needed. This design enabled our model to do fast transfer to unseen environments. While the control portion of the network in this work was entirely trained by the simulated data, the perception part of our network was complemented by collecting a small amount of static images with object bounding boxes without needing to collect the trajectory of whole action sequence with a physical robot. This work is published in [164].

Chapter 7: In this chapter we conclude by discussing about future work and open challenges in learning domain invariant and semantic aware vision-based robot policies.

Chapter 2

VISUAL KNOWLEDGE EXTRACTION AND VISUAL VERIFICATION OF RELATION PHRASES

"The things that make me different are the things that make me. "

A. A. Milne, *Winnie-the-Pooh*

How can we know whether a statement about our world is valid. For example, given a relationship between a pair of entities e.g., 'eat(horse, hay)', how can we know whether this relationship is true or false in general. Gathering such knowledge about entities and their relationships is one of the fundamental challenges in knowledge extraction. Most previous works on knowledge extraction have focused purely on text-driven reasoning for verifying relation phrases. In this work, we introduce the problem of visual verification of relation phrases and developed a Visual Knowledge Extraction system called VisKE. Given a verb-based relation phrase between common nouns, our approach assess its validity by jointly analyzing over text and images and reasoning about the spatial consistency of the relative configurations of the entities and the relation involved. Our approach involves no explicit human supervision thereby enabling large-scale analysis. Using our approach, we have already verified over 12000 relation phrases. Our approach has been used to not only enrich existing textual knowledge bases by improving their recall, but also augment open-domain question-answer reasoning.

2.1 Knowledge Extraction & Visual Reasoning

Do dogs eat ice cream? If you know the answer to this question, then you either have witnessed dogs eating ice cream or have observed either visual or textual recordings of this phenomenon. Extracting such knowledge has been a long-standing research focus in AI, with a variety of techniques

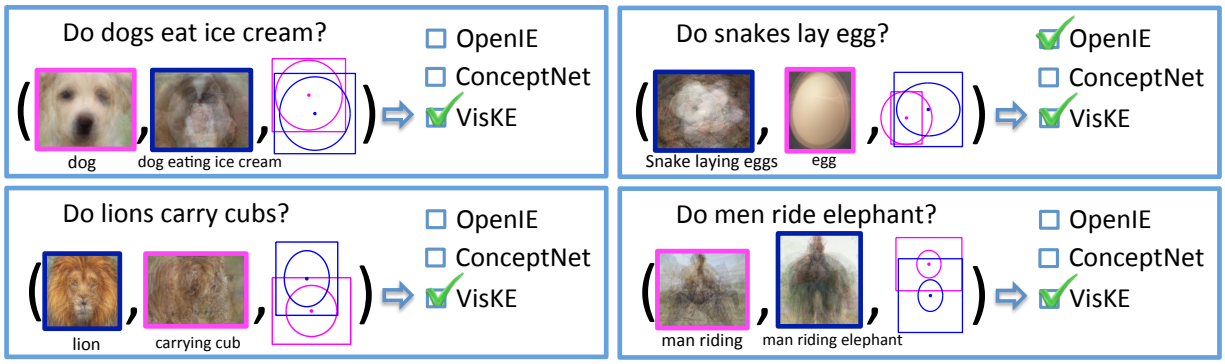

Figure 2.1: Do dogs eat ice cream? While we humans have no trouble answering this question, existing text-based methods have a tough time. In this chapter, we present a novel approach that can visually verify arbitrary relation phrases.

for automatically acquiring information of our world.

Vision is one of the primary modalities for us humans to learn and reason about our world. We gather our everyday basic knowledge such as *horses eat hay* or *butterflies flap wings* by simply observing these phenomenon in our real world. Yet when extracting knowledge for building intelligent systems, most previous research has focused on reasoning primarily using language and text.

Why such a disconnect? This disparity has mainly stemmed from the fact that we have had easier access to copious amounts of text data on the web along with well-performing feature representations and efficient unsupervised learning methods for text. However this does not hold true any more. Thanks to the proliferation of camera phones and the popularity of photo-sharing websites, recent years have witnessed a deluge of images on the web. Coupled with the growing success of text-based image search engines and the recent progress in weakly-supervised object localization methods, we believe the time is ripe now for extracting knowledge by reasoning with images.

Problem Overview: The key component of any knowledge extraction system involves verifying the validity of a piece of gathered information before adding it to a knowledge base. The most typical format of the information being considered is in the form of a *relationship* between a pair of *mentions* e.g., *eat(horse, hay)*, *flutter(butterfly, wings)*, etc.

The primary focus of our work is to estimate the confidence of such mentions-relation predi-

cates by reasoning with images. We focus our attention to verb-based relations between common nouns. The input to our system is a relation predicate e.g., 'eat(horse, hay)' and the output is a confidence value denoting its validity. In order to correctly validate a relation, we need to reason about the underlying entities while grounding them in the relation being considered. Here, we present a novel verification approach that reasons about the entities in the context of the relation being considered using webly-supervised models for estimating the spatial consistency of their relative configurations.

The attractive feature of our proposed framework is that both our model learning as well as inference steps are performed using no explicit human supervision. Most previous research on analyzing objects and their relationships in computer vision have assumed a supervised setting i.e., images along with some annotations of the objects and actions involved are available at training. This limitation has prevented these methods to scale to a large number of objects and relations. Our proposed approach overcomes this limitation by carefully leveraging unlabeled images found on the web, thereby enabling image-based reasoning for knowledge extraction.

In summary, our key contributions are: (i) We introduce the problem of visual verification of relation phrases for the task of knowledge extraction. (ii) We present an unsupervised approach for verifying relationships by analyzing the spatial consistency of the relative configurations of the entities and the relation involved. (iii) We empirically demonstrate the utility of our approach on a large relation phrase dataset and analyze the relative contributions of the different system components. (iv) To date, we have verified over 12000 relation phrases and doubled the size of the ConceptNet knowledge base [180] at a precision of 0.85. (v) We released our data and system for enabling future research and applications in this direction. (We invite the interested reader to verify a relation phrase of their choice using our online system.)

2.2 Related Work

The task of verifying relation phrases has received extensive attention in the field of information extraction. Phenomenal progress has been achieved using a variety of methods [7, 14, 28, 29, 51, 153, 184]. The core idea behind these methods involves analyzing the frequency of occurrence

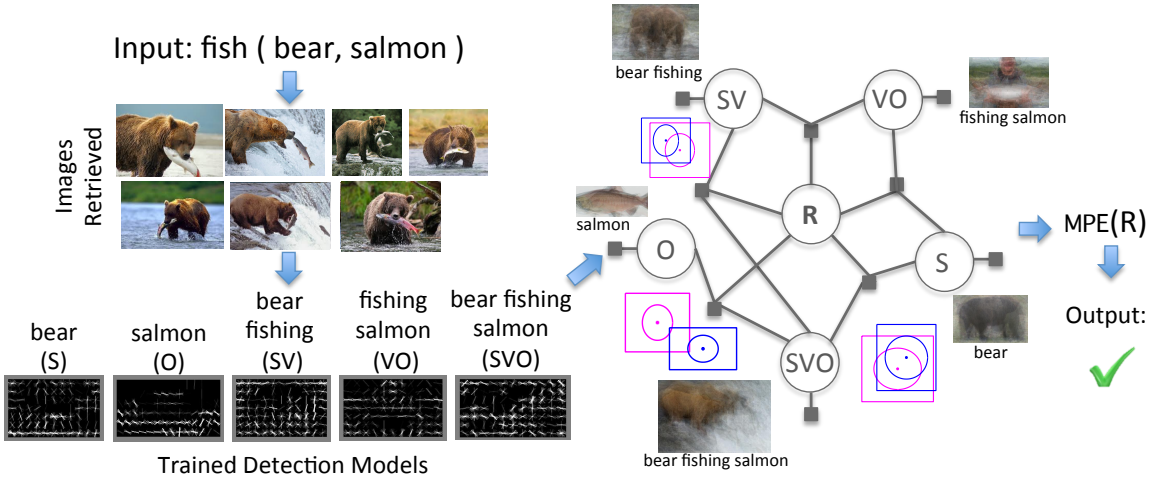

Figure 2.2: Approach Overview. Given a relation predicate, such as fish(bear,salmon) VisKE formulates visual verification as the problem of estimating the most probable explanation (MPE) by searching for visual consistencies among the patterns of subject, object and the action being involved.

of a given relation predicate in large text corpora [14, 51]. While frequency of occurrence in text is a reasonable indicator for the validity of a relation, it is not completely fool-proof. Many high frequency relations occur in text but are not true in the real world e.g., 'pierce(pelican, breast)'. Conversely many relations occur in text with low frequency but are true in the real world e.g., 'eat(chimpanzee, icecream)'. This anomaly springs from the fact that we humans often fail to explicitly state (in text) several obvious pieces of knowledge [74, 198] and therefore text-based methods can miss many basic relationships. Nonetheless these phenomenon are captured in the photos that we take in our daily life. Therefore by reasoning with images, we can leverage complementary cues that are hard to gather purely from text.

However, the task of relation verification has not yet received much attention in computer vision. Most previous research in this area has primarily focused on the tasks of image classification [47], scene recognition [49, 138, 163, 204] and object detection [55, 61, 69, 157] that form the fundamental building blocks for higher order visual reasoning systems. Subsequent research has leveraged the success in the classification and detection tasks to gather structured visual knowledge about objects, scenes and other concepts on an Internet scale [36, 48]. Also, in [220, 219] the problem of learning common sense knowledge from clip art images was studied.

Recent years have also witnessed a surge in reasoning about human-object relationships [75, 206] as well as more general object-object relationships [61, 92, 109, 123] and object-attribute relationships [36, 58, 217]. However, almost all works have studied this problem in the supervised setting i.e., images along with some form of annotations for the objects and the actions involved are assumed to be provided during training.

Quite related to our work is the work of [36], wherein the goal was to extract common-sense relationships between objects in an unsupervised setting. By analyzing the co-detection pattern between a pair of objects, the relationship between them was determined. Their focus was on two types of relationships: 'part of' and 'similar to'. In this work, we attempt to generalize their goal by learning and extracting more general relationships. We show it is possible to learn arbitrary relationships (such as 'eat', 'ride', 'jump', etc.,) by reasoning about the objects in the context of the relation connecting them. We have used our method to learn over 1100 relation types. Our work is complementary to the work of [217], where the utility of a knowledge base of relationships for performing diverse set of visual inference tasks was demonstrated. The knowledge gathered by our work can help enrich their underlying knowledge base, thereby facilitating more advanced inference tasks.

2.3 Visual Verification

Our key assumption in visual verification is that true relation phrases are those that happen in our real world and therefore there should exist enough visual recordings (images, videos) of them online. We consider visual verification of a relation phrase as the problem of searching for meaningful and consistent patterns in the visual recordings of the relation. But what are these patterns of consistencies for relation phrases?

For example, given a relation predicate, such as *fish(bear,salmon)* (read as Do bears fish salmon?) in Figure 2.2, we observe an *open-mouthed bear* attempting to catch hold of a *leaping salmon*, with the salmon appearing *in front* of the bear. This observation leads us to the following valuable insights about the subject and object involved in a relationship.

First, the appearance of the subject as well as the object may change during the relationship. In

this example, we see that the appearance of the bear while fishing salmon is different from a canonical appearance of a bear (i.e., open-mouthed). Similarly the appearance of a salmon when being fished is different from its canonical appearance (i.e., leaping). Therefore to find the occurrence of the subject and the object pattern (i.e., bear or salmon) in the images, it is important to search not only for their canonical patterns but also for their patterns under the relationship in consideration (i.e., 'bear fishing' and 'fishing salmon'). This change in the appearance due to interaction is aligned with the idea of visual phrases [166].

Second, the spatial locations of the subject and the object should be in a consistent behavior for the relationship to hold. In this example, we see that the salmon consistently appear in a specific location with respect to the bear for the relationship of fishing to be valid. Therefore to validate a relationship between a subject and an object, we need to check for the spatial consistency between the subject (bear) and the object (salmon) patterns and also between their modified versions (i.e., 'bear fishing', 'fishing salmon'). In the following, we present our formulation that generalizes these intuitions.

We refer to a relation as $\mathcal{V}(\mathcal{S},\mathcal{O})$, where $\mathcal{V}$ denotes the verb or the action connecting the subject $\mathcal{S}$ and the object $\mathcal{O}$). Based on our observations, participation in a relationship changes the appearance of the participating entities in different ways. For a relation $\mathcal{V}(\mathcal{S},\mathcal{O})$, we envision the following possibilities of meaningful patterns ($\mathcal{R}$): First, a relation ($\mathcal{V}$) might form a visual phrase and change the appearance of the subject ($\mathcal{S}$) i.e., $(\mathcal{SVO},\mathcal{SV})$; Second, the relation might affect the object ($\mathcal{O}$) i.e., $(\mathcal{S},\mathcal{VO})$; Third, the relation might form a

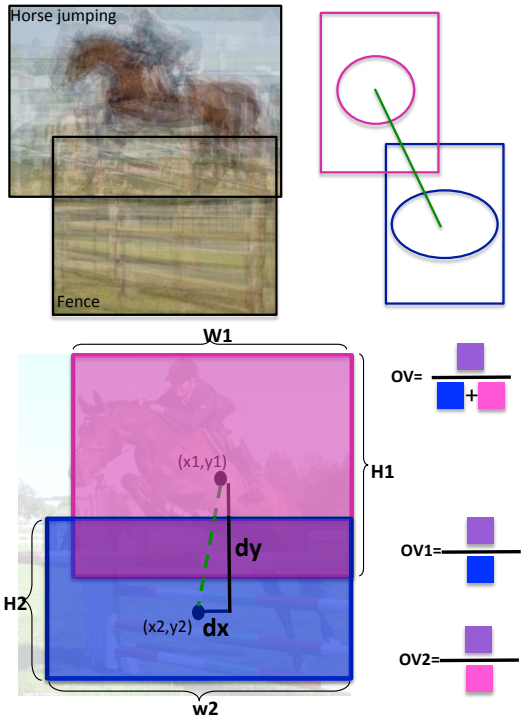

Figure 2.3: Our feature representation used for estimating the parametric relation between two detection elements. This figure shows the feature computed between 'Horse jumping' ($\mathcal{SV}$) and 'Fence' ($\mathcal{O}$). The ellipses on top show the distribution of spatial position of 'Horse jumping'(in pink) with respect to 'Fence'(in blue) as learned by our method. There is high horizontal variation in the position of 'Fence' compared to its vertical variation while the spatial position of Horse jumping is above 'Fence' and has small horizontal and vertical variation.

visual phrase and change the appearance of both subject and object i.e., $(\mathcal{VO}, \mathcal{SV})$; Fourth, the relation might impose specific visual characteristics on the subject but the object is not affected i.e., $(\mathcal{SVO}, \mathcal{O})$; and Fifth, the relation might impose specific visual characteristics on the object but the appearance of the subject remains intact i.e., $(\mathcal{SVO}, \mathcal{S})$. We ignored the $\mathcal{V}$, $\mathcal{SO}$ variables as in isolation they are highly visually ambiguous. We enforced the participation of all the three $\mathcal{S}$, $\mathcal{V}$, $\mathcal{O}$ entities in patterns and therefore avoide patterns like $(\mathcal{S}, \mathcal{SV})$ as it does not involve the $\mathcal{O}$.

Searching for consistencies among the above patterns require detectors for each of the elements of relations i.e., the subject ($\mathcal{S}$), the object ($\mathcal{O}$), the subject-verb combination ($\mathcal{SV}$), the verb-object combination ($\mathcal{VO}$), and the subject-verb-object combination ($\mathcal{SVO}$).

Assuming we have access to these individual detection models (explained later), we formulate visual verification as the problem of estimating the most probable explanation (MPE) of the multinomial distribution that governs $\mathcal{R}$. We factorize the marginalization of the joint distribution of $\mathcal{R}$ and the relation elements using a factor graph (depicted in Figure 2.2):

$$P(\mathcal{R}, \mathcal{S}, \mathcal{O}, \mathcal{SV}, \mathcal{VO}, \mathcal{SVO}) \propto \prod_{x \in \{\mathcal{O}, \mathcal{S}, \mathcal{SV}\}} \Phi(\mathcal{R}, \mathcal{SVO}, x) * \prod_{y \in \{\mathcal{SV}, \mathcal{S}\}} \Phi(\mathcal{R}, \mathcal{VO}, y) * \prod_{z \in \{\mathcal{S}, \mathcal{O}, \mathcal{SV}, \mathcal{VO}, \mathcal{SVO}\}} \Psi(z), \tag{2.1}$$

where $\mathcal{R}$ corresponds to the relation type and has a multinomial distribution over the patterns of consistency, the rest of the nodes correspond to relation element detectors. The potential function Φ provides the maximum likelihood estimates of each relation type. More specifically,

$$\Phi^i(\mathcal{R}, x, y) = \begin{cases} \max_\theta \mathcal{L}(x, y, \bar{I}; \theta) & \mathcal{R} \equiv i \\ 1 & \text{otherwise} \end{cases} \tag{2.2}$$

where $\bar{I}$ is the set of images collected for a relation phrase, and $\mathcal{L}(x, y, \bar{I}, \theta)$ is the log likelihood of the parametric relations between detections of x and y on image set $\bar{I}$ parameterized by θ. For

the parametric models we use Gaussians. The $\Psi(x)$ is the unary factor representing the maximum log likelihood estimates of predictions of detector x. Referring back to the example of bear fishing salmon, our factor graph checks for at least one of the five patterns to hold i.e., either 'bear fishing' and 'fishing salmon', or 'bear' and 'fishing salmon', or 'bear fishing salmon' and 'bear', 'bear fishing salmon' and 'salmon' or 'bear fishing salmon' and 'bear fishing' should have a highly consistent pattern for this relationship to be valid.

The features to estimate the maximum likelihood estimate $\mathcal{L}(x, y, \bar{I}; \theta)$ should capture the spatial relations between the predictions of detectors x and y. Towards this end, we use the following feature representation (See Figure 2.3): $\{dx, dy, ov, ov_1, ov_2, h_1, w_1, h_2, w_2, a_1, a_2\}$, where dx, dy correspond to the translation between detections, ov is the intersection over the union of the two detection boxes, ov_1 is the ratio of intersection over the area of the bounding box x, ov_2 is the ratio of the intersection over the area of bounding box y, h_1, w_1 are the dimensions of the bounding box x, h_2, w_2 are the dimensions of the bounding box y and a_1, a_2 are the x and y bounding box areas. For unary potentials we use a 4 dimensional representation that encodes $\{h, w, x, y\}$, where h, w are the height and width of the bounding box and x, y are its (mid-point) coordinates. Under this model, visual verification is the problem of MPE in our factor graph [141].

Implementation Details: We use the publicly-available implementation of [48] for learning our $\mathcal{S}$, $\mathcal{O}$, $\mathcal{SV}$, $\mathcal{VO}$, $\mathcal{SVO}$ detectors (without parts). For each detector, a mixture of components Deformable Part Model (DPM) [61] is trained using retrieved images from the web and the noisy components are pruned in a separate validation step. [1] As our individual detectors (i.e., $\mathcal{SV}$, $\mathcal{SVO}$, $\mathcal{S}$, $\mathcal{O}$, $\mathcal{VO}$) are mixture models, each of our factors (e.g., $(\mathcal{SV}, \mathcal{SVO})$) incorporate these mixtures.

2.4 Relation Phrase Dataset

To the best of our knowledge, there exists no gold-standard dataset of relation phrases in the knowledge extraction community. Different extraction systems use different strategies for extracting re-

[1] Using our current unoptimized linux-based Matlab implementation on a Intel Xeon E5 CPU, the entire run-time per relation is around 30mins.

	Base Set	Permute Set	Combined Set
Visual Phrase [166]	49.67	14.12	42.49
Co-detection Model	49.24	14.65	43.14
Google Ngram Model [7]	46.17	NA	NA
Language Model [88]	56.20	**22.68**	50.23
VisKE	**62.11**	20.93	**54.67**

Table 2.1: Results (M.A.P.) on the Relation Phrase Dataset. While the language model achieves a higher accuracy on the 'Permute' set, VisKE gets the best result on the 'Base' set and the 'Combine' set.

lational phrases [7, 14, 29]. To avoid biasing our proposed approach to the idiosyncrasies of any one of these methods, we have put together a generic dataset of relation phrases.

Our relation phrases were gathered using the Google Books Ngram (English 2012) corpus [125]. This corpus contains parts-of-speech tagged Ngrams along with their frequency of occurrence in books. To extract relations of 'verb(subject, object)' format, we considered all Ngrams that have a <noun, verb, noun> pattern (ignoring any other words in between). For the purpose of our experiments, we focused our attention to the relations involving animals. This resulted in a list of 6093 relations covering 45 different subjects, 1158 different verbs (actions) and 1839 different objects. To avoid biasing this list to contain only true relations, we generated new relations by randomly permuting the subjects, verbs and objects together yielding an additional 6500 relations (resulted in a total of 5776 pairs of $\mathcal{SV}$ and 5186 pairs of $\mathcal{VO}$ in our dataset). We refer to these relations as the 'Permute' set and the former as the 'Base' set. Some of the sample relations can be seen in Figures 5.1 2.2 2.5 2.8.

For evaluating the performance of our method as well as to compare different baselines, each relation phrase was annotated with its ground-truth validity. The validity was primarily decided based on whether a relations refers to a phenomenon that commonly happens in our real-world. Out of the total 12593 relations, 2536 statements were annotated as true and the rest as false[2].

[2]We have released our list of relational relations along with their annotations in our project website (`http://viske.allenai.org/`).

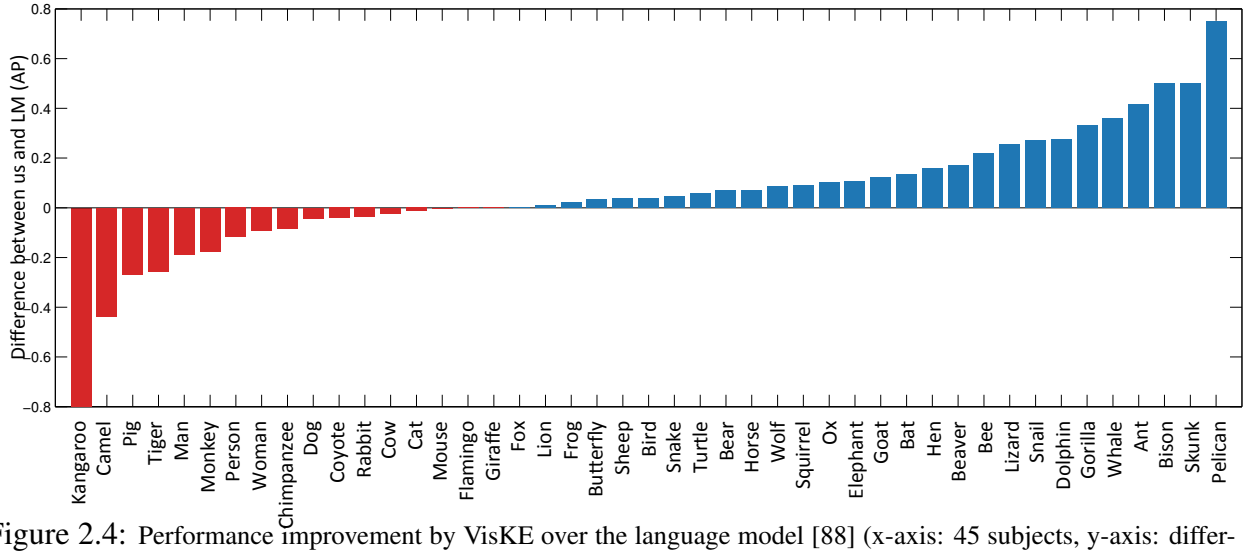

Figure 2.4: Performance improvement by VisKE over the language model [88] (x-axis: 45 subjects, y-axis: difference in A.P.). Blue indicates our approach does better than [88] (27/45 classes), while red indicates vice versa (14/45 classes).

2.5 Experimental Results & Analysis

We analyzed the efficacy of our approach by conducting experiments on the relation phrase dataset. The input to our approach is a relation phrase e.g., 'eat(horse, hay)' and the output is a confidence score denoting its validity (i.e., larger value indicates greater confidence in being true). We use these scores along with their corresponding ground-truth labels to compute a precision-recall curve and use the area under the precision-recall curve (Average Precision, A.P.) metric [55] as a principal quantitative measure. We computed the A.P. independently for each subject and then averaged the A.P. across all subjects (Mean A.P., M.A.P.). Table 2.1 summarizes the key results obtained using our approach. We separately evaluated the A.P over the 'Base' set and the 'Permute' set to analyze the impact in the different scenarios. Our proposed approach achieves an M.A.P. of 62.11% on the 'Base' set, and 20.93% on the 'Permute' set, indicating the difficulty of the latter compared to the former. Validating 12593 relations involved training over 26739 detectors and processing around 9 million images. Our experiments were run on a computer cluster. We also compared our results to the following baseline models to analyze different aspects of our system.

Co-detection model: We first compared against a simple approach that trains separate detection

Model	M.A.P.
VisKE (All Factors)	62.11
Without $\Phi(\mathcal{R}, \mathcal{VO}, \mathcal{SV})$	60.41
Without $\Phi(\mathcal{R}, \mathcal{VO}, \mathcal{S})$	61.16
Without $\Phi(\mathcal{R}, \mathcal{SVO}, \mathcal{S})$	60.40
Without $\Phi(\mathcal{R}, \mathcal{SVO}, \mathcal{O})$	59.55
Without $\Phi(\mathcal{R}, \mathcal{SVO}, \mathcal{SV})$	59.55
Without binary terms	60.61
Without unary terms	58.52
CRF	58.01

Table 2.2: Ablation analysis: while each of the factors help improving the overall performance, removing any of them does not drastically hurt its performance, indicating the robustness of our overall model. Removing either of the binary or unary terms hurts the performance. Using a CRF model results in poorer performance.

models for the entities and the relation involved (using the web images) and then analyzes the pattern of their co-detections. For example, in the case of 'eat(horse, hay)', separate detection models were trained for horse and hay as well as a detector for the relation eat (using their corresponding web images) and then reasoned about the pattern of their co-detections on 'horse eating hay' images. This approach is most similar to that of [36]. As seen in Table. 2.1 (row2), this approach fails to perform well as it considers each constituent of the relation phrase independently and thereby fails to account for the changes in appearance of the entities when involved together in an action [166]. For example, in case of the horse eats hay example, the appearance of the 'eating horse' is different from that of a canonical horse.

Visual Phrase model: We next compared our approach to the visual phrase model of [166], where a single $\mathcal{SVO}$ detector is trained and its performance on predicting its unseen samples is evaluated. As seen in Table. 2.1 (row1), this model fares poorly. We found it to classify several incorrect relations to be true as it does not validate the pattern of its constituent entities. For example, in case of 'horse read book', the retrieved images contain cover pages of books (about horses) all having a picture of horse[3]. Validating a detector on these images would lead to falsely claiming this relation to be true as it just analyzes the consistency of the overall pattern without reasoning

[3]This phenomenon happens as image-search engines predominantly rely on auxiliary text (around the image) in the documents for retrieving images.

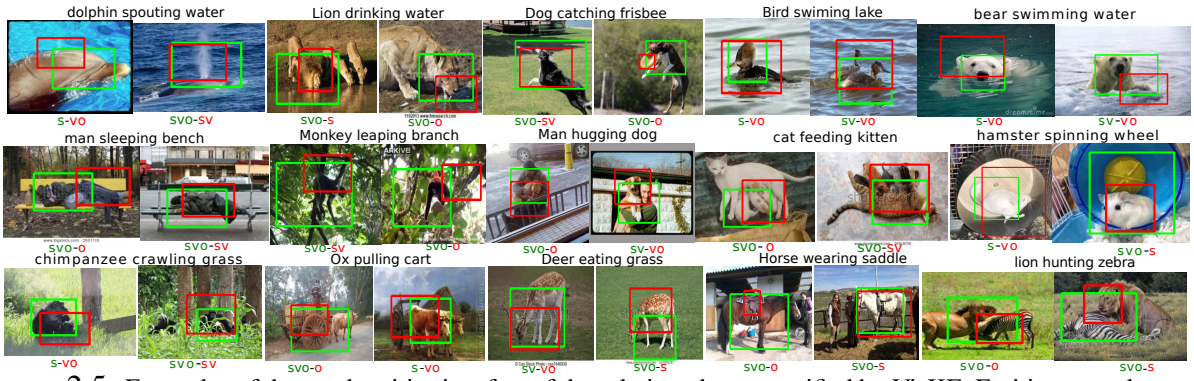

Figure 2.5: Examples of detected entities in a few of the relation phrases verified by VisKE. Entities are color coded within pattern.

about the action of horse reading.

Language model: Of course our task of visually verifying relation phrases has been motivated from the domain of text-driven knowledge extraction. It is therefore interesting to compare our method to contemporary text-driven methods for verifying relation phrases. We evaluated the performance of two popular industrial-sized language models (Bing, Google). The method of [88] estimates the real-world plausibility of any relation phrase using a sophisticated statistical language model learned from a large text corpora, while the method of [7] estimates the probabilities using a language model trained on the GoogleNgram corpus. As seen in Table. 2.1, although the language model of [88] outperforms the co-detection and phrasal baselines, it does not perform as well as our proposed approach.

To analyze performance per subject, in Figure. 2.4, we display the individual relative differences in A.P. Out of the 45 subjects, our approach does better on 27 of them, while the language model of [88] does better on 14. For subjects like 'pelican', 'lizard' etc., our approach does better, while for subjects like 'pig', 'monkey', language model does better. We hypothesize this to the fact that classes like monkey are more common than classes like pelican and therefore the language model has more data for these classes. This difference in performance between our model and the language model hints at the complementarity of the vision and language methods. To validate this hypothesis, we ran a separate evaluation (on the Permute set) by linearly combining the confidences produced by these two methods. This combined model indeed produced a higher

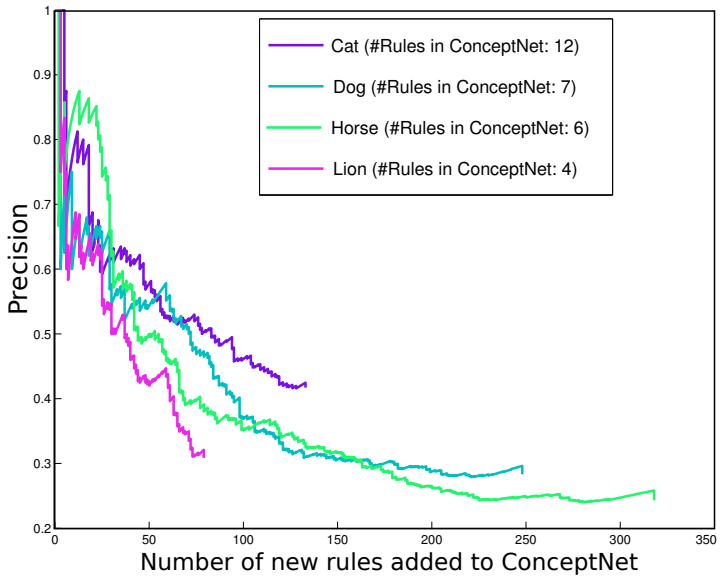

Figure 2.6: Enriching ConceptNet Knowledge Base: Figure shows the number of correct relationships added by VisKE at different levels of recall. For example, in case of 'cat', we have added 10 new relations to ConceptNet at a precision of 0.8.

M.A.P. of 24.25%[4], ascertaining the fact that reasoning with images offers cues complementary to text for relation phrase verification. As the number of relation phrases per subject in our dataset is imbalanced, we also evaluated the performance over the entire set of relation phrases (across all subjects) on the 'Base' set. This yielded an A.P. of 44.6% for our method, while the LM [17] obtained 40.2%.

Ablation Study: To understand which factors within our model are critical towards the final performance, we analyzed results by running experiments with the different factors turned off/on. As displayed in Table. 2.2, while each of the factors helps in improving the overall performance, removing any one of them does not drastically hurt the performance, indicating the robustness of our overall model. Also, as observed in Table 2.2, both the unary and the binary factors contribute towards the final performance. We also ran a separate experiment where we used a simple CRF based pooling strategy to combine the responses of the different pattern relationships i.e., $(\mathcal{SVO}, \mathcal{SV})$, $(\mathcal{VO}, \mathcal{SV})$, etc., which resulted in poorer performance.

[4]The performance on the Base set did not improve. We hypothesize this due to our simple linear combination strategy. Given the different score calibrations of the visual and language model, it is a challenge to combine them meaningfully in an unsupervised setting.

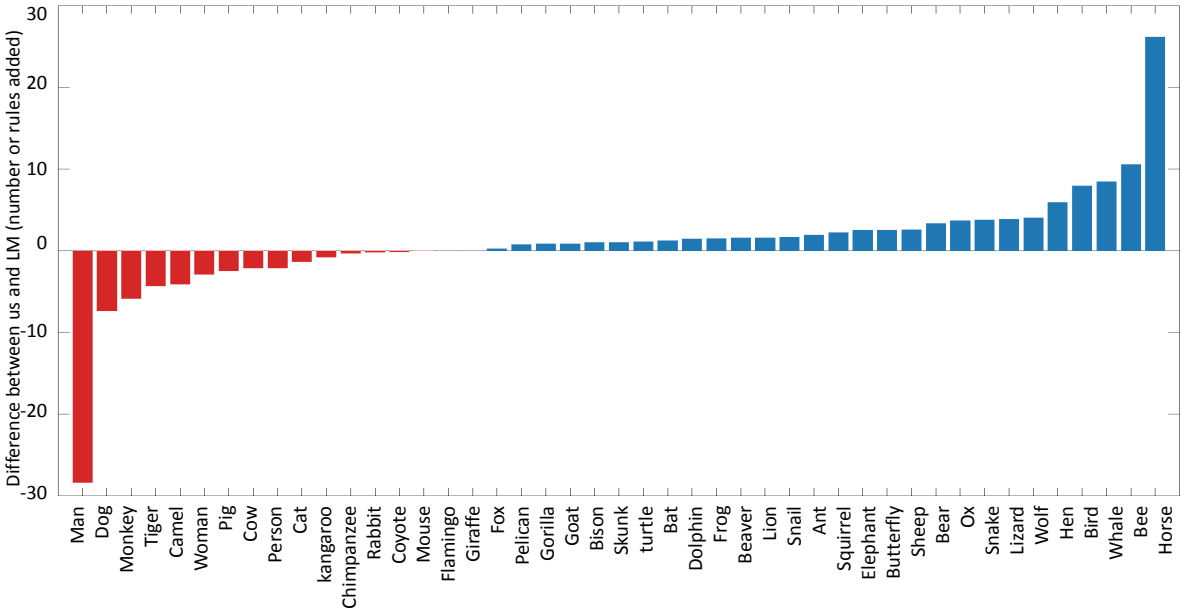

Figure 2.7: Relative difference of the number of added relations to ConceptNet using VisKE over the language model [88] (x-axis: 45 subjects, y-axis: difference in number of relations). Blue indicates that VisKE adds more relations than [88] (28/45 classes), while red indicates vice versa (14/45 classes).

What are the sources of errors that prevent our model in correctly verifying some of the relationships? We found a couple of issues. First, our method is dependent on web image search engines to gather the relevant set of images. For some relations, e.g. make(fox,den), the retrieved images are not relevant, while for some other relations, e.g. shed(cow, horn), the images are misleading. Second, our method uses the webly-supervised approach of [48] for training the detectors, which sometimes fails either when the variance within the set of retrieved images is large, e.g. eat(horse, fruit), or if the relation involves complex visual characteristics, e.g. drink(hedgehog, milk). Finally, the inherent spatial relationships in case of certain relation phrases is complex, e.g. cross(horse, road). Verifying such relations require deeper understanding of spatial relations. Future work could explore leveraging (textual) prepositions to better understand complex spatial relationships.

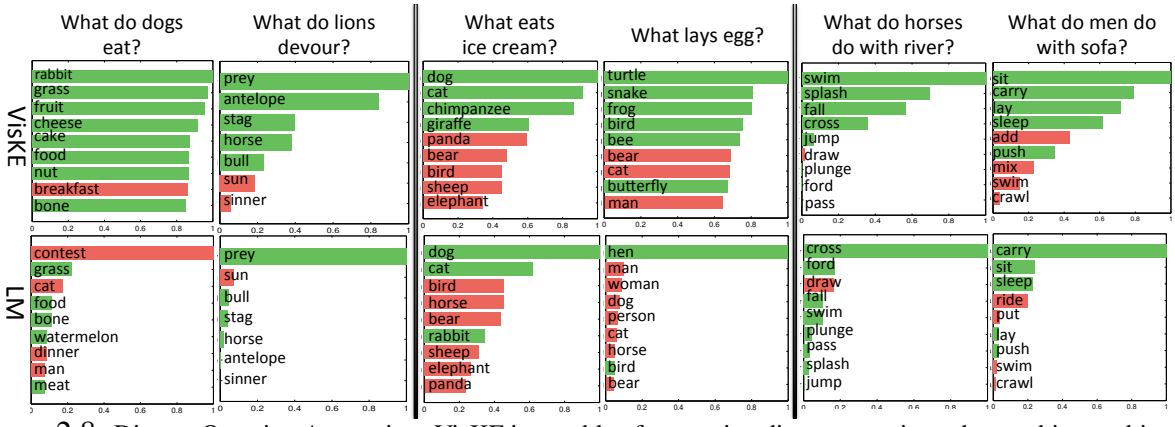

Figure 2.8: Diverse Question Answering: VisKE is capable of answering diverse questions about subjects, objects or actions. In comparison to the Language model [88], VisKE obtains richer and more precise answers to the questions.

2.5.1 Application: Enriching Knowledge Bases

Current knowledge bases such as WordNet, Cyc, ConceptNet, etc., seldom extract common-sense knowledge directly from text as the results tend to be unreliable and need to be verified by human curators. Such a manual process is both labor intensive and time consuming. A nice feature of our method is that it offers complementary and orthogonal source of evidence that helps in discovering highly confident facts from amongst the pool of all facts extracted from text. This feature helps us towards automatically improving the recall of knowledge bases. We demonstrate this feature on the popular ConceptNet knowledge base.

ConceptNet is a semantic network containing common-sense knowledge collected from volunteers on the Internet since 2000 [180]. This knowledge is represented as a directed graph whose nodes are concepts and edges are assertions of common sense relations about these concepts. The set of possible relations is chosen to capture common informative patterns, such as 'IsA', 'PartOf', 'HasA', 'MemberOf', 'CapableOf', etc.

In our analysis, we measured the number of relation predicates our visual approach could add to this resource. More specifically, for each of the 45 subjects in the relation phrase dataset, we measured the precision of correct relationships (that are unavailable in ConceptNet) added at different levels of recall. While some of the relationships (e.g., 'IsA(horse, animal)', 'PartOf(wheel, car)', 'HasA(horse, leg)') are easier to acquire and have been previously explored in computer

Model	M.A.P.
OpenIE [56]	73.03
Co-detection Model	76.65
Visual Phrase [166]	78.45
Language Model [88]	83.65
VisKE	85.80

Table 2.3: Results on the OpenIE dataset.

vision [182, 123, 36], more complex *action-centric* relationships (e.g., 'CapableOf(horse, jump fence)') have not received much attention. In fact, to our surprise, across the 45 concepts there were only 300 'CapableOf' relation facts within ConceptNet[5]. In our analysis, we primarily focused on increasing the number of facts pertaining to this relationship.

Figure. 2.6 summarizes the key results of our analysis. It displays the number of relations added at various precision levels for four (of the 45) subjects. For example, in case of 'cat', we have added 10 new relations to ConceptNet at a precision of 0.8. Some of the newly added relations are (i.e., unavailable in ConceptNet): 'CapableOf(cat, lick kitten)', 'CapableOf(cat, carry mouse)', 'CapableOf(cat, sit basket)', 'CapableOf(cat, chase bird)', and 'CapableOf(cat, lay cushion)'. On average, at a precision of .85, we doubled the size of the 'CapableOf' relations in ConceptNet (related to our subjects). In Figure. 2.7, we compare the number of relations added by our approach with respect to the language model [88]. Out of the 45 subjects, our approach adds more relations on 28 of them, while the language model [88] does better on 14. This visualization again reveals the the complementarity in performance between our model and the language model.

OpenIE: We conducted a similar analysis on OpenIE [14, 56], a popular web-scale information extraction system that reads and extracts meaningful information from arbitrary web text. We selected the relations corresponding to the 45 subjects of interest within their extractions [56] and ran our approach to compute the confidences. Table. 2.3 summarizes our results. Our approach helps in improving the extractions obtained by OpenIE as well.

[5]ConceptNet is an everchanging repository. The results here correspond to the latest version downloaded on September 26 2014.

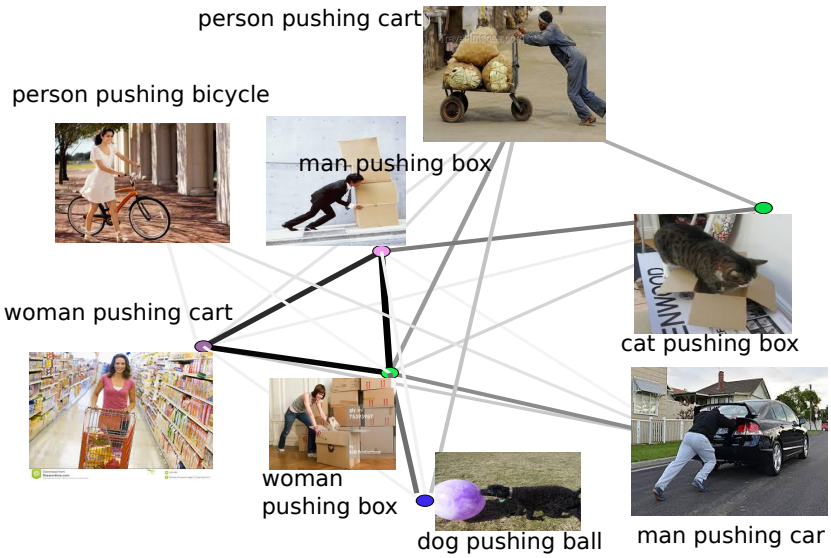

Figure 2.9: Higher-order reasoning of relationships: This relation graph reveals to us that the action of 'man pushing box' is very similar to 'woman pushing cart', but not to 'person pushing bicycle'. The edge thickness shows the strength of a relation. Darker edge indicates higher similarity.

Towards Higher-order Reasoning: An interesting feature enabled by our approach is reasoning about higher-order relationships. Figure. 2.9 shows the relation graph learned by our approach based on the learned spatial models for the different relation phrases involving the relationship of 'pushing'. These higher-order relations were estimated by computing the cosine similarity of the different pairs of relations based on the maximum similarity of their corresponding patterns (i.e., $(\mathcal{SVO}, \mathcal{SV})$, $(\mathcal{VO}, \mathcal{SV})$, etc.,) in the factor graph. The cosine similarity is computed using the feature representation as explained in section 2.3. The relation graph reveals that the action of 'man pushing box' is very similar to 'woman pushing cart', but not to 'person pushing bicycle'. Such reasoning about similarities and dissimilarities between relationships (in the context of the entities involved) is valuable for several tasks in vision and language.

2.5.2 Application: Question Answer Reasoning

Question-Answering is an important AI problem where the goal is to answer questions by querying large knowledge bases [19, 57]. The visual knowledge that we have gathered using our approach can help improve the reasoning within question-answering systems. For example, as shown in

Figure. 2.8, it could be possible for users to explore and discover a variety of interesting facts about concepts and their relationships, such as:

What do dogs eat? Given a query of the form 'verb(subject,?)', we can use our approach to retrieve the top set of objects relevant to it. For e.g., our approach reveals that dogs typically eat rabbit, grass, fruit, etc. In contrast, the LM has produced high probability for 'contest' as it is confused by 'hot dog eating contest'.

What lays eggs? Given a query of the form 'verb(?, object)', we can use our approach to retrieve the top set of subjects relevant to it. For e.g., our approach reveals that the most probable animals that can lay eggs are turtle, snake, etc.

What do horses do with river? Given a query of the form '?(subject, object)', our approach can retrieve the top set of relations between the subject and object. For e.g., our approach reveals that the most probable relationship between butterfly and wings are flutter and flap, and between man and sofa are sit, carry, sleep, push, etc.

Towards Answering Elementary-level Science Questions: Apart from answering generic questions, our approach can be useful for answering more specific questions such as those related to elementary-level general science [39, 87, 172, 83]. To demonstrate this, we ran a preliminary experiment using our approach on a small subset of the NewYork Regents' 4th grade science exam [39] that were visually relevant. Given a question such as 'What part of a plant produces seeds? (a) Flower, (b) Leaves, (c) Stem, (d) Roots', it is decomposed into its constituent relations[6] i.e., 'produce(flower, seeds)', 'produce(leaves, seeds)', 'produce(stem, seeds)', 'produce(roots, seeds)'. Our approach validates each of the relations and outputs a confidence value for them. This confidence is used to pick the right answer. Our approach achieved an accuracy of 85.7% (compared to 71.4% by a text-driven reasoning approach [39]) highlighting the benefit of our visual reasoning based question answering approach.

[6]The relations for the questions were manually created. Automatic relations generation is a challenging research problem [39].

2.6 Conclusion

Relation verification constitutes a fundamental component within any knowledge extraction system. In this chapter, we highlighted the importance of visual reasoning for relation phrase verification. We presented a novel approach for visual verification that reasons about the entities in the context of the relation being considered, by estimating the spatial consistency of their relative configurations using webly-supervised models. Using our approach, we demonstrated impressive results on a large relation phrase dataset and also highlighted the complementarity of the cues provided by our approach in comparison to existing linguistic models. Further we also demonstrated the utility of our approach in enriching existing knowledge bases and visual question answering.

Chapter 3

SIM2REAL COLLISION AVOIDANCE VIA DEEP REINFORCEMENT LEARNING

Miranda begins, "The glory of creation is in its infinite diversity. " Spock replies with, "And the ways our differences combine, to create meaning and beauty."

Star Trek

Deep reinforcement learning has emerged as a promising and powerful technique for automatically acquiring control policies that can process raw sensory inputs, such as images, and perform complex behaviors. However, extending deep RL to real-world robotic tasks has proven challenging, particularly in safety-critical domains such as autonomous flight, where a trial-and-error learning process is often impractical. In this chapter, we explore the following question: can we train vision-based navigation policies entirely in simulation, and then transfer them into the real world to achieve real-world flight without a single real training image? We propose a learning method that we call CAD^{2}RL, which can be used to perform collision-free indoor flight in the real world while being trained entirely on 3D CAD models. Our method uses single RGB images from a monocular camera, without needing to explicitly reconstruct the 3D geometry of the environment or perform explicit motion planning. Our learned collision avoidance policy is represented by a deep convolutional neural network that directly processes raw monocular images and outputs velocity commands. This policy is trained entirely on simulated images, with a Monte Carlo policy evaluation algorithm that directly optimizes the network's ability to produce collision-free flight. By highly randomizing the rendering settings for our simulated training set, we show that we can train a policy that generalizes to the real world, without requiring the simulator to be particularly

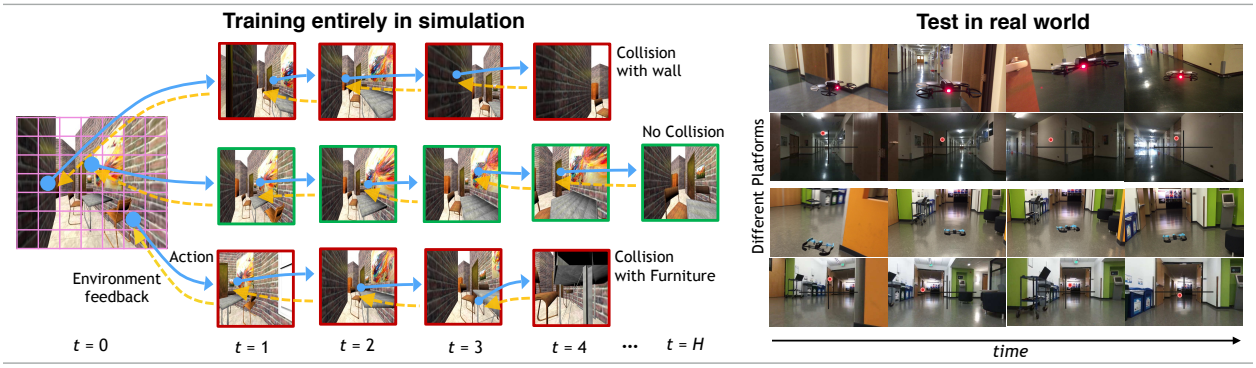

Figure 3.1: We propose the **C**ollision **A**voidance via **D**eep **R**einforcement **L**earning algorithm for indoor flight which is entirely trained in a simulated **CAD** environment. Left: CAD^{2}RL uses *single image* inputs from a monocular camera, is exclusively trained in simulation, and does not see any real images at training time. Training is performed using a Monte Carlo policy evaluation method, which performs rollouts for multiple actions from each initial state and trains a deep network to predict long-horizon collision probabilities of each action. Right: CAD^{2}RL generalizes to real indoor flight.

realistic or high-fidelity. We evaluate our method by flying a real quadrotor through indoor environments, and further evaluate the design choices in our simulator through a series of ablation studies on depth prediction. For supplementary video see: https://youtu.be/nXBWmzFrj5s

3.1 Overview

Indoor navigation and collision avoidance is one of the basic requirements for robotic systems that must operate in unstructured open-world environments, including quadrotors, mobile manipulators, and other mobile robots. Many of the most successful approaches to indoor navigation have used mapping and localization techniques based on 3D perception, including SLAM [12], depth sensors [215], stereo cameras [170], and monocular cameras using structure from motion [31]. The use of sophisticated sensors and specially mounting multiple cameras on the robot imposes additional costs on a robotic platform, which is a particularly prominent issue for weight and power constrained systems such as lightweight aerial vehicles. Monocular cameras, on the other hand, require 3D estimation from motion, which remains a challenging open problem despite considerable recent progress [52, 104]. In this chapter, we explore a learning-based approach for indoor navigation, which directly predicts collision-free motor commands from monocular images, without attempting to explicitly model or represent the 3D structure of the environment. In contrast to previous learning-based navigation work [20], our method uses reinforcement learning to obtain

supervision that accurately reflects the actual probabilities of collision, instead of separating out obstacle detection and control. The probability of future collision is predicted from raw monocular images using deep convolutional neural networks.

Using reinforcement learning (RL) to learn collision avoidance, especially with high-dimensional representations such as deep neural networks, presents a number of major challenges. First, RL tends to be data-intensive, making it difficult to use with platforms such as aerial vehicles, which have limited flight time and require time-consuming battery changes. Second, RL relies on trial-and-error, which means that, in order to learn to avoid collisions, the vehicle must experience at least a limited number of collisions during training. This can be extremely problematic for fragile robots such as quadrotors.

A promising avenue for addressing these challenges is to train policies in simulation, but it remains an open question whether simulated training of vision-based policies can generalize effectively to the real world. In this work, we show that we can transfer indoor obstacle avoidance policies based on monocular RGB images from simulation to the real world by using a randomized renderer, without relying on an extremely high degree of realism or visual fidelity. Our renderer forces the network to handle a variety of obstacle appearances and lighting conditions, which makes the learned representations invariant to surface appearance. As the result, the network learns geometric features and can robustly detect open spaces.

In contrast to prior work on domain adaptation [158, 196], our method does not require any real images during training. We demonstrate that our approach can enable navigation of real-world hallways by a real quadrotor using only a monocular camera, without depth or stereo. By training entirely in simulation, we can also use a simple and stable RL algorithm that exploits the ability to reset the environment to any state. Figure 5.1 shows a diagram of our CAD^{2}RL algorithm. The algorithm evaluates multiple actions at each state using the current policy, producing dense supervision for the Q-values at that state. Training the Q-function to regress onto these Q-values then corresponds to simple supervised learning. This algorithm sidesteps many of the hyperparameter tuning challenges associated with conventional online RL methods, and is easy to parallelize for efficient simulated training.

The main contribution of our work is an approach for training collision avoidance policies for indoor flight using randomized synthetic environments and deep RL. We designed a set of synthetic 3D hallways that can be used to generate large datasets of randomized scenes, with variable furniture placement, lighting, and textures. Our synthetic data is designed for the task of indoor robot navigation and can also be used as a testbed for RL algorithms. Our proposed RL method is also a novel contribution of this work, and is particularly simple and well-suited for simulated training.

We present an extensive empirical evaluation that assesses generalization to the real world, as well as ablations on a supervised proxy task that studies which aspects of the randomized simulation are most important for generalization. Our simulated comparative evaluation shows that our approach outperforms several baselines, as well as a prior learning-based method that predicts turning directions [71]. Our real-world experiments demonstrate the potential for purely simulation-based training of deep neural network navigation policies. Although the policies trained entirely in simulation do experience some collisions in the real world, they outperform baseline methods and are able to navigate effectively around many kinds of obstacles, using only monocular images as input. We therefore conclude that simulated training is a promising direction for learning real-world navigation for aerial vehicles as well as other types of mobile robots.

3.2 Related Work

Any robotic system that must traverse indoor environments is required to perform basic collision avoidance. Standard methods for collision-free indoor navigation take a two step approach to the problem: first map out the local environment and determine its geometry, and then compute a collision-free path for reaching the destination [176]. This approach benefits from independent developments in mapping and localization as well as motion planning [174, 133, 17]. The 3D geometry of the local environment can be deduced using SLAM with range sensors [12], consumer depth sensors [215, 79], stereo camera pairs [170], as well as monocular cameras [31]. In [126], laser range scanned real images are used to estimate depth in a supervised learning approach and then the output is used to learn control policies. In [76] simultanouse mapping and planning using

RGB-D images is done via a memory network. Reconstruction from monocular images is particularly challenging, and despite considerable progress [104, 52], remains a difficult open problem. In a recent approach, IM2CAD, CAD model of a room is generated from a single RGB image [93]. While the synthetic data generated by [93] could be used for various robotics simulations, the computational overhead makes it less suitable for autonomous indoor flight, where quick inference for finding open spaces is more critical than categorical exact 3D models.

In our work, we sidestep the challenges of 3D reconstruction by proposing a learning algorithm that can directly predict the probability of collision, without an explicit mapping phase. Learning has previously been used to detect obstacles for indoor flight [20, 101], as well as to directly learn a turn classifier for outdoor forest trail following [71]. In contrast to the work of [20], our method directly learns to predict the probability of collision, given an image and a candidate action, without attempting to explicitly detect obstacles. However, our approach still affords considerable flexibility in choosing the action: a higher-level decision making system can choose any collision-free action based, for example, on a higher-level navigational goal. This is in contrast to the prior work, which simply predicts the action that will cause the vehicle to follow a trail [71]. Unlike [71], our method does not require any human demonstrations or teleoperation.

Besides presenting a deep RL approach for collision avoidance, we describe how this method can be used to learn a generalizable collision predictor in simulation, such that it can then generalize to the real world. Simulated training has been addressed independently in the computer vision and robotics communities in recent years. In computer vision, a number of domain adaptation methods have been proposed that aim to generalize perception systems trained in a source domain into a target domain [197, 85]. In robotics, simulation to real-world generalization has been addressed using hierarchies of multi-fidelity simulators [45], priors imposed on Bayesian dynamics models [44]. At the intersection of robotics and computer vision, several works have recently applied domain adaptation techniques to perform transfer for robotic perception systems [196, 158, 155]. In contrast to these works, our method does not use any *explicit* domain adaptation. Instead, we show how the source domain itself can be suitably randomized in order to train a more generalizable model, which we experimentally show can make effective predictions on a range of systematically

different target domains.

Our method combines deep neural networks for processing raw camera images [111] with RL. In the seminal work of [146], a fully connected neural network is used for generating steering commands for the task of road following using raw pixels and laser range finder. Recently, a similar approach was proposed by [22] for a self-driving car. We also generate direction commands from raw visual inputs. However, unlike these prior works, we use RL and do not require any human demonstration data. Furthermore, our method commands the vehicle in 3D, allowing it to change both heading and altitude. Vision-based RL has previously been explored in the context of Q-iteration [151], and more recently for online Q-learning using temporal-difference algorithms [131]. However, these methods were evaluated primarily on synthetic video game domains. Several recent works have extended deep RL methods to real-world robotics applications using either low-dimensional estimated state [37] or by collecting an exhaustive real-world dataset under gridworld-like assumptions [211]. In contrast, we propose a simple and stable deep RL algorithm that learns a policy from raw monocular images and does not require seeing any images of the real-world test environment.

3.3 Collision Avoidance Via Deep Reinforcement Learning

Our aim is to choose actions for indoor navigation that avoid collisions with obstacles, such as walls and furniture. While we do not explicitly consider the overall navigation objective (e.g. the direction that the vehicle should fly to reach a goal), we present a general and flexible collision avoidance method that predicts which actions are more or less likely to result in collisions, which is straightforward to combine with higher-level navigational objectives. The input to our model consists only of monocular RGB images, without depth or other sensors, making it suitable for low-cost, low-power platforms, though additional sensory inputs could be added in future work. Formally, let $\mathbf{I}_t$ denote the camera observation at time t, and let $\mathbf{a}_t$ denote the action, which we will

define in Section 3.3.1. The goal of the model is to predict the Q-function $Q(\mathbf{I}_t, \mathbf{a}_t)$:

$$Q(\mathbf{I}_t, \mathbf{a}_t) = \sum_{s=t, \mathbf{a} \sim \pi}^{t+H} \gamma^{s-t} \mathcal{R}(\mathbf{I}_s, \mathbf{a}_s), \tag{3.1}$$

where $\gamma \in (0, 1)$ is the discount factor, and actions are assumed to be chosen by the current policy π, which we discuss in Section 3.3.1. The horizon H should ideally be ∞, but in practice is chosen such that γ^H is small. $\mathcal{R}$ is the reward function and is equal to zero if collision event happens. Collisions are assumed to end the episode, and therefore can occur only once. Otherwise, the reward at time s is defined as $\min(1, \frac{d_s - r}{\tau_d - r})$, where r is the radius of the vehicle, d_s is the distance to the nearest obstacle at time s, and τ_d is a small threshold distance. This reward function encourages the vehicle to stay far from any obstacles. We could also use the latest Q-function estimate to label the last time step $t + H$, but we found this to be unnecessary to obtain good results. $Q(\mathbf{I}_t, \mathbf{a}_t)$ is learned using reinforcement learning, from the agent's own experience of navigating and avoiding collisions. Once learned, the model can be used to choose collision-free actions $\mathbf{a}_t$ simply by maximizing the Q-function. Training is performed entirely in simulation, where we can easily obtain distances to obstacles and simulate multiple different actions to determine the best one. By randomizing the simulated environment, we can train a model that generalizes effectively to domains with systematic discrepancies from our training environment. We will first describe the formulation of our model and reinforcement learning algorithm, and then present details of our simulated training environment.

3.3.1 Perception-Based Control

Our perception-based policy uses an action representation that corresponds to positions in image space. The image $\mathbf{I}_t$ is discretized into an $M \times M$ grid of bins, and each bin has a corresponding action, such that $\mathbf{a}_t$ is simply the choice of bin. Once chosen, the bin is transformed into a velocity command $\mathbf{v}_t$, which corresponds to a vector from the camera location through the image plane at the center of the bin $\mathbf{a}_t$, normalized to a constant target speed. Intuitively, choosing a bin $\mathbf{a}_t$ causes

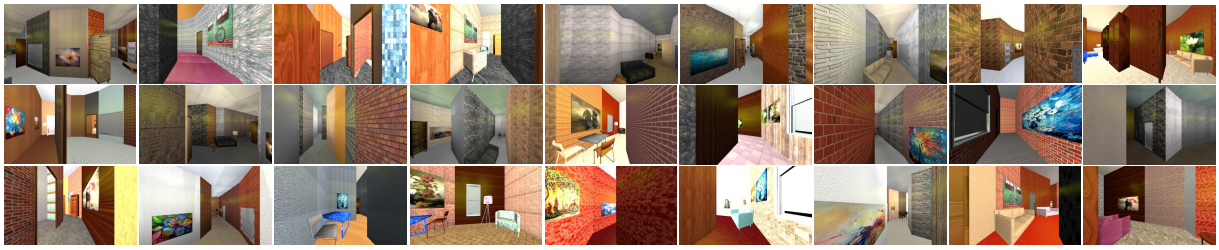

Figure 3.2: Examples of rendered images using our simulator. We randomize textures, lighting and furniture placement to create a visually diverse set of scenes.

the vehicle to fly in the direction of this bin in image space. A greedy policy can use the model $Q(\mathbf{I}_t, \mathbf{a}_t)$ to choose the action with the highest expected reward. We will use $\pi(\mathbf{I}) = \mathbf{a}$ to denote this policy.

This representation provides the vehicle with enough freedom to choose any desired navigation direction, ascend and descent to avoid obstacles, and navigate tight turns. One advantage of this image-space grid action representation is the flexibility that it provides for general navigational objectives, since we could easily choose the bin using a higher-level navigational controller, subject to the constraint that the probability of collision not exceed some user-chosen threshold. However, in order to evaluate the method in our experiments, we simply follow the greedy strategy.

3.3.2 Initialization via Free Space Detection

In order to initialize our model with a reasonable starting policy, we use a heuristic pre-training phase based on free space detection. In this pretraining phase, the model is trained to predict $P(l|\mathbf{I}_t, \mathbf{a}_t)$, where $l \in \{0, 1\}$ is a label that indicates whether a collision detection raycast in the direction $\mathbf{v}_t$ corresponding to $\mathbf{a}_t$ intersects an obstacle. The raycast has a fixed length of 1 meter. This is essentially equivalent to thresholding the depth map by one meter. This initialization phase roughly corresponds to the assumption that the vehicle will maintain a predefined constant velocity $\mathbf{v}_t$. The model, which is represented by a fully convolutional neural network as described in Section 3.3.4, is trained to label each bin with the collision label l, analogously to recent work in image segmentation [35]. The labels are obtained from our simulation engine, as described in Section 3.4.

3.3.3 Reinforcing Collision Avoidance

The initial model can estimate free space in front of the vehicle, but this does not necessarily correspond directly to the likelihood of a collision: the vehicle might be able to maneuver out of the way before striking an obstacle within 1 meter, or it may collide later in the future even if there is sufficient free space at the current time step, for example because of a narrow dead-end. We therefore use deep reinforcement learning to finetune our pretrained model to accurately represent $Q(\mathbf{I}_t, \mathbf{a}_t)$, rather than $P(l|\mathbf{I}_t, \mathbf{a}_t)$. To this end, we simulate multiple rollouts by flying through a set of training environments using our latest policy. Our score map of $M \times M$ bins, explained in 3.3.1, determines the space of actions. Based on our score map, we consider a total of M^2 actions $a = \{a^1, ..., a^{M^2}\}$

that can be taken after perceiving each observation $\mathbf{I}$. To generate the training set at each iteration, we sample a collection of states by placing the agent at a random location and with random orientation and generate a rollout of size K, given by $(\mathbf{I}_0, \mathbf{a}_0, \mathbf{I}_1, \mathbf{a}_1, ..., \mathbf{a}_{K-1}, \mathbf{I}_K)$. These states should in principle be obtained from the state distribution of the current policy. Using the model obtained from our pretraining step, the initial policy is simply $\arg\max_{i \in \{1,...,M^2\}} P(l|\mathbf{I}, a^i)$. We found that we could obtain good performance by sampling the states independently at random, though simply running the latest policy starting from an initial state distribution would also be a simple way to obtain the training states. Once the training states are obtained, we perform $M \times M$ rollouts from *each* training state using the policy π for every possible action $a^i, i \in \{1, ..., M^2\}$ and evaluate the return of a^i according to Equation (3.1). Since evaluating Equation (3.1) requires rolling out the policy for H steps for every action, we choose $H = 5$ to reduce computation costs, and instead use a simple approximation to provide smooth target values for $Q(\mathbf{I}, a^i)$.

This policy evaluation phase provides us with a dataset of observation, action, and return tuples $(\mathbf{I}_t, \mathbf{a}_t, Q(\mathbf{I}_t, \mathbf{a}_t))$, which we can use to update the policy. Since we evaluate every action for each image $\mathbf{I}_t$, the dataset consists of densely labeled images with Q values reflecting the expected sum of future rewards for the current policy π.

Our method can be interpreted as a modification of fitted Q-iteration [151], in the sense that we

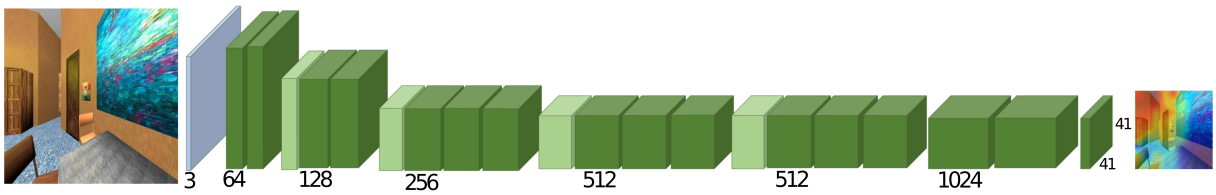

Figure 3.3: We use a fully convolutional neural network to learn the Q-function. Our network, shown above, is based on VGG16 with dilated operations.

iteratively refit a Q-function estimator to samples, as well as a variant of modified policy iteration (MPI) [148] or Monte Carlo policy evaluation, in the sense that we estimate Q-values using multi-step rollouts of the current policy. To our knowledge, ours is the first algorithm of this class to be extended to deep reinforcement learning with raw image inputs. The particular details of the approach, including the evaluation of each action at each state, are specifically designed for our simulated training setup to exploit the capabilities of the simulation and provide for a simple and stable learning algorithm. We perform rollouts in simulated training hallways. This allows us to perform multiple rollouts from the state at each time step, perform ground truth collision detection raycasts for pretraining, and removes concerns about training-time collisions. Unlike conventional RL methods that perform rollouts directly in the test environment [131], we perform rollouts in simulated training hallways. However, this also means that our model must have generalization from the simulated training hallways to real-world environments at test time. To that, we developed a randomized simulated training environment, which we describe in the next section.

3.3.4 Network Architecture

In order to represent the Q-function and the initial open space predictor, we use a deep fully convolutional neural network with dilation operations, built on the VGG16 [177] architecture following [35] as shown in Figure 3.3. The output score map corresponds to a grid of 41×41 bins, which constitutes the action space for deep reinforcement learning. The network is trained with stochastic gradient descent (SGD), with a cross-entropy loss function.

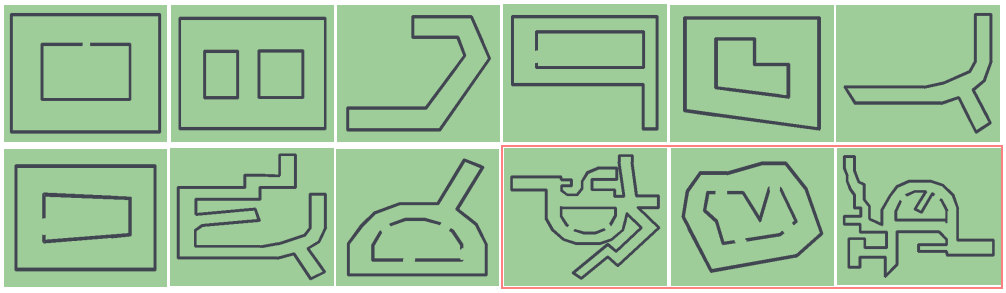

Figure 3.4: Floor plans of the synthetic hallways. The last three hallways are used for evaluation while the first 9 are used during training.

3.4 Learning from Simulation

Conventionally, learning-based approaches to autonomous flight have relied on learning from demonstration [3, 4, 147, 155]. Although the learning by demonstration approach has been successfully applied to a number of flight scenarios, the requirement for human-provided demonstrations limits the quantity and diversity of data that can be used for training. Since dataset size has been demonstrated to be critical for the success of learning methods, this likely severely limits the generalization capacity of purely demonstration-based methods. If we can train flight controllers using larger and more diverse datasets collected autonomously, we can in principle achieve substantially better generalization. However, in order to autonomously learn effective collision prediction models, the vehicle needs to see enough examples of collisions during training to build an accurate estimator. This is problematic in real physical environments, where even a single collision can lead to damage or loss of the vehicle. To get the benefits of an autonomous learning from the agent's own experience and overcome the limitations of data collection in learning from demonstration method, we use a simulated training environment that is specifically designed to enable effective transfer to real-world settings.

We manually designed a collection of 3D indoor environments to form the basis of our simulated training setup. The environments were built using the Blender [21] open-source 3D modeling suite. Our synthetic dataset contains different hallways, shown in Figure 3.4, and represent a variety of structures that can be seen in real hallways, such as long straight or circular segments with multiple junction connectivity, as well as side rooms with open or closed doors. We use furnitures

with various type and size to populate the hallways. The walls are textured with randomly chosen textures(e.g. wood, metal, textile, carpet, stone, glass, etc.), and illuminated with lights that are placed and oriented at random. In order to provide a diversity of viewpoints we render pretraining images by flying a simulated camera with randomized height and random camera orientation.

The randomization of the hallway parameters produces a very large diversity of training scenes, a sample of which can be seen in Figure 3.2. Although the training hallways are far from being photo-realistic, the large variety of appearances allows us to train highly generalizable models, as we will discuss in the experimental evaluation. The intuition behind this idea is that, by forcing the model to handle a greater degree of variation than is typical in real hallways (e.g., wide ranges of lighting conditions and textures, some of which are realistic, and some not), we can produce a model that generalizes also to real-world scenes, which might be systematically different from our renderings. That is, the wider we vary the parameters in simulation, the more likely we are to capture properties of the real world somewhere in the set of all possible scenes we consider. Our findings in this regard are aligned with the results obtained in other recent works [150], which also used only synthetic renderings to train visual models, but did not explicitly consider wide-ranging randomization of the training scenes.

3.5 Experimental Results

Despite that reinforcement learning evaluations emphasize mastery over generalization here our focus is on to evaluate the generalization capability of our proposed approach. Testing generalization is specially important from robotics perspective since the autonomous agent should be able to generalize to the diverse real-world settings. To this end, we evaluate our performance by running several experiments both in synthetic and real environments none of which had been seen during the training time. We compared our results against a set of baselines and also qualitatively evaluate our performance in various real-world scenarios. Additionally, we present an ablation study on a real-world RGB-D dataset to quantitatively evaluate our proposed randomized simulator for simulation to real-world transfer. In all the experiments (synthetic and real-world flights), CAD^{2}RL is trained on a fixed set of synthetic 3D models of hallways and in a fully simulated environment

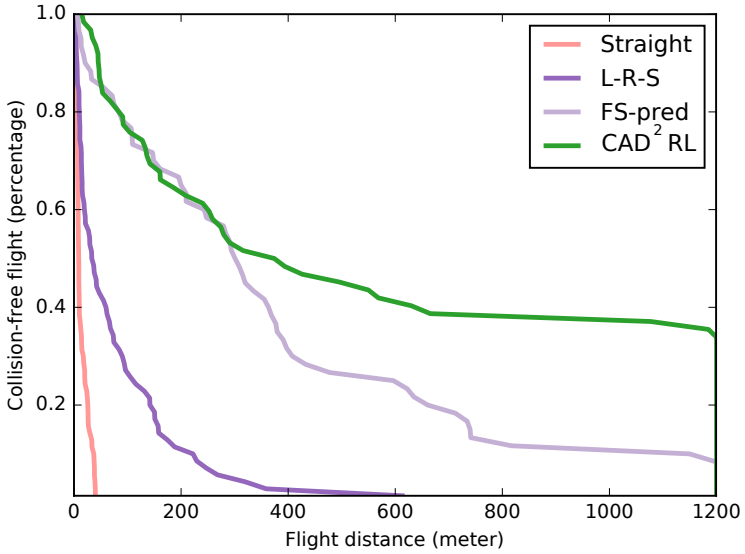

Figure 3.5: Quantitative results on a realistically textured hallway. Our approach, CAD^{2}RL, outperforms the prior method (L-R-S) and other baselines.

without being exposed to any real images.

3.5.1 Realistic Environment Evaluation

In order to evaluate how well such a model might transfer to a realistic environment, we used a realistic 3D mesh provided by [107]. Testing on this data can provide us a close proxy of our performance in a real indoor environment and also evaluates the generalization capability of our method in a systematically different environment than our training environments. Figure 3.6 shows the floorplan of this hallway, as well as several samples of its interior view. We generated 60 random initialization point from various locations in the hallways. These points are fixed and all baselines are evaluated on the same set of points so that their performance is directly comparable. Figure 3.6.a depicts the initialization points as red dots. The velocity of the quadrotor is fixed to 0.2 meters per time step in this experiment, and the maximum number of steps is set to 6000 which is equal to 1.2 kilometers.

Our aim is to evaluate the performance of our trained policy in terms of the duration of collision free flight. We run continuous episodes that terminate upon experiencing a collision, and count how many steps are taken before a collision takes place. We set the maximum number of steps to a fixed

number throughout each experiment. We evaluate performance in terms of the percentage of trials that reached a particular flight length. To that end, we report the results using a curve that plots the distance traveled along the horizontal axis, and the percentage of trials that reached that distance before a collision on the vertical axis. This provides an accurate and rigorous evaluation of each policy, and allows us to interpret for each method whether it is prone to collide early in the flight, or can maintain collision-free flight at length. Note that achieving completely collision-free flight in all cases from completely randomized initial configurations is exceptionally difficult.

In this experiment, we compare against two baselines explained below. We also report the performance of our base Free Space prediction (FS-pred) controller to analyze the improvement obtained by incorporating deep reinforcement learning. In the FS-pred, the model described in 3.3.2 is used.

Straight Controller This lower bound baseline flies in a straight line without turning. In a long straight hallway, this baseline establishes how far the vehicle can fly without any perception, allowing us to ascertain the difficulty of the initialization conditions.

Left, Right, and Straight (LRS) Controller This baseline, based on [71], directly predicts the flight direction from images. The commands are discretized into three bins: "left," "right," or "straight," and the predictions are made by a deep convolutional neural network from raw images. For training the model, prior work used real-world images collected from three cameras pointing left, right and straight that were carried manually through forest trails. We simulated the same training setup in our training environments. We finetuned a VGG16 [177] model, pretrained with ImageNet classification. This method can be considered a human-supervised alternative to our autonomous collision avoidance policy.

Quantitative Evaluation

Figure 3.5 summarizes the performance of our proposed CAD^{2}RL method compared with other baselines. Our method outperforms the prior methods and baselines by a substantial margin. Qualitatively, we found that the LRS method tends to make poor decisions at intersections, and the coarse granularity of its action representation also makes it difficult for it to maneuver near ob-

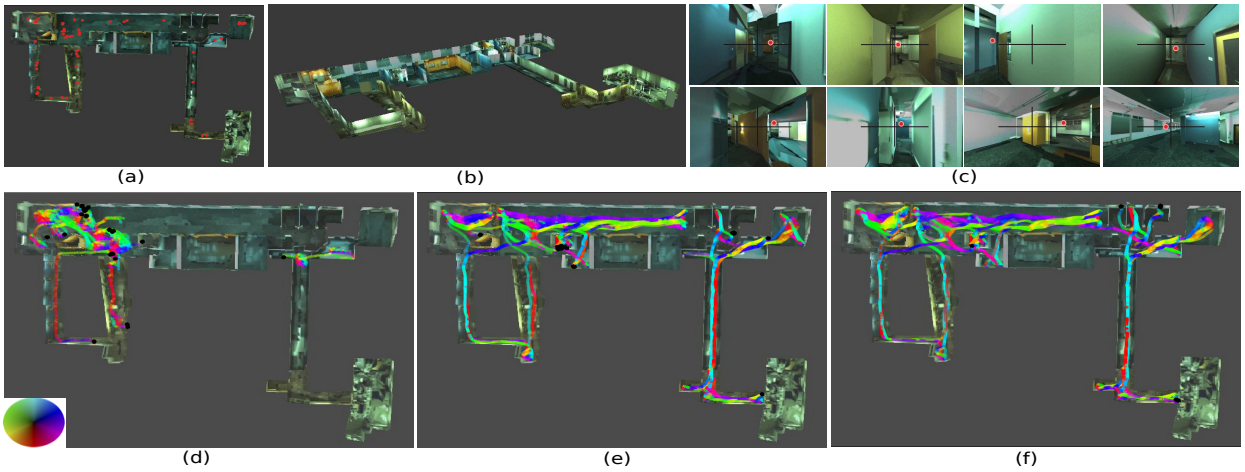

Figure 3.6: Qualitative results on a realistically textured hallway. Colors correspond to the direction of trajectory movement at each point in the hallway as per the color wheel. (a) Red dots show flight initialization points (b) Overlook view of the hallway (c) Red dots show the control points produced by CAD^{2}RL. (d) LRS trajectories (e) Perception controller (FS-pred) trajectories (f) CAD^{2}RL trajectories.

stacles. CAD^{2}RL is able to maintain a collision-free flight of 1.2 kilometers in about 40% of the cases, and substantially outperforms the model that is simply trained with supervised learning to predict 1 meter of free space in front of the vehicle. This experiment shows that although we did not use real images during training, our learned model can generalize to substantially different and more realistic environments, and can maintain collision-free flight for relatively long periods.

Qualitative Evaluation

To be able to qualitatively compare the performance and behavior of CAD^{2}RL with our perception based controller and the LRS method, we visualized the trajectory of the flights overlaid on the floor-plan of the hallway as shown in Figure 3.6. For this purpose, we sorted the trajectories of each method based on the traveled distance and selected the top 25 longest flights from each method. The trajectory colors show the flight direction at each point. The black dots indicate the locations of the hallway where collisions occurred. This visualization shows that CAD^{2}RL could maintain a collision-free flight in various locations in the hallway and has fewer collisions at the dead-ends, corners, and junctions compared with the other two methods. LRS often experienced collisions in corners and is more vulnerable to bad initial locations. The policy trained with free space prediction outperformed the LRS method, but often is trapped in rooms or fail near junctions

and corners. This illustrates that the controller trained with RL was able to acquire a better strategy for medium-horizon planning, compared to the directly supervised greedy methods.

3.5.2 Real World Flight Experiments

We evaluated our learned collision avoidance model by flying a drone in real world indoor environments. These flights required flying through open spaces, navigating hallways, and taking sharp turns, while avoiding collisions with furniture, walls, and fixtures. We used two different drone platforms: the Parrot Bebop 1.0 and the Bebop 2.0, both controlled via the ROS Bebop autonomy package [134]. We perform real-world flight in several different scenarios and evaluate our performance both quantitatively and qualitatively.

Details on real environment tests

We used the Parrot Bebop drone controlled via the ROS Bebop autonomy package [134]. We ran real flight experiments using two different drone platforms: the Parrot Bebop 1.0 and the Bebop 2.0. Although these two drones have similar SDK, they have different physical specifications in terms of dimensions, weight, and maximum speed. Note that the images produced by the onboard drone camera are center cropped to remove the fish-eye effect, and thus the objects appear closer than they really are.

Figure 3.10 shows the sequence of images captured by the flying drone in various scenarios (for the more complete sequences please check the supplementary video https://youtu.be/nXBWmzFrj5s). The red dots show the action computed by our policy. Our controller can successfully navigated the drone throughout free spaces while avoiding collision. Due to imperfect stabilization, turbulence and air currents, the drone may sometimes drift to the left or right, and our controller can recover from these situations by stabilizing the drone at each time step based on the current image observation. This suggests that, though our model is fully trained in simulation without seeing any real images or using any human provided demonstration, it has the capability to generalize to real-world images and conditions. In the following sections we evaluate the real flight performance

Table 3.1: Real world flight results.

Environment	Traveled Distance (meters)	Travel Time (minutes)	Collision (per meter)	Collision (per minute)	Safe Flight (meters)	Safe Flight (minutes)	Total Collisions
Cory FS-pred	162.458	12.01	0.080	1.081	12.496	0.924	13
Cory CAD^{2}RL	163.779	11.950	0.0366	0.502	27.296	1.991	6
SDH FS-pred	53.492	4.016	0.130	1.742	7.641	0.573	7
SDH CAD^{2}RL	54.813	4.183	0.072	0.956	13.703	1.045	4

both quantitatively and qualitatively.

Quantitative Evaluation

For quantitative evaluation, we ran controlled experiments on the task of hallway following. We fixed all the testing conditions while navigating the drone with either of the CAD^{2}RL and a baseline controller. The testing conditions include the initial velocity, angular speed, drone platform and the test environment. As was concluded from the experiments in section 3.5.1, FS-pred was the strongest baseline, and we therefore included it as a comparison in this experiment. We ran experiments in two different buildings, Cory Hall and SDH (Sutardja Dai Hall), both located on the UC Berkeley campus. These buildings have considerably different floor plans, wall textures, and lighting conditions, as can be seen in Figure 3.10.c and Figure 3.10.d. Our testing environment in Cory Hall contained three turns and two junctions, while the SDH test environment had one turn and one junction. The width of the Cory hall hallway is ~ 3 meters while the SDH hallway is ~ 2 meters wide.

Table 6.3 summarizes the results. The safe flight time is given by the average length of a collision free flight in terms of distance or time between collisions. CAD^{2}RL experienced fewer collisions and has longer expected safe flight. This suggests that the CAD^{2}RL policy makes fewer mistakes and is more robust to perturbations and drift. Both methods performed better in Cory, since SDH has narrower hallways with glossy textureless walls as well as stronger air currents. While we fixed the test environment and the flying speed, the traveled distance and time is slightly different from one algorithm to another due to the fact that the algorithms generated different commands and navigated the drone to slightly different locations in the hallways.

Qualitative Evaluation

We performed real world flight in various indoor scenarios. We briefly explain each scenario and sequence snapshots are shown in Figure 3.10.

(a) Flying near furniture, around corners, and through a window: As shown in Figure 3.10.a. the drone starts from one end of a hallway connected to a small lounge area with furniture. The drone first flies toward the open lounge area, and then turns toward a corner of the room. There, it detects an opening in the wall which is visually similar to an open doorway or window, and adjust its height to fly through it. The drone then encounters a reflective glass door, which reflects the hallway behind it. Since no such structures were present during training, the reflective door fools the controller, causing the drone to crash into the door. Note that the controller navigates multiple structures that are substantially different, both visually and geometrically, from the ones encountered during simulated training.

(b) Flying up a staircase: Here, our goal is to evaluate the generalization capability of the controller to changes in elevation. A staircase provides a good example of this. To avoid colliding with the stairs, the drone must continuously increase altitude. As can be seen from the snapshots in the Figure 3.10.b, the controller produces actions that increase the altitude of the drone at each step along the staircase. Since we used an altitude limit for safety reasons, the drone only flew halfway up the staircase, but this experiment shows that the controller could effectively generalize to structures such as staircases that were not present during training.

(c) Navigating through narrow corridors: In this scenario, the drone flies through a corridor. The drone successfully takes a turn at Frames 1-4 in Figuree 3.10.c to avoid flying into a dead end. The corridors in this test scenario are narrow (~ 2 meters) and have strong air currents due to air conditioning.

(d) Flying through junctions and rooms: Here, the drone navigates through a long hallway with junctions. At the end it enters a doorway which is connected to a study room. The controller successfully navigates the drone through the narrow door and into the room without colliding with the chairs.

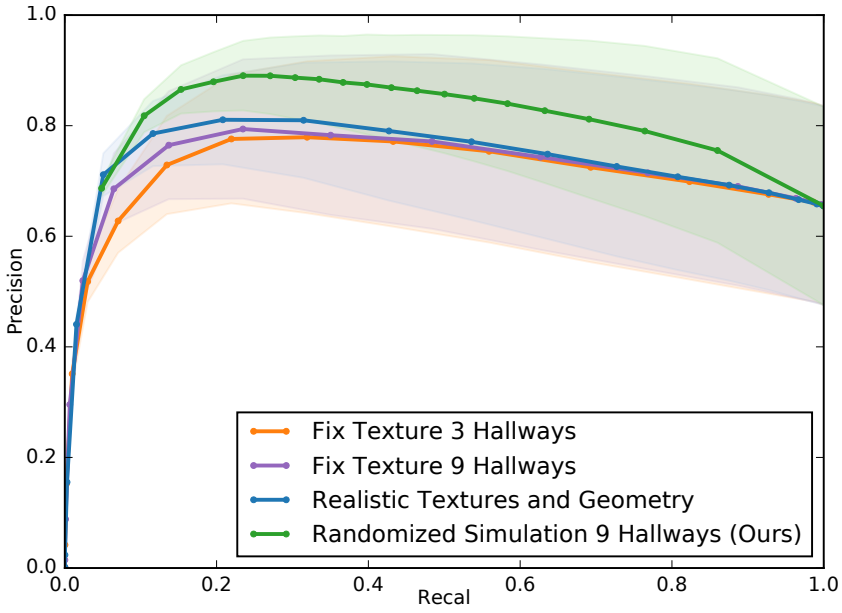

Figure 3.7: Quantitative results for free-space prediction with different simulators. The network trained on randomized hallways outperforms networks trained on less randomized simulations and even on realistic textured hallways.

(e) Flying through a maze of random obstacles in a confined space: We built a small U-shaped maze out of low obstacles in the lab. This maze is built using chairs and pieces of board with various appearance and colors. To prevent the drone from simply flying over the obstacles, we limited the altitude to 3 feet. Note that flying the drone at low altitude is challenging, as the air turbulence becomes significant and affects the drone's stability. The cardboard shifts due to air turbulence, and the open area is very narrow ($\sim$ 1 meter), making this a challenging test environment. The sequence in Figure 3.10.e shows that the controller successfully navigates the drone throughout the maze, making a turn near the red chair and turning back into the maze, without colliding.

(f) Avoiding dynamic obstacles: In this scenario, the drone begins in the lab with no obstacles and an altitude of around 3 feet. We then place a chair in the path of the drone, as seen in frames 3-4 of Figure 3.10.f. The controller recovers and avoids an imminent collision with the chair, passing it on the left.

The above qualitative evaluation study shows the generalization capability of our trained model and demonstrates the extent of the maneuvering skills learned by CAD^{2}RL. Although our model is

specifically trained for the task of hallway navigation, the limited number of furniture items present in simulation also force the policy to be robust to oddly shaped obstacles, and train it to change altitude to avoid collisions. Navigating through the obstacles in the scenarios (a), (b), (e), and (f) required collision avoidance with general obstacles and other than just walls. We observed that our model could perform reasonably well in these cases, and could often recover from its mistakes, though particularly novel situations proved confusing.

3.5.3 Ablation Study for Real World Transfer

In this section, we present an ablation study to identify how important the randomization of the environment is for effective simulation to real-world transfer. Since conducting statistically significant real-world flight trials for many training conditions is time-consuming and subject to confounding factors (air currents, lighting conditions, etc.), we instead opted for a proxy task that corresponds to free-space prediction from real RGB images, with ground truth labels obtained via a depth camera. The goal in this task is to predict, for each point in the image, whether there is an obstacle within a certain threshold distance of the camera or if the pixel corresponds to free space. Although this proxy task does not directly correspond to collision-free flight, the reduced variance of the evaluation (since all methods are tested on exactly the same images) makes this a good choice for the ablation study. While we obtained reasonably good performance for avoiding collisions in the hallways, more detailed depth estimation [169, 119, 118] could also be used without loss of generality.

We used the same architecture as in Section 3.3.2 for the free-space prediction network and trained free-space predictors using rendered images from different simulated setups. We compared the obtained results against a similar network trained using our proposed randomized simulation. We used the same number of images sampled similarly from various locations in the hallways. The ablated networks are trained with images rendered from (a) a simulator that used **F**ixed **T**extures and lighting, and only 3 of our training hallways (FT3); (b) **F**ixed **T**extures and lighting using all 9 training hallways (FT9) (c) the more **R**ealistic **T**extures and **G**eometry hallway provided by [107] (RTG) and (d) our approach, with randomized textures and lighting from 9 hallways. While (a),

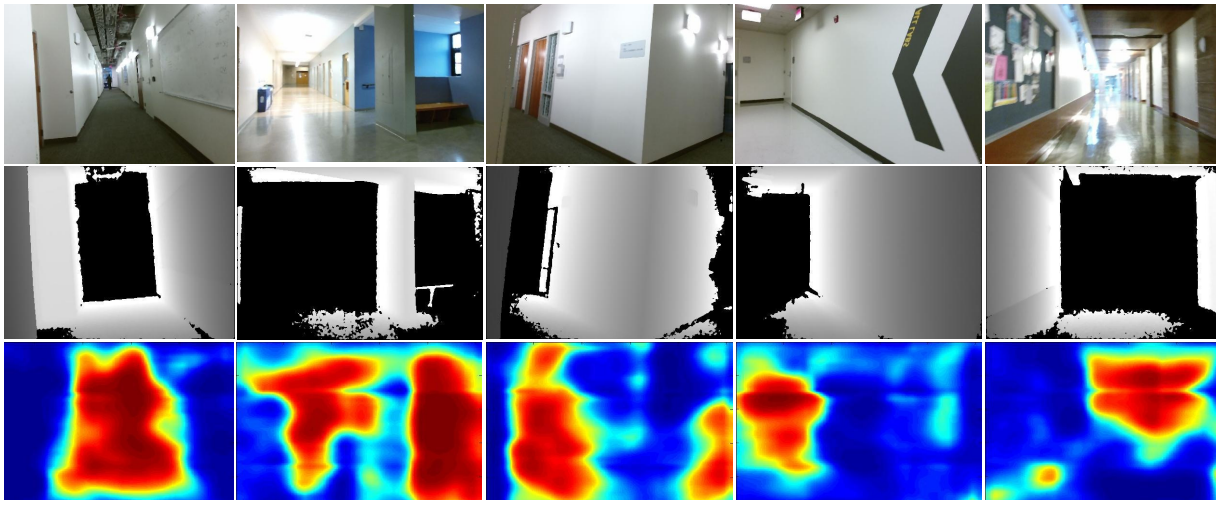

Figure 3.8: Examples of the collected pairs of RGB (top row) and depth (mid row) data for the free-space test set. The free-space probability map predicted by our approach is shown in the bottom row.

(b), and (d) are captured from synthetic hallways, in (c) the data is captured via a SLAM-based reconstruction system from the Cory Hall in the UC Berkeley campus. Therefore, this data has realistic geometry textured with natural images, and allows us to understand how the method would perform if trained on reconstructed RGBD data.

Our dataset contains RGB-D images captured from 5 hallways, in Cory Hall and SDH (Sutardja Dai Hall) located in UC Berkeley, with various lighting and texture conditions. We used a Kinect v2 and our dataset contains a total of 620 RGB-D images. Several example images of this data are shown in Figure 3.8. We used the depth channel to automatically annotate the images with free-space vs. non-free-space labels.

For each pixel in the input image, the network produces a probability value for free-space prediction. To evaluate the accuracy of free-space prediction we sweep a threshold from 0 to 1 to label each pixel using our prediction network. We compute the precision and recall at each threshold and plot the precision-recall curve as the performance metric. Precision is the number pixels correctly labeled as free-space divided by the total number of pixels predicted as free-space, while recall is the the number pixels correctly labeled as free-space divided by the total number pixels belonging to the free-space according to the ground truth. Since we use monocular images, there is a scale ambiguity in the size of hallways as well as in the range of sensible depths, which may not match

between the simulated and real images. To overcome this ambiguity and to make a fair comparison, we labeled image pixels (for free-space vs non-free-space) by varying the depth threshold from 1 to 4 meters (steps of $\sim 30cm$) and computed the average precision/recall corresponding for each threshold over 13 runs.

Figure 3.7 shows the results, with shaded areas showing the standard deviation in precision. The network trained with the synthetic data rendered by our proposed randomized simulator outperforms the other networks. The images used for FT3 and FT9 are rendered on the same hallways as RT9, except that the textures and lighting are not randomized. As a result, these networks do not learn texture and color invariant features and cannot generalize well to the real images. In RTG, the images are rendered with realistic geometry and textures, and thus they are less affected by the scale ambiguity. Furthermore, the realistic textures in RTG are obtained from similar hallways as the one we used for our RGB-D test set. Despite this, the network trained on a realistic rendering of the same hallway actually performs worse than the network trained on our randomized simulator, by a substantial margin. For qualitative analyis, we show the probability map of free-space prediction obtained from our approach in the last row of Figure 3.7. We see that high probabilities are assigned to free spaces. Although the free-space prediction proxy task is not a perfect analogue for collision-free flight, these results suggest that randomization is important for good generalization, and that more realistic renderings should not necessarily be preferred to ones that are less realistic but more diverse.

3.5.4 Synthetic Environment Test

Here we provide several more experiments and ablation study to further evaluate the generalization capability of CAD^{2}RL in unseen environment. Also, we present more details of various experiments as well as more details of our simulator here.

We conduct experiments on the three test mazes shown in Figure 3.4. This experiment is aimed at comparing CAD^{2}RL in the presence of different hallway geometries, distractors, and obstacles, in synthetic hallways that are distinct from the ones used but similar in visual style. Note that the test hallways were intentionally designed to be larger and more challenging. We rendered all

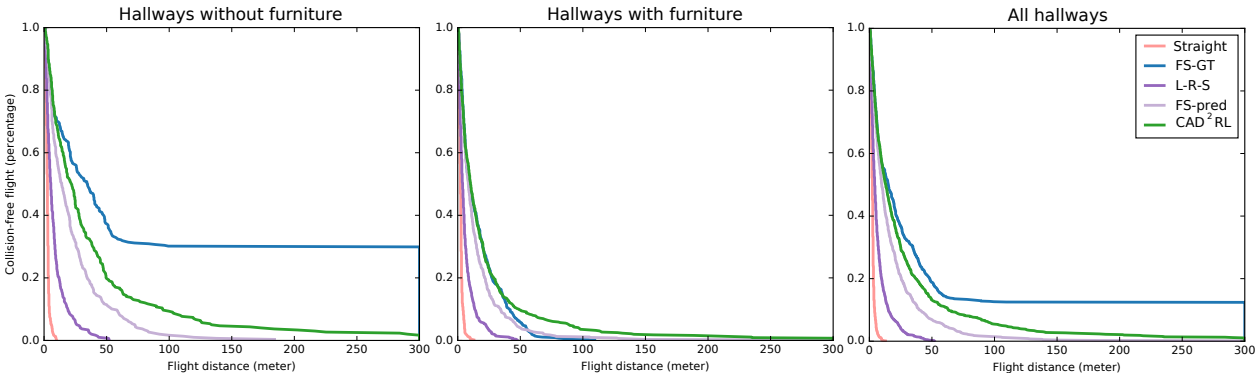

Figure 3.9: Quantitative results on the simulated test hallways. Aside from the upper bound baseline FS-GT, which uses ground truth collision raycasts, our method CAD^{2}RL, achieves the longest collision-free flights.

images at the test time using a randomization of 100 different test textures that were not seen at the training time. We tested our method and the prior baseline methods in two conditions: in the first condition, the hallways contained randomly placed furniture, and in the second, no furniture was present. Both scenarios have fixtures such as open or closed doors, windows, and paintings, but the hallways with furniture provide an additional challenge due to the more complex geometry, which is substantially more elaborate than the scanned hallway used in the previous section.

Evaluation criteria

Details about our Simulator Setup

We designed a total of 24 different hallways using 12 different floorplans. Some of our hallways are populated with furnitures while some of them are not. We use 21 different items of furniture that commonly exist in the hallways (benches, chairs, etc). We have slightly randomized the size of these furniture to provide diversity. The walls are textured with randomly chosen textures, chosen from a pool of 200 possible textures (e.g. wood, metal, textile, carpet, stone, glass, etc.), and illuminated with lights that are placed and oriented at random. Pretraining images are generated by flying a simulated camera through the hallways with randomized height and random perturbations of the yaw angle, in order to provide a diversity of viewpoints. In all, we randomize textures, lighting, furniture placement (including placement and identity of furniture items), and camera position and angle.

Table 3.2: Pixel-wise free space prediction results.

	Precision	Jaccard(FS)	Jaccard(O)
Synthetic hallways	90.38	56.99	87.26
Realistic hallways	80.33	63.39	63.30

Our aim is to evaluate the performance of our trained policy in terms of the duration of collision free flight. To do this, we run continuous episodes that terminate upon experiencing a collision, and count how many steps are taken before a collision takes place. We set the maximum number of steps to a fixed number throughout each experiment.

An episode can begin in any location in the choice environment, but the choice of the initial position can have a high impact on the performance of the controller, since some parts of the hallway, such as dead ends, sharp turns, or doorways, can be substantially more difficult. Therefore, to make an unbiased evaluation and to test the robustness of the learned policies, we start each flight from a location chosen uniformly at random within the free space of each environment. We use random initialization points and keep them fixed throughout all experiments to provide a fair comparison between different methods, including prior work. In the experiments, the quadrotor has constant velocity during each flight, and we convert the number of steps to meters in order compute the distance traveled in each flight.

We evaluate performance in terms of the percentage of trials that reached a particular flight length. To that end, we report the results using a curve that plots the distance traveled along the horizontal axis, and the percentage of trials that reached that distance before a collision on the vertical axis. More formally, vertical axis represent $\sum_{i=1}^{|T|} (1 - \mathbb{1}(C_d(T_i)))$ where T denotes a trial and $C_d(T_i)$ is the event of collision happening for the ith trail at distance d in the x axis. This provides an accurate and rigorous evaluation of each policy, and allows us to interpret for each method whether it is prone to collide early in the flight, or can maintain collision-free flight at length. Note that achieving completely collision-free flight in all cases from completely randomized initial configurations is exceptionally difficult. Please note that we used the same evaluation metric in the quantitative experiment in section 3.5.1.

We compare the performance of our method with previous methods for learning-based visual obstacle avoidance introduce in Section 3.5.1. In addition, we compare against *ground truth free space controller* baseline which simulates a quadrotor equipped with perfect LIDAR range sensors or depth cameras. The performance of this baseline shows the upper-bound of the performance of a free-space prediction based controller. The policy always selects the most central free-space labeled bin in the spatial grid of the current image. Note that this does not always result in the best performance, since the behavior of this baseline is myopic, but it does serve to illustrate the difficulty of the test environments.

Results and Analysis

We randomly sampled 100 random locations as initialization point. These points were selected uniformly from challenging locations such as junctions as well as less challenging locations in the middle of hallways. The velocity was 0.3 meters per step. Figure 3.9 compares the performance of our method compared with other baselines in each of the "with furniture" and "without furniture" test scenarios. CAD^{2}RL consistently outperforms the other baselines, as well as the model trained with supervised learning for free space prediction, FS-pred. In the hallways that do not contain any furniture, the ground truth free space baseline (FS-GT) obtains the best performance, while the presence of furniture in the second test scenario effectively reduce its performance due to the greedy strategy. In both scenarios, CAD^{2}RL has the highest performance among the learning-based methods.

3.5.5 Free Space Prediction Evaluation

In this experiment, we are interested to see how well our free space prediction model can detect free spaces and obstacles compared with its performance on the synthetic images. To this end, we used the simulator to compute the mask of Free Space (FS)/Obstacle(O) of 4k rendered frames which were sampled uniformly along the hallways. For the performance metrics, we use precision and Jaccard similarity. The precision shows the ratio of correctly labeled pixels as corresponding

to “FS” or “O”. The Jaccard similarity is the intersection over union of the result and ground truth labels for both free-space and obstacle labels. Table 3.5.4 summarizes the obtained results. The first row of the table shows the results obtained for the images rendered in from our test hallways with test textures. The second row shows the results obtained on the photo realistic images of [107]. Although, there is a 10% precision gap between the performance on synthetic images and photo-realistic images, which is due to the domain shift, the obtained precision results on the unseen photo-realistic images is high, i.e. 80%. Note that our synthetic hallways are much narrower than the real hallways of [107]. This results in smaller free-space areas and larger obstacle areas in the synthetic images compared with [107] where images have more balanced distribution of free-space vs obstacles. This results in lower Jaccard(FS) in the synthetic images.

3.6 Discussion

In this chapter, we presented a method for training deep neural network policies for obstacle avoidance and hallway following, using only simulated monocular RGB images. We described a new simple and stable deep reinforcement learning algorithm for learning in simulation. We also demonstrate that training on randomized simulated scenes produces a model that can successfully fly and avoid obstacles in the real world, and quantitatively evaluated our randomized scenes on a proxy free-space prediction task to show the importance of randomization for real-world transfer. Our simulated evaluation further shows that our method outperforms several baselines, as well as a prior end-to-end learning-based method. Our aim here is to evaluate the potential of policies trained *entirely* in simulation to transfer to the real world, so as to understand the benefits and limitations of simulated training. To attain the best results in real environments, future work could combine simulated training with real data. Extending our approach via finetuning or domain adaptation is therefore a promising direction for future work that is likely to improve performance substantially, and lead to effective learned real-world visual navigation policies using only modest amounts of real-world training. Our approach could incorporate data from other sensors, such as depth cameras, which should improve the performance of the learned policies.

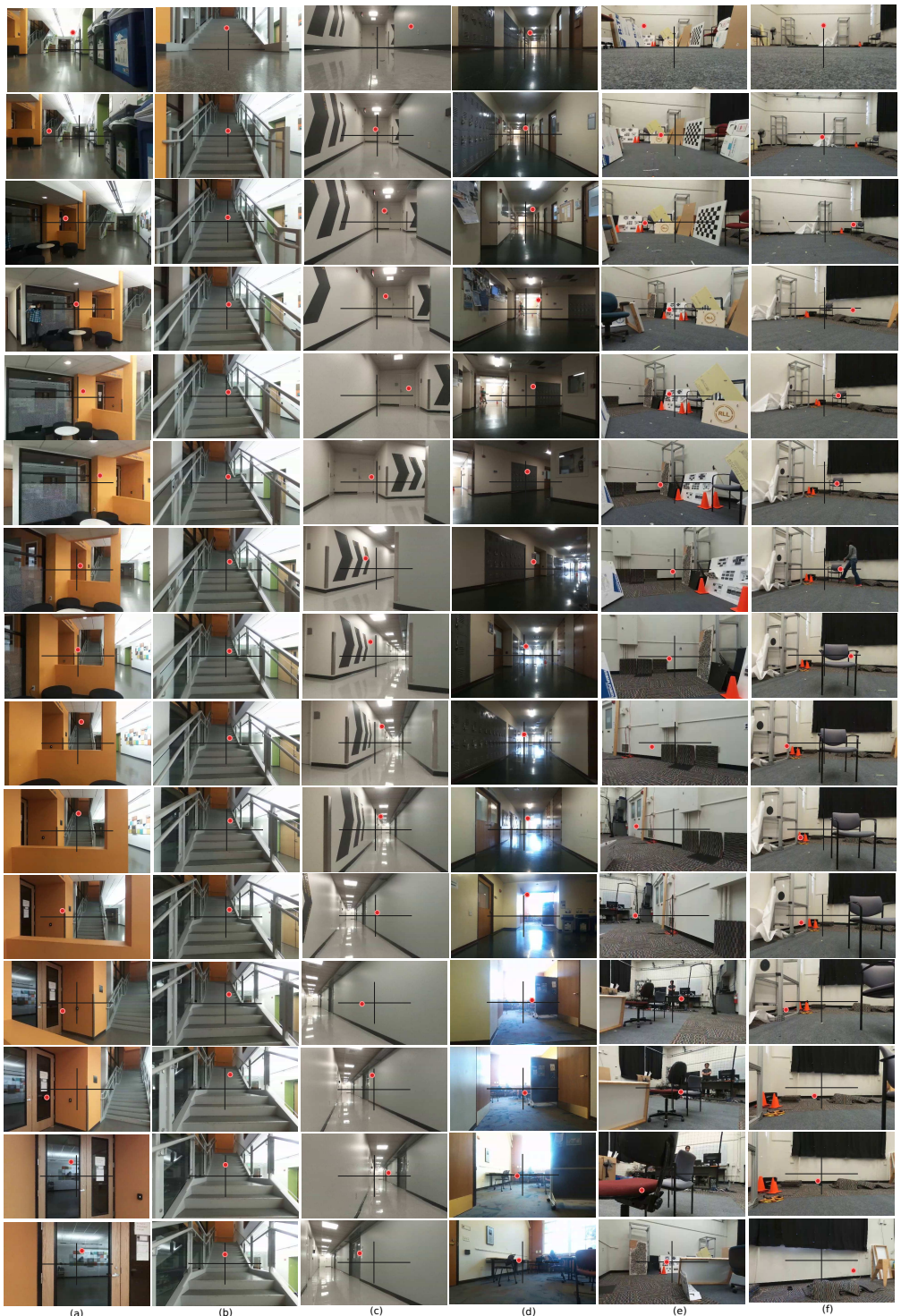

Figure 3.10: Snapshots of autonomous flight in various real indoor scenarios. Frames ordered from top to bottom. Red dots show the commanded flight direction by CAD^{2}RL. (a) Flying near furniture, around corners, through a window; (b) Flying up a staircase; (c) Navigating in narrow corridors; (d) Navigating through junctions, fly through rooms; (e) Flying through a maze of random obstacles in a confined space; (f) Avoiding dynamic obstacles.

Chapter 4

SEMANTIC VISUAL SERVOING

"Begin at the beginning," the King said, very gravely, " and go on till you come to the end: then stop. "

Lewis Carroll, *Alice in Wonderland*

Robots should understand both semantics and physics to be functional in the real world. While robot platforms provide means for interacting with the physical world they cannot autonomously acquire object-level semantics without needing human. In this chapter, we investigate how to minimize human effort and intervention to teach robots perform real world tasks that incorporate semantics. We study this question in the context of visual servoing of mobile robots and propose DIViS, a ***D***omain ***I***nvariant policy learning approach for collision free ***Vi***sual ***S***ervoing. DIViS incorporates high level semantics from previously collected static human-labeled datasets and learns collision free servoing entirely in simulation and without any real robot data. However, DIViS can directly be deployed on a real robot and is capable of servoing to the user-specified object categories while avoiding collisions in the real world. DIViS is not constrained to be queried by the final view of goal but rather is robust to servo to image goals taken from initial robot view with high occlusions without this impairing its ability to maintain a collision free path. We show the generalization capability of DIViS on real mobile robots in more than 90 real world test scenarios with various unseen object goals in unstructured environments. DIViS is compared to prior approaches via real world experiments and rigorous tests in simulation. For supplementary videos, see: https://fsadeghi.github.io/DIViS

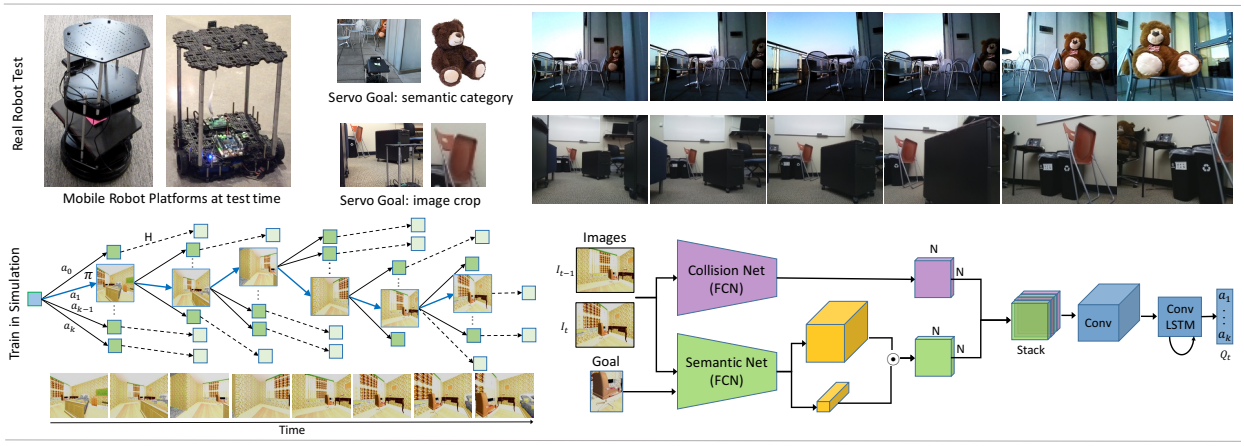

Figure 4.1: Domain Invariant Visual Servoing (DIViS) learns collision-free goal reaching entirely in simulation using dense multi-step rollouts and a recurrent fully convolutional neural network (top). DIViS can directly be deployed on real physical robots with RGB cameras for servoing to visually indicated goals as well as semantic object categories (bottom).

4.1 Overview

Perception and mobility are the two key capabilities that enable animals and human to perform complex tasks such as reaching food and escaping predators. Such scenarios require the intelligent agent to make high level inference about the physical affordances of the environment as well as semantics to recognize goals and distinguish walkable areas from the obstacles to take efficient actions towards the goal. Visual Servoing (VS), is a classic robotic technique that relies on visual feedback to control the motion of a robot to a desired goal which is visually specified [89, 42, 34, 207]. Historically, VS has been incorporated for both robotic manipulation and navigation [89, 42, 18, 30, 95]. Figure 4.2 depicts an early visual servoing mobile robot in 1995 that servos to a goal image by matching geometric image features between the view at the final desired position and robot's current camera view [145, 54]. While visual servoing mechanisms aim to acquire the capability to move in the 3D world, they inherently do not incorporate object semantics. In addition, conventional servoing mechanisms need to have access to the robot's view in its final goal position. Such requirement, can restrict the applicability as it may not be possible to have the camera view at the goal position specially in unstructured real world environments.

Deep learning has achieved impressive results on a range of supervised semantic recognition

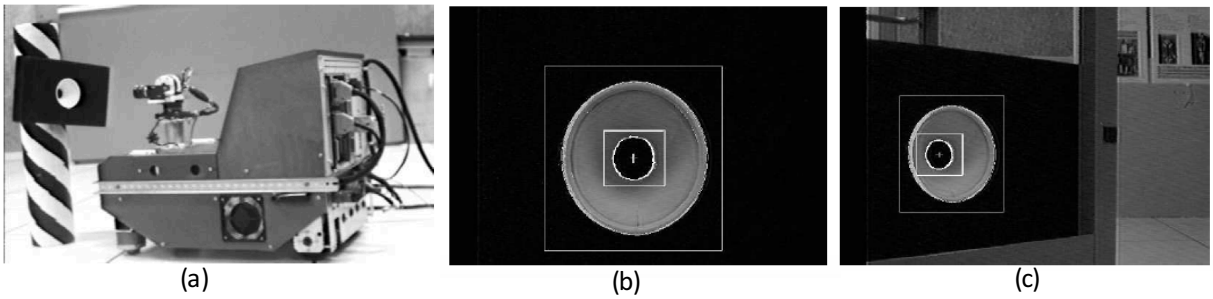

Figure 4.2: (a) The classic 1995 visual servoing robot [145, 54]. The image at final position (b) was given as the goal and the robot was started from an initial view of (c).

problems in vision [105, 70] language [127] and speech [82, 73] where supervision typically comes from human-provided annotations. In contrast, deep reinforcement learning (RL) [131, 171, 130] have focused on learning from experience, which enables performance of physical tasks, but typically do not focus on widespread semantic generalization capability needed for performing tasks in unstructured and previously unseen environments. Robots, should be capable of understanding both semantics and physics in order to perform real world tasks. Robots have potential to autonomously learn how to interact with the physical world. However, they need human guidance for understanding object-level semantics (e.g., chair, teddy bear, etc.). While it is exceedingly time-consuming to ask human to provide enough semantic labels for the robot-collected data to enable semantic generalization, static computer vision datasets like MS-COCO [117] or ImageNet [105] already offer this information, but in a different context. In this work, we address the question of how to combine autonomously collected robot experience with the semantic knowledge in static datasets, to produce a semantic aware robotic system.

We study this question in the context of goal reaching robotic mobility in confined and cluttered real-world environments and introduce DIViS, **D**omain **I**nvariant **Vi**sual **S**ervoing that can maintain a collision-free path while servoing to a goal location specified by an image from the initial robot position or a semantic object category. This is in contrast to the previous visual servoing approaches where the goal image is taken in the final goal position of the robot. Performing this task reflects robots capability in understanding both semantics and physics; The robot must be able to have sufficient physical understanding of the world to reach objects in cluttered rooms

without colliding with obstacles. Also, semantic understanding is required to associate images of goal objects, which may be in a different context or setting, with objects in the test environment. In our setup, collisions terminate the episode and avoiding obstacles may necessitate turning away from the object of interest. This requires the robot to maintain memory of past movements and learn a degree of viewpoint invariance, as the goal object might appear different during traversal than it did at the beginning of the episode.

The main contribution of our work is a novel domain invariant policy learning approach for direct simulation to real world transfer of visual servoing skills that involve object-level semantics and 3D world physics such as collision avoidance. We decouple physics and semantics by transferring physics from simulation and leveraging human annotated real static image datasets. Therefore, our approach addresses the challenges and infeasibilities in collecting large quantities of semantically rich and diverse robot data. To keep track of changes in viewpoint and perspective, our visual servoing model maintains a short-term memory via a recurrent architecture. In addition, our method predicts potential travel directions to avoid colliding with obstacles by incorporating a simple reinforcement learning method based on multi-step Monte Carlo policy evaluation. We conduct wide range of real-world quantitative and qualitative evaluations with two different real robot platforms as well as detailed analysis in simulation. Our policy which is entirely trained in simulation can successfully servo a real physical robot to find diverse object instances from different semantic categories and in a variety of confined and highly unstructured real environments such as offices and homes.

4.2 Related Work

Visual servoing techniques rely on carefully designed visual features and may or may not require calibrated cameras [42, 54, 202, 208, 95]. Our method does not require calibrated cameras and does not rely on geometric shape cues. While photometric image-based visual servoing [30] aims to match a target image, our approach can servo to goals images which are under occlusion or partial view. Recently several learning based visual servoing approaches using deep RL are proposed for manipulation [115, 108, 164], tracking [112] and image-based navigation via visual imitation

Figure 4.3: Examples of the diverse goal-reaching mobility tasks. In each task, agent should reach a different visually indicated goal object. In each scenario, we use domain randomization for rendering.

learning in real world [140] or deep reinforcement learning in simulation [218]. In contrast to the the goal-conditioned navigation methods of [140, 218], our approach uses the image of goal object from the initial view of the robot which can be partially viewed or heavily occluded. Also, our method learns a domain invariant policy that can be transferred to the real world despite the fact that it is entirely trained in simulation.

Transferring from simulation to real and bridging the reality gap has been an important area of research for a long time in robotics. Several early works include using low-dimensional state representations [135, 96, 46, 185]. Given the flexibility and diversity provided by the simulation environments, learning policies in simulation and transferring to the real world has recently gained considerable interest in vision-based robotic learning [162, 164, 152, 97, 23]. Prior works on representation learning either learn transformation from one domain to another [72, 158] or train domain invariant representations for transfer learning [120, 24, 66]. [213] proposed inverse mapping from real images to renderings using CycleGAN [216] and evaluated on a self-driving application and an indoor scenario. Domain randomization in simulation for vision-based policy learning was

first introduced in [162] for learning generalizable models that can be directly deployed on a real robot and in the real world environments. It was then broadly used for various robot learning tasks [164, 97, 190, 9, 143, 188, 142, 136, 181]. In contrast, our focus is on combining transfer from simulation (for physical world understanding) with transfer from semantically labeled data (for semantic understanding). The physical challenges we consider include navigation and collision avoidance, while the semantic challenges include generalization across object instances and invariance to nuisance factors such as object background and viewpoint. To our knowledge, direct simulation to real world transfer for semantic object reaching with diverse real-world goals and environments is not explored before.

Visual navigation is a closely related problem to our work. A number of prior works have explored navigation via deep reinforcement learning using recurrent policy, auxiliary tasks and mixture of multiple objectives in video game environments with symbolic targets that do not necessarily have the statistics of real indoor scenes [130, 50, 129, 94]. We consider a problem setup with realistic rooms populated with furniture and our goal is defined as reaching specific objects in realistic arrangements. Vision-based indoor navigation in simulation under grid-world assumption is considered in several prior works [218, 76]. [212] addressed visual navigation via deep RL in real maze-shaped environment and colored cube goals without possessing semantics of realistic indoor scenes. [168] benchmarked basic RL strategies [50, 129, 130]. With the increasing interest in learning embodied perception for task planning and visual question answering, new simulated indoor environments have been developed [168, 203, 179, 8, 64]. While these environments facilitate policy learning in the context of embodied and active perception for room navigation, they do not consider generalization to the real world with a physical robot. As opposed to all aforementioned prior works, the focus in our work is on transfer of both semantic and physical information, with the aim of enabling navigation in real-world environments. In our work, we do not focus on long-term navigation (e.g., between rooms) or textual navigation [8], but we focus on local challenges of highly confined spaces, collision avoidance, and searching for occluded objects which require joint understanding of physics and semantics.

Simultaneous localization and mapping (SLAM) had been used for localization, 3D reconstruc-

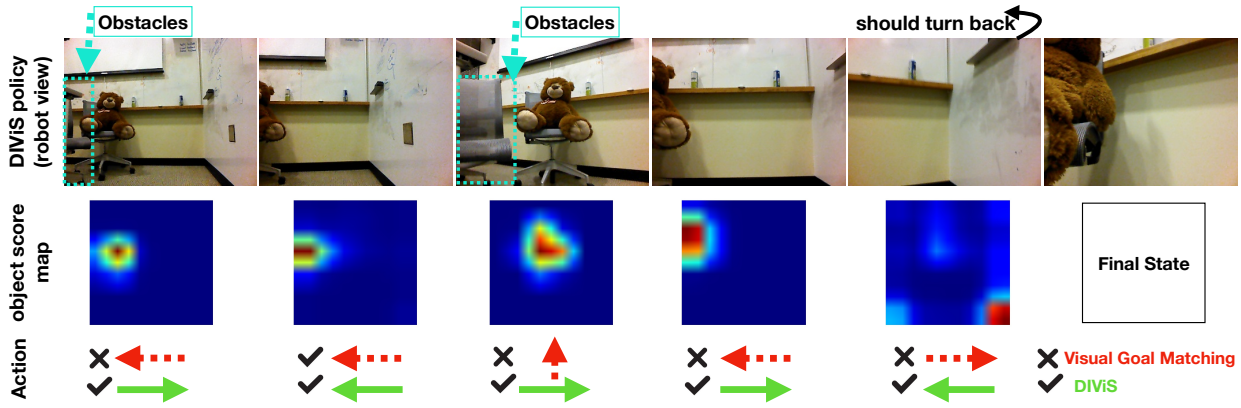

Figure 4.4: Qualitative comparison of DIViS against Visual Goal Matching policy while the robot is tasked to reach "teddy bear". *Green arrows* show action directions taken by DIViS and *red arrows* (bottom row) show action directions chosen by Visual Goal Reaching which only relies on object score maps (middle row). Visual Goal Matching fails by colliding into obstacles while DIViS reaches the goal successfully by taking turns around the obstacles (top row).

tion/mapping as well as path planning. One group of SLAM methods rely on combining a pipeline of different algorithms to first build a map of the environment using geometry constraints. Then, they perform local or global path planning and control to navigate to a designated goal in the estimated 2D or 3D map. Another group, aims to automatically traverse and build a representation for a new environment simultaneously and can use local path planning for navigation to a desired goal. While these approaches provide promising solutions for building the 3D structure or map of the environment for navigation, their main focus is not on solving visually indicated goal reaching. There is a large body of work on SLAM approaches, for which we refer the reader to these comprehensive surveys [187, 27].

4.3 Diverse Collision Free Goal Reaching

We build a new simulator for the collision-free visual goal-reaching task using a set of 3D room models with diverse layouts populated with diverse furniture placements. Our goal is to design diverse tasks that require the agent to possess both semantic and physical understanding of the world. Figure 4.3 depicts examples of our tasks. In each example, the agent is tasked to get as close as possible to a visually indicated goal while avoiding collisions with various obstacles. The map of the environment is unknown for the agent and the agent is considered to be similar to a

ground robot which can only move in the walkable areas. This scenario is aligned with real-world settings where we have diverse environments populated with variable furniture. The furniture and room objects are not registered in any map, and the robot should be able to move towards objects without collision, since otherwise it can damage itself and the environment. Collisions terminate the episode which also enforces the agent to learn an efficient policy without needing to often backtrack.

In our setup, the robot action is composed of rotation τ and velocity v. We consider a fixed velocity which implies that a fix distance is traversed in each step. We discretize the rotation range into $K-1$ bins and the action at each step is the rotation degree corresponding to the chosen bin. We also add an stop action making the total number of actions equal to K. Note that although our action space is discrete, we move the agent in a continuous space rather than a predefined discrete grid. Following the success of [162], we devise a randomized simulator where light and textures are randomized during rendering both at training and test time.

4.4 Domain Invariant Visual Servoing

The visual servoing task of collision-free goal-reaching involves multiple underlying challenges: (1) Learning to visually localize goal objects. (2) Learning to predict collision map from RGB images. (3) Learning the optimal policy π to reach the objects. To integrate rich visual semantics while learning the control policy we opt to disentangle perception from control. We first learn a semantically rich model $\Phi_{\hat{\theta}}$ to represent visual sensory input as observation o. Freezing the $\hat{\theta}$ parameters, we then learn the control policy π_θ.

4.4.1 Network Architecture

Our representation learning module consists of two fully convolutional networks both based on VGG16 architecture [177] which we call them as *Collision Net* and *Semantic Net* and shown in Figure 4.1.

For the Collision Net, we follow [162] to pre-train a free-space prediction network. Collision

Net takes in an RGB image I and the output logit of its last convolutional layer provides an $N \times N$ collision map $\phi_{\hat{\theta}_c}(I)$ which predicts if an obstacle exists in the distance of 1 meter to the agent in its 2D ego-centric view. In Figure 4.1, $\phi_{\hat{\theta}_c}(I)$ is shown as a purple $N \times N$ map.

Our Semantic Net is built and trained based on [91]. We pre-train our Semantic Net on the MS-COCO object categories [117] with a weakly supervised object localization setup similar to [91]. We use the penultimate layer of the fully convolutional neural network of [91] to encode visual semantics in the spatial image space. Semantic Net describes each RGB image input via an $N \times N \times 2048$ map which we denote it as $S(I)$ and is shown by yellow box in Figure 4.1.

For each goal reaching task, our network takes in a goal image I^g that is fixed during the entire episode. At each timestep t, the network takes in the current image I_t, the previous image I_{t-1} as well as a goal image I^g. We use our Semantic Net to compute semantic feature representations $S(I^t)$, $S(I^{t-1})$ and $S(I^g)$, respectively. To represent the semantic correlation between the goal image and each of the input images in the 2D ego-centric view of the agent, we convolve the semantic visual representation of each input image with that of the goal image $\phi_{\hat{\theta}_s}(I, I_g) = S(I) \odot S(I^g)$. Also, for each pair of consecutive input images I_t and I_{t-1}, we compute the optical flow map $\psi(I_t, I_{t-1})$. We then stack the resulted pairs of collision maps $\phi_{\hat{\theta}_c}$, semantic correlation maps $\phi_{\hat{\theta}_s}$, and optical flow map ψ to generate the visual representation $\Phi_{\hat{\theta}}$ at each timestep t. For brevity we omit Is from the equation.

$$\Phi_{\hat{\theta}_t} = [\phi_{\hat{\theta}_c,t}, \phi_{\hat{\theta}_c,t-1}, \phi_{\hat{\theta}_s,t}, \phi_{\hat{\theta}_s,t-1}, \psi_{t,t-1}] \tag{4.1}$$

At each timestep t, we feed the observation $o_t = \Phi_{\hat{\theta}_t}$ into our policy learning module which consists of a convolution layer of size $2 \times 2 \times 64$, and a ConvLSTM [205] unit to incorporate the history. The policy is aimed to learn Q-values corresponding to the k discrete actions.

Implementation Details: We implement our network in TensorFlow [1] with input image size of 321×321. We pretrain Collision Net and Semantic Net with exact same parameters as in [162, 91]. In our policy module we use a single layer convLSTM with 256 units. We use Adam optimizer [102] with a learning rate of 10^{-3} and an exponential decay schedule with decay factor

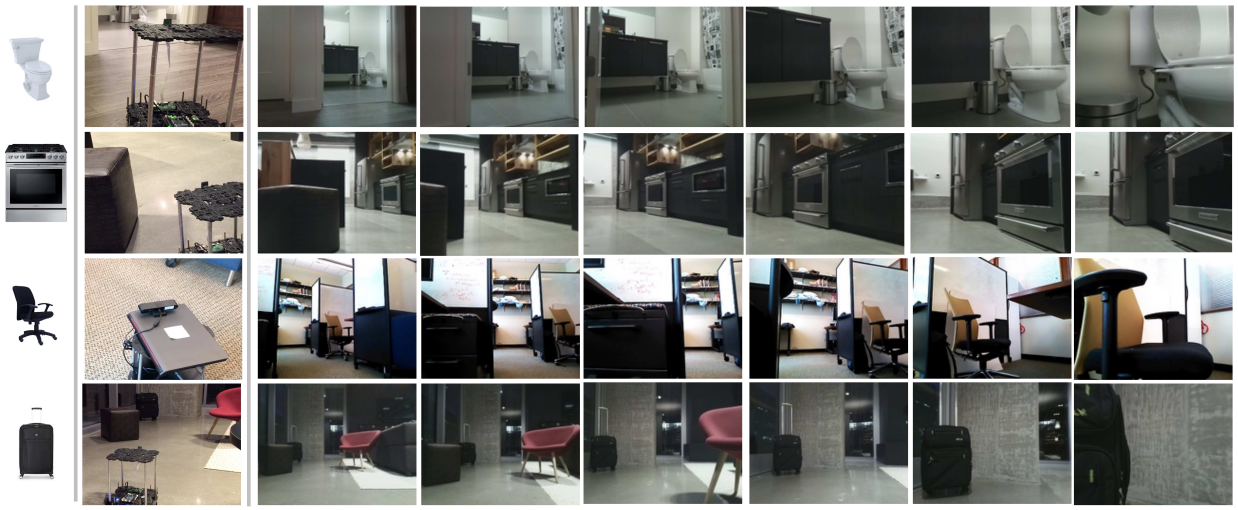

Figure 4.5: Real world experiment for reaching semantic goals of toilet, teddy bear, chair and suitcase. The sequences show the robot view captured by head mounted monocular camera. DIViS can generalize to reach semantic goal objects in the real world.

of 0.97 at each 4000 steps during 200000 iterations with batches of size 5.

4.4.2 Direct Policy Transfer to the Real World

Our Semantic Net incorporates robust visual features trained on rich semantic object categories and is capable of producing correlation map between the visual goal and the robot observation. Our simple mechanism for computing the visual correlation between the goal and the observation is the crux of our network for modeling domain invariant stimuli $\phi_{\hat{\theta}_s}$. This stimuli is directly fed to our policy network so that the agent can learn generalizable policies that can be directly transferred to the real world.

At the test time, we are able to use the same network for querying our visual servoing policy with semantic object categories. To do that, we mask the last layer of the Semantic Net with the category id l to produce $S_l(I)$ and use it in lieu of the correlation map $\widehat{\phi_{\hat{\theta}_s}}(I, l) = S_l(I)$ which will be fed into our policy module. Given the fact that our Semantic Net is trained with real images and is robust to noisy samples [91], our approach for computing the semantic correlation map as an input to the policy network provides domain invariant representations which we will empirically show to work well in real world scenarios. Our Collision Net is also domain invariant as it is

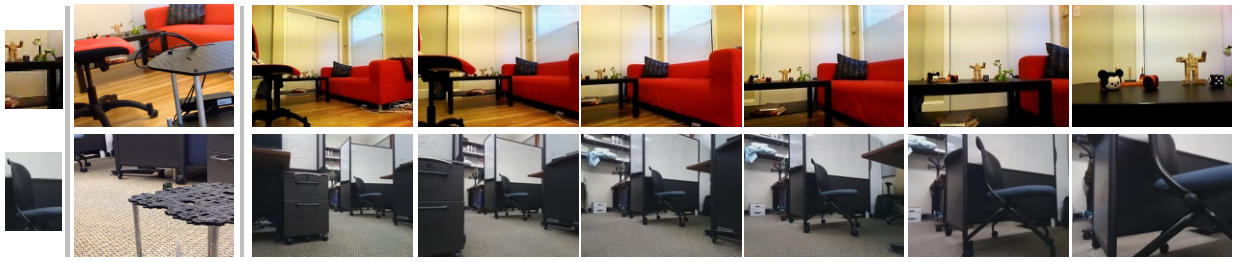

Figure 4.6: Real world experiments for reaching a goal in the robots view identified with a goal image. The first and second columns show the goal image and the third person view of the robot respectively. The image sequence show the input RGB images. Our goal reaching policy can generalize to diverse goals and can successfully avoid collisions in real world situations.

pre-trained via domain randomization technique [162].

4.4.3 Object Reaching via Deep RL

We consider a goal conditioned agent interacting with an environment in discrete timesteps. Starting from a random policy the agent is trained to choose actions towards getting closer to a goal. The goal is defined by cropping out the image patch around the goal object as seen at the initial state and is denote by I^g. At each timestep t, the agent receives an observation o_t, takes an action a_t from a set of k discrete actions $\{a^1, a^2, ..., a^k\}$ and obtains a scalar reward R_t. By following its policy π_θ, the agent produces a sequence of state-action pairs $\tau = \{s_t, a_t\}_{t=0}^{T-1}$ after T steps. The goal of the agent is to maximize the expected sum of discounted future rewards with a discounting factor $\gamma \in [0, 1]$:

$$Q(s_t, a) = R(s_t, a) + E_{\tau \sim \pi_\theta}[\sum_{t'=t+1}^{T} \gamma^{t'-t} R(s_{t'}, a_{t'})] \tag{4.2}$$

Dense Multi-Step Monte Carlo Rollouts We perform multi-step Monte Carlo policy evaluation [183] for all possible K actions at each state visited during an episode to generate dense rollouts. Starting the agent from an initial state we generate dense rollouts with maximum length of T. For each state along the trajectory τ, the dense rollouts are generated by performing $K - 1$ additional rollouts corresponding to the actions $\{a^i\}_{i=1, a^i \neq a_t}^{K}$ which are not selected according to the agent's current policy $a_t \sim \pi_\theta$ Figure 4.1, demonstrates our dense Monte Carlo rollouts along a trajectory. We evaluate the return of each action a according to Equation 4.2. Therefore, for each

state along the trajectories, we compute $\mathbb{Q}_{s_t} = \{Q(s_t, a^i)\}_{i=1}^{K}$ that densely encapsulates Q values that quantify the expected sum of future rewards for each of the possible actions a^i. This policy evaluation provides us with a dataset of trajectories of the form:

$$(s_0, \mathbb{Q}_{s_0}, a_0, ..., s_{T-1}, \mathbb{Q}_{s_{T-1}}, a_{T-1}) \tag{4.3}$$

If at any point during the episode the agent collides the corresponding rollout branch will be terminated. During training, we use batch RL [110], where we generate dense rollouts with Monte Carlo return estimates as explained above. For each of the training tasks, we collect dense Monte Carlo rollouts multiple times each with a randomized rendering setup. During training, we collect samples from all our environments and learn a single policy over all different reaching tasks simultaneously. This enforces the agent to learn the common shared aspect between various tasks (i.e. to reach different goals) and acquire generalization capability to unseen test scenarios.

Reactive Policy: The *reactive* agent starts from a random policy and does not save history from its past observations. The state at each timestep is described by the observation $s_t = o_t$. The visual observation o_t at timestep t is represented by $[\phi_{\hat{\theta}_c,t}, \phi_{\hat{\theta}_s,t}]$ and $(o_t, \mathbb{Q}_{o_t})$ pairs are used to update the policy.

Recurrent Policy: Starting from a random policy, the agent learns a recurrent policy using a sequence of observations. The recurrent policy uses the entire history of the observation, action pairs to describe the state $s_t = (o_1, a_1, ..., a_{t-1}, o_t)$. Each observation along the trajectory is represented by $\Phi_{\hat{\theta}_t}$ according to Equation 4.1. Given the sequential nature of the problem, we use dense trajectories described in Equation 4.3 and we model the policy π using a ConvLSTM unit. Intuitively, the history of past observations and actions will be captured in the hidden state of ConvLSTM.

Reward Function: For collision-free goal reaching, the agent should traverse a trajectory that decreases its distance to the goal object while avoiding obstacles. This implies two objectives to be reflected in the reward function: (1) We define the collision reward function as $R_c = \min(1, \frac{d_o - r}{\tau_d - r})$ to penalize the agent for colliding with various objects in the environment. Here, r is the vehicle radius, d_o denotes distance to the closest obstacle, and τ_d is a distance threshold. (2) The agent is rewarded whenever it makes progress towards the goal. Considering d_t as the distance of the agent

Table 4.1: DIViS success rate using two real robot platforms.

Robot	Goal Image		Semantic Object	
	success rate	#trials	success rate	#trials
WafflePi	82.35%	17	85.18%	27
TurtleBot2	81.81%	22	73.07%	26
Total	82.35%	39	79.24%	53

to the goal and d_{init} as the initial distance of the agent to the goal, our progress reward function is defined as $R_g = \max(0, 1 - \frac{\min(d_t, d_{init})}{d_{init}})$. The total reward is $R = R_c + R_g$.

4.5 Experimental Results

We evaluate the performance of DIViS for transferring semantic vision-based policy to the real world by conducting real-world experiments with two different real mobile robots. We also, study various design decisions via detailed quantitative simulated experiments and compare against baselines, alternative approaches, and different network architectures.

4.5.1 Real-World Evaluations

We use two different mobile robot platforms, TurtleBot2 and Waffle Pi (TurtleBot3), equipped with different monocular cameras (Astra and Raspberry Pi camera) to capture RGB sensory data used as input to our network.We compare various settings of training DIViS entirely in our domain randomized simulator and without any further fine-tuning or adaptation. Our goal is to answer the following key questions: (1) How well DIViS generalizes to real world settings while no real robot navigation data is used at training time? (2) How well does our approach transfer real world object-level semantics into the policy that is entirely trained in simulation? (3) How effective is our proposed recurrent policy compared to a reactive policy that is trained with similar pre-trained visual features? What is our performance compared to a conventional approach that visually matches the goal with current camera view? We study answer to these questions in the context of quantitative and qualitative real-world experiments.

Table 4.2: Comparing DIViS to baselines in reaching diverse goals in real world

	Goal Image		Semantic object		Total
	success rate	#trials	success rate	#trials	success rate
DIViS(ours)	**83.78%**	37	**75.6%**	41	**79.48%**
DIViS-reactive(ours)	54.04%		43.9%		48.71%
DIViS(ours)	**82.35%**	17	**85.18%**	27	**84.1%**
Visual Goal Matching	35.29%		18.51%		25.0%
DIViS(ours)	**86.66%**	15	**80.0%**	15	**83.33%**
DIViS-reactive(ours)	53.0%		53.53%		53.53%
Visual Goal Matching	40.0%		20.0%		30.0%

Quantitative Real World Experiments

Our quantitative real world evaluation consists of two different setups for collision-free goal-reaching (a) Goal Image: reaching a visual goal as specified by a user selecting an image patch from the initial robot view. (b) Semantic object: Reaching a semantic object category such as *chair*, *teddy bear*, etc. that is inside the initial robot view.

Generalizing to real world: Table 4.1, shows that our policy is robust to visually diverse inputs. Our Semantic Net and Collision Net and policy can directly transfer to real world for reaching image goals while avoiding obstacles obtaining an average success rate of 82.35% in goal image and 79.24% in semantic object scenarios using two different real robot platforms.

Generalizing to semantic objects: Our experiments outlined in Table 4.1 and Table 4.2 show that our policy can successfully generalize to reach semantic object categories with an average success rate of 79.24% (over 53 trials) although it has not been directly trained for this task. Since our Semantic Net (explained in section 4.4) is capable of localizing various object categories, it can provide the visual semantic correlation map for our policy network resulting in high generalization capability.

Ablation and comparison with baseline: We compare the performance of DIViS to its reactive version explained in Section 4.4 as well as a Visual goal matching policy that mimics a conventional visual servoing baseline as explained in Section 4.5.2. To compare each method against DIViS, we run same scenarios with same goal and same initial robot pose. Each section in Table 4.2,

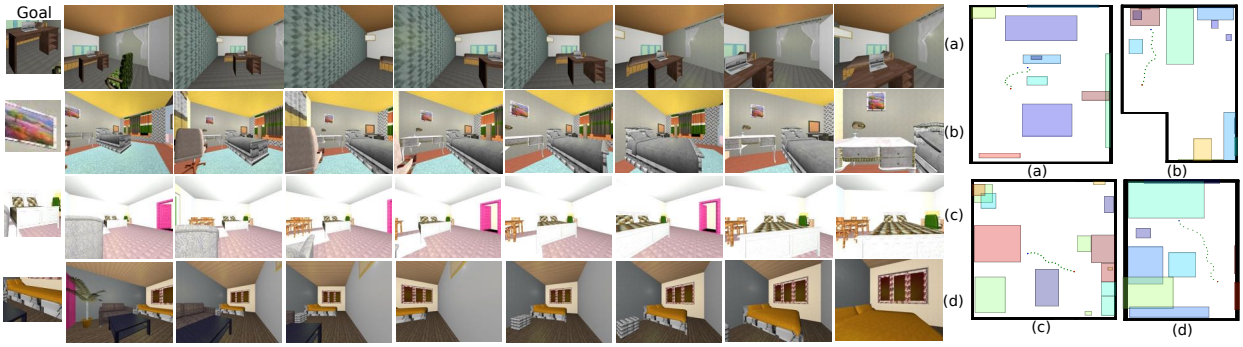

Figure 4.7: Qualitative results on several complex test scenarios. In each scenario the trajectory taken by the agent is overlaid on the map (right) and several frames along the path are shown (left). To avoid collisions, the agent needs to take turns that often takes the goal object out of its view and traverse walkable areas in order to achieve the visually indicated goal in diverse scenarios.

compares the methods over similar scenarios. A successful policy should be capable of making turns to avoid obstacles while keeping track of the target object that can go out of the monocular view of the robot in sub-trajectories. Table 4.2 outlines that DIViS has the highest success rate in all setups. Whilethe reactive policy can avoid obstacles it fails reaching the goal when it looses track of the objects at turns. Visual goal matching collides with obstacles more frequently as it greedily moves towards the object without considering the path clearance. Having saved the memory of past observations via a recurrent policy, DIViS is able to keep track of the goal object when it gets out of the view and makes better decisions to avoid obstacles.

Qualitative Real World Experiments

Figure 4.4 visualizes the performance of DIViS in a real wold “semantic goal” scenario where the goal is to reach the “teddy bear”. This example demonstrates the importance of incorporating both object semantics and free space reasoning in choosing best actions to find a collision-free path in order to reach the goal object. First row in Figure 4.4 shows the RGB images observed by the robot and the second row shows the object localization score map for the “teddy bear” as obtained by our Semantic Net. Red arrows in the third row of Figure 4.4 show the action direction chosen by visual goal matching policy which only incorporates semantic object understanding. Green arrows show the action directions chosen by DIViS. While visual goal matching guides the robot to get close to

the goal object, it does not have any mechanism to avoid obstacles and thus fails by colliding into other room furniture. On the other hand, DIViS maintains a collision-free path by choosing actions that both involve object semantics and free space reasoning. During traversal, DIViS may decide to take turns in order to avoid obstacles. This can result in loosing the sight of object for several steps. Being capable of maintaining a short memory, DIViS can turn back to the goal object after avoiding the obstacle.

More qualitative examples of DIViS for *goal image* and *semantic object* such as "toilet", "teddy bear", "chair" and "suitcase" are provided in Figure 4.6 and Figure 4.5, respectively. As demonstrated, DIViS can generalize to various real-world scenarios including diverse set of image goals, diverse object categories and various indoor and outdoor environments. For more qualitative examples please see supplementary videos at https://fsadeghi.github.io/DIViS

4.5.2 Simulation Evaluation

To generate simulation test scenarios, we sample the free spaces in the environments uniformly at random to select the initial location and camera orientation of the agent. Therefore, the distance to the goal object and the initial view points change from one scenario to another. To further diversify the test scenarios and test the generalization capability, we do simulation randomization (also known as domain randomization) [162] in each test scenario using textures that were unseen during the training time. During the course of each trial, if distance of the agent to any of the scene objects other than the goal object becomes less than the agents radius (i.e. $\sim 16cm$) a collision event is registered and the trial is terminated.

Quantitative Simulation Experiments

For the evaluation criteria, we report *success rate* which is the percentage of times that the agent successfully reaches the visually indicated object. If distance of the agent with the goal object is less than $30cm$, it is registered as success. We report the average success rate over a total of 700 different scenarios involving 65 distinct goal objects collected from train(380 scenarios) and novel test (320 scenario) environments . We compare DIViS (full model with recurrent policy and use of

Table 4.3: Success rate in simulation.

Method	Seen Env.	Unseen Env.
DIViS-Recurrent w/flow (ours)	**87.6**	**81.6**
DIViS-Recurrent (ours)	82.1	75.3
DIViS-Reactive (ours)	76.3	69.7
Visual Goal Matching	56.3	54.4
Visual Goal Matching w/ collis.	48.9	47.8
Random policy	23.4	22.2

optical flow) against several alternative approaches explained below. Quantitative comparisons are summarized in Table 4.3.

Random Policy: At each step, the agent selects one of the k actions uniformly at random.

Visual Goal Matching: This baseline models a greedy policy that follows an oracle rule of following the path with highest visual similarity to the goal and mimics conventional image-based visual servoing techniques to find the best matching visual features with the goal. Note that this policy uses a high-level prior knowledge about the underlying task while our agent s trained via RL does not and instead should learn a policy from scratch without any priors. Visual Goal Matching selects one of the k actions based on the spatial location of the maximum visual matching score of the visual goal and the current observation. To compute visual correlation map, we use same Semantic Net pre-trained features used in our network for fairness.

Visual Goal Matching with Collision Avoidance:. This baseline combines prior sim2real collision avoidance method of [162] with conventional visual servoing for following the path with highest visual match to the goal and lowest chance of collision. We incorporate our predicted collision maps to extend Visual Goal Matching policy for better collision avoidance. Using our Collision Net and Semantic Net, we compute the spatial free space map and semantic correlation map for each observation. We sum up these two maps and obtain a single spatial score map that highlights the action directions with highest visual correlation and lowest chance of collision. The agent selects one of the k actions based on the spatial location with highest total score. To be fair, we use the exact same pre-trained features of Collision Net and Semantic Net in our network for

this policy.

DIViS-Reactive policy: The agent selects the best action based on the maximum Q-value produced by the reactive policy explained in Section 4.4.

DIViS-Recurrent: The agent selects the best action based on the maximum Q-value produced by our recurrent policy without incorporating optical flow ψ explained in Section 4.4.

DIViS-Recurrent with flow: Our full model that select the best action based on the maximum Q-value produced by our network that also uses the optical flow between each two consecutive observations as explained in Section 4.4 and Equation 4.1.

Table 4.3, compares the success rates between different approaches. The highest performance is obtained by our recurrent policy and the best results are obtained when optical flowψ is also incorporated. Interestingly, naive combination of collision prediction probabilities with visual goal matching results in lower performance than only using visual goal matching. This is because the collision avoidance tends to select actions that navigate the agent to spaces with smallest probability of collision. However, in order to reach goals the agent should be able to take narrow paths in confined spaces which might not have the lowest collision probability. Given the results obtained in this experiment, we used our recurrent policy with optical flow during our real-world experiments in Section 4.5.1.

Qualitative Simulation Experiments

Figure 4.7 demonstrates several qualitatively evaluation of our best policy i.e. DIViS (with recurrent policy and optical flow) in a number of complex test scenarios. In each scenario, the trajectory taken by the agent is overlaid on the top view of map. The initial and final position of the agent are shown by a red and blue dots, respectively. During these scenarios, the agent needs to take actions which may increase its distance to the goal but will result in avoiding obstacles. However, the agent recovers by taking turns around the obstacles and successfully reaches the goal object.

4.6 Discussion

We described a novel sim-to-real learning approach for visual servoing which is invariant to the domain shift. DIViS, our proposed domain invariant visual servoing approach, is entirely trained in simulation for reaching visually indicated goals. Despite this fact, DIViS can successfully be deployed on real robot platforms and can flexibly be tasked to reach semantic object categories at the test time in the real environments. Our approach proposes transferring visual semantics from real static image datasets and learning physics in simulation. We evaluated the performance of our approach via detailed quantitative and qualitative evaluations both in the real world and in simulation. Our experiments show that DIViS can indeed accomplish reaching various visual goals and semantic objects at drastically different and unstructured real environments. Our approach can lead into learning domain invariant policies for various vision-based control problems that involve semantic objects. Future directions include investigating how DIViS can be extended to work in dynamic environments with moving obstacles or goals which is an important challenge to be addressed in real-world robotics.

Chapter 5
VISUAL ANALOGY

"To bring anything into your life, imagine that it's already there. "

Richard Bach

In this chapter, we study the problem of answering visual analogy questions. These questions take the form of *image A is to image B as image C is to what*. Answering these questions entails discovering the mapping from image A to image B and then extending the mapping to image C and searching for the image D such that the relation from A to B holds for C to D. We pose this problem as learning an embedding that encourages pairs of analogous images with similar transformations to be close together using convolutional neural networks with a quadruple Siamese architecture. We introduce a dataset of visual analogy questions in natural images, and show first results of its kind on solving analogy questions on natural images.

5.1 Overview

Analogy is the task of mapping information from a source to a target. Analogical thinking is a crucial component in problem solving and has been regarded as a core component of cognition [68]. Analogies have been extensively explored in cognitive sciences and explained by several theories and models: shared structure [68], shared abstraction [173], identity of relation, hidden deduction [100], etc. The common two components among most theories are the discovery of a form of relation or mapping in the source and extension of the relation to the target. Such a process is very similar to the tasks in analogy questions in standardized tests such as the Scholastic Aptitude Test (SAT): *A is to B as C is to what?*

In this chapter, we introduce VISALOGY to address the problem of solving visual analogy

questions. Three images I_a, I_b, and I_c are provided as input and a fourth image I_d must be selected such that I_a *is to* I_b *as* I_c *is to* I_d. This involves discovering an extendable mapping from I_a to I_b and then applying it to I_c to find I_d. Estimating such a mapping for natural images using current feature spaces would require careful alignment, complex reasoning, and potentially expensive training data. Instead, we learn an embedding space where reasoning about analogies can be performed by simple vector transformations. This is in fact aligned with the traditional logical understanding of analogy as an arrow or homomorphism from source to the target.

Our goal is to learn a representation that given a set of training analogies can generalize to unseen analogies across various categories and attributes. Figure 5.1 shows an example visual analogy question. Answering this question entails discovering the mapping from the brown bear to the white bear (in this case a color change), applying the same mapping to the brown dog, and then searching among a set of images (the middle row in Figure 5.1) to find an example that respects the discovered mapping from the brown dog best. Such a mapping should ideally prefer white dogs. The bottom row shows a ranking imposed by VISALOGY.

We propose learning an embedding that encourages pairs of analogous images with similar mappings to be close together. Specifically, we learn a Convolutional Neural Network (CNN) with Siamese quadruple architecture (Figure 5.2) to obtain an embedding space where analogical reasoning can be done with simple vector transformations. Doing so involves fine tuning the last layers of our network so that the difference in the unit normalized activations between analogue images is similar for image pairs with similar mapping and dissimilar for those that are not. We also evaluate VISALOGY on generalization to unseen analogies. To show the benefits of the proposed method, we compare VISALOGY against competitive baselines that use standard CNNs trained for classification. Our experiments are conducted on datasets containing natural images as well as synthesized images and the results include quantitative evaluations of VISALOGY across different sizes of distractor sets. The performance in solving analogy questions is directly affected by the size of the set from which the candidate images are selected.

In this chapter we study the problem of visual analogies for natural images and show the first results of its kind on solving visual analogy questions for natural images. Our proposed method

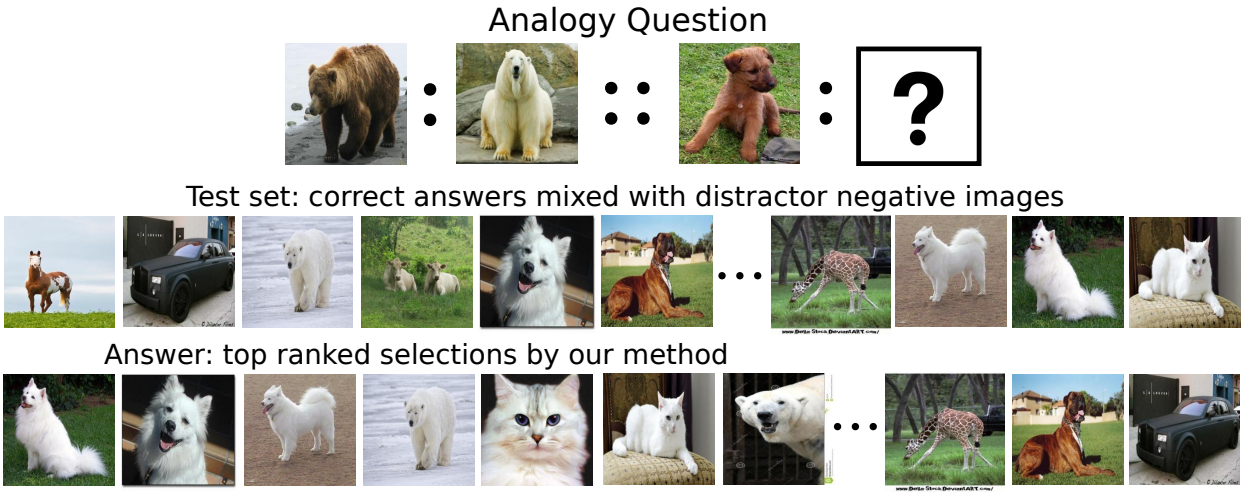

Figure 5.1: Visual analogy question asks for a missing image I_d given three images I_a, I_b, I_c in the analogy quadruple. Solving a visual analogy question entails discovering the mapping from I_a to I_b and applying it to I_c and search among a set of images (the middle row) to find the best image for which the mapping holds. The bottom row shows an ordering of the images imposed by VISALOGY based on how likely they can be the answer to the analogy question.

learns an embedding where similarities are transferable across pairs of analogous images using a Siamese network architecture. We introduce Visual Analogy Question Answering (VAQA), a dataset of natural images that can be used to generate analogies across different objects attributes and actions of animals. We also compile a large set of analogy questions using the 3D chair dataset [11] containing analogies across viewpoint and style. Our experimental evaluations show promising results on solving visual analogy questions. We explore different kinds of analogies with various numbers of distracters, and show generalization to unseen analogies.

5.2 Related Work

The problem of solving analogy questions has been explored in NLP using word-pair connectives [192], supervised learning [193, 16, 99], distributional similarities [194], word vector representations and linguistic regularities [116], and learning by reading [15].

Solving analogy questions for diagrams and sketches has been extensively explored in AI [32]. These papers either assume simple forms of drawings [63], require an abstract representation of diagrams [62], or spatial reasoning [33]. In [80] an analogy-based framework is proposed to learn 'image filters' between a pair of images to creat an 'analogous' filtered result on a third image.

Related to analogies is learning how to separate category and style properties in images, which has been studied using bilinear models [186]. In this chapter, we study the problem of visual analogies for natural images possessing different semantic properties where obtaining abstract representations is extremely challenging.

Our work is also related to metric learning using deep neural networks. In [38] a convolutional network is learned in a Siamese architecture for the task of face verification. Attributes have been shown to be effective representations for semantical image understanding [59]. In [139], the relative attributes are introduced to learn a ranking function per attribute. While these methods provide an efficient feature representation to group similar objects and map similar images nearby each other in an embedding space, they do not offer a semantic space that can capture object-to-object mapping and cannot be directly used for object-to-object analogical inference. In [90] the relationships between multiple pairs of classes are modeled via analogies, which is shown to improve recognition as well as GRE textual analogy tests. In our work we learn analogies without explicity considering categories and no textual data is provided in our analogy questions.

Learning representations using both textual and visual information has also been explored using deep architectures. These representations show promising results for learning a mapping between visual data[103] the same way that it was shown for text [128]. We differ from these methods as our objective is to directly optimized for analogy questions and our method does not use textual information.

Different forms of visual reasoning has been explored in the Question-Answering domain. Recently, the visual question answering problem has been studied in several papers [67, 121, 161, 10, 209, 122]. In [121] a method is introduced for answering several types of textual questions grounded with images while [10] proposes the task of open-ended visual question answering. In another recent approach [161], knowledge extracted from web visual data is used to answer open-domain questions. While these works all use visual reasoning to answer questions, none have considered solving *analogy* questions.

5.3 Deep Visual Analogy

We pose answering a visual analogy question $I_1 : I_2 :: I_3 :?$ as the problem of discovering the mapping from image I_1 to image I_2 and searching for an image I_4 that has the same relation to image I_3 as I_1 to I_2. Specifically, we find a function T (parametrized by θ) that maps each pair of images (I_1, I_2) to a vector $x_{12} = T(X_1, X_2; \theta)$. The goal is to solve for parameters θ such that $x_{12} \approx x_{34}$ for positive image analogies $I_1 : I_2 :: I_3 : I_4$. As we describe below, T is computed using the differences in ConvNet output features between images.

5.3.1 Quadruple Siamese Network

A positive training example for our network is an analogical quadruple of images $[I_1 : I_2 :: I_3 : I_4]$ where the transformation from I_3 to I_4 is the same as that of I_1 to I_2. To be able to solve the visual analogy problem, our learned parameters θ should map these two transformations to a similar location. To formalize this, we use a contrastive loss function L to measure how well T is capable of placing similar transformations nearby in the embedding space and pushing dissimilar transformations apart. Given a d-dimensional feature vector x for each pair of input images, the contrastive loss is defined as:

$$\mathcal{L}^m(x_{12}, x_{34}) = y||x_{12} - x_{34}|| + (1 - y)\max(m - ||x_{12} - x_{34}||, 0) \tag{5.1}$$

where x_{12} and x_{34} refer to the embedding feature vector for (I_1, I_2) and (I_3, I_4) respectively. Label y is 1 if the input quadruple $[I_1 : I_2 :: I_3 : I_4]$ is a correct analogy or 0 otherwise. Also, $m > 0$ is the margin parameter that pushes x_{12} and x_{34} close to each other in the embedding space if $y = 1$ and forces the distance between x_{12} and x_{34} in wrong analogy pairs ($y = 0$) be bigger than $m > 0$, in the embedding space. We train our network with both correct and wrong analogy quadruples and the error is back propagated through stochastic gradient descent to adjust the network weights θ. The overview of our network architecture is shown in Figure 5.2.

To compute the embedding vectors x we use the quadruple Siamese architecture shown in Figure 5.2. Using this architecture, each image in the analogy quadruple is fed through a ConvNet

(AlexNet [106]) with shared parameters θ. The label y shows whether the input quadruple is a correct analogy ($y = 1$) or a false analogy ($y = 0$) example. To capture the transformation between image pairs (I_1, I_2) and (I_3, I_4), the outputs of the last fully connected layer are subtracted. We normalize our embedding vectors to have unit L2 length, which results in the Euclidean distance being the same as the cosine distance. If X_i are the outputs of the last fully connected layer in the ConvNet for image I_i, $x_{ij} = T(X_i, X_j; \theta)$ is computed by:

$$T(X_i, X_j; \theta) = \frac{X_i - X_j}{||X_i - X_j||} \tag{5.2}$$

Using the loss function defined in Equation (5.1) may lead to the network overfitting. Positive analogy pairs in the training set can get pushed too close together in the embedding space during training. To overcome this problem, we consider a margin $m_P > 0$ for positive analogy quadruples. In this case, x_{12} and x_{34} in the positive analogy pairs will be pushed close to each other only if the distance between them is bigger than $m_P > 0$. It is clear that $0 \leq m_P \leq m_N$ should hold between the two margins.

$$\mathcal{L}^{m_P, m_N}(x_{12}, x_{34}) = y \max(||x_{12} - x_{34}|| - m_P, 0) + (1 - y) \max(m_N - ||x_{12} - x_{34}||, 0) \tag{5.3}$$

5.3.2 Building Analogy Questions

For creating a dataset of visual analogy questions we assume each training image has information (c, p) where $c \in \mathcal{C}$ denotes its category and $p \in \mathcal{P}$ denotes its property. Example properties include color, actions, and object orientation. A valid analogy quadruple should have the form:

$$[I_1^{(c_i, p_1)} : I_2^{(c_i, p_2)} :: I_3^{(c_o, p_1)} : I_4^{(c_o, p_2)}]$$

where the two input images I_1 and I_2 have the same category c_i, but their properties are different. That is, I_1 has the property p_1 while I_2 has the property p_2. Similarly, the output images I_3 and I_4

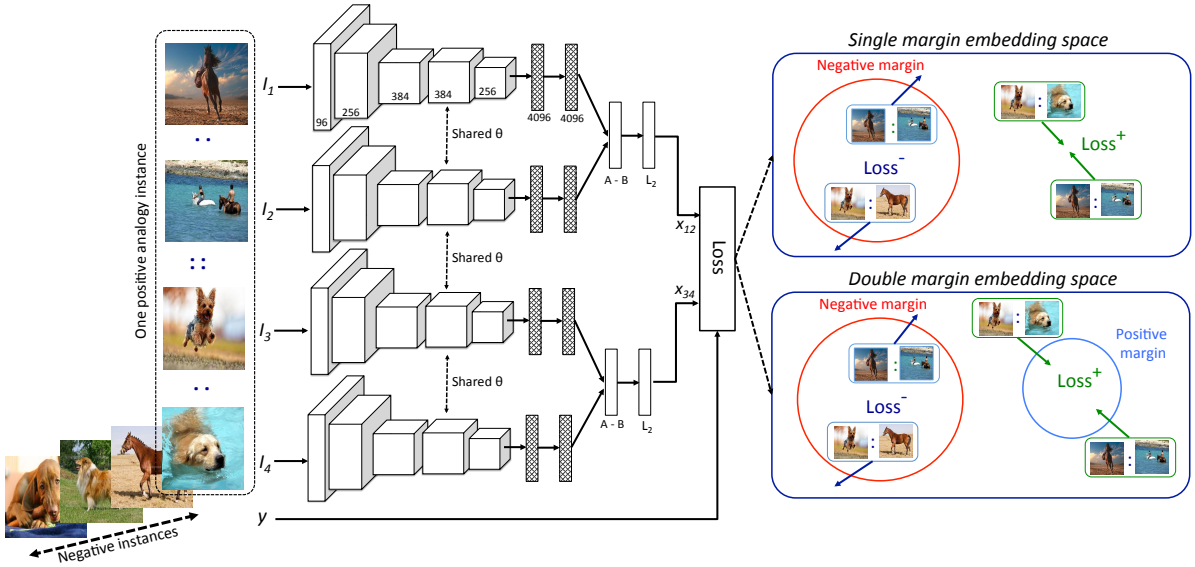

Figure 5.2: VISALOGY Network has quadruple Siamese architecture with shared θ parameters. The network is trained with correct analogy quadruples of images [I_1, I_2, I_3, I_4] along with wrong analogy quadruples as negative samples. The contrastive loss function pushes (I_1, I_2) and (I_3, I_4) of correct analogies close to each other in the embedding space while forcing the distance between (I_1, I_2) and (I_3, I_4) in negative samples to be more than margin m.

share the same category c_o where $c_i \neq c_o$. Also, I_3 has the property p_1 while I_4 has the property p_2 and $p_1 \neq p_2$.

Generating Positive Quadruples: Given a set of labeled images, we construct our set of analogy types. We select two distinct categories $c, c' \in \mathcal{C}$ and two distinct properties $p, p' \in \mathcal{P}$ which are shared between c and c'. Using these selections, we can build 4 different analogy types (either c or c' can be considered as c_i and c_o and similarly for p and p'). For each analogy type (e.g. $[(c_i, p_1) : (c_i, p_2) :: (c_o, p_1) : (c_o, p_2)]$), we can generate a set of positive analogy samples by combining corresponding images. This procedure provides a large number of positive analogy pairs.

Generating Negative Quadruples: Using only positive samples for training the network leads to degenerate models, since the loss can be made zero by simply mapping each input image to a constant vector. Therefore, we also generate quadraples that violate the analogy rules as negative samples during training. To generate negative quadruples, we take two approaches. In the first approach, we randomly select 4 images from the whole set of training images and each time check

that the generated quadruple is not a valid analogy. In the second approach, we first generate a positive analogy quadruple, then we randomly replace either of I_3 or I_4 with an improper image to break the analogy. Suppose we select I_3 for replacement. Then we can either randomly select an image with category c_o and property p^* where $p^* \neq p_1$ and $p^* \neq p_2$ or we can randomly select an image with property p_1 but with a category c^* where $c^* \neq c_o$. The second approach generates a set of hard negatives to help improve training. During the training, we randomly sample from the whole set of possible negatives.

5.4 Experiments

Testing Scenario and Evaluation Metric: To evaluate the performance of our method for solving visual analogy questions, we create a set of correct analogy quadruples $[I_1 : I_2 :: I_3 :?]$ using the (c, p) labels of images. Given a set $\mathcal{D}$ of images which contain both positive and distracter images, we would like to rank each image I_i in $\mathcal{D}$ based on how well it completes the analogy. We compute the corresponding feature embeddings x_1, x_2, x_3, for each of the input images as well as x_i for each image in $\mathcal{D}$ and we rank based on:

$$rank_i = \frac{T(I_1, I_2).T(I_3, I_i)}{||T(I_1, I_2)||.||T(I_3, I_i)||}, \;\; i \in 1, ..., n \tag{5.4}$$

where $T(.)$ is the embedding obtained from our network as explained in section 5.3. We consider the images with the same category c as of I_3 and the same property p as of I_2 to be a correct retrieval and thus a positive image and the rest of the images in $\mathcal{D}$ as negative images. We compute the recall at top-k to measure whether or not an image with an appropriate label has appeared in the top k retrieved images.

Baseline: It has been shown that the output of the 7th layer in AlexNet produces high quality state-of-the-art image descriptors [106]. In each of our experiments, we compare the performance of solving visual analogy problems using the image embedding obtained from our network with the image representation of AlexNet. In practice, we pass each test image through AlexNet and our network, and extract the output from the last fully connected layer using both networks. Note that for solving general analogy questions the set of properties and categories are not known at the

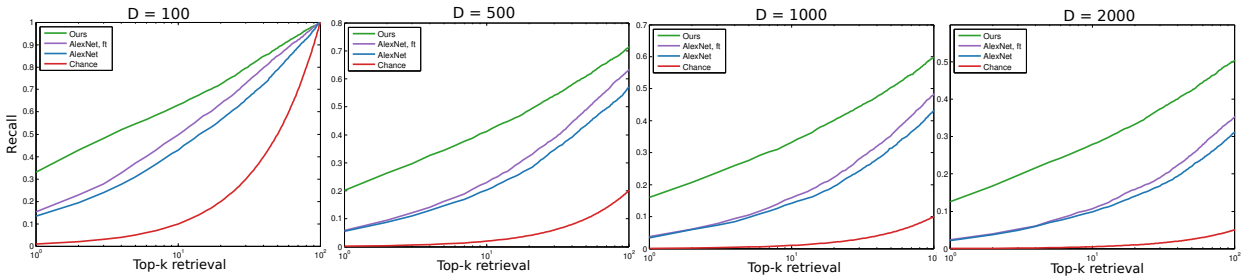

Figure 5.3: Quantitative evaluation (log scale) on 3D chairs dataset. Recall as a function of the number (k) of images returned (Recall at top-k). For each question the recall at top-k is either 0 or 1 and is averaged over 10,000 questions. The size of the distractor set D is varied $D = [100, 500, 1000, 2000]$. 'AlexNet': AlexNet, 'AlexNet ft': AlexNet fine-tuned on chairs dataset for categorizing view-points.

test time. Accordingly, our proposed network does not use any labels during training and is aimed to generalize the transformations without explictily using the label of categories and properties.

Dataset: To evaluate the capability of our trained network for solving analogy questions in the test scenarios explained above, we use a large dataset of 3D chairs [11] as well as a novel dataset of natural images (VAQA), that we collected for solving analogy questions on natural images.

5.4.1 Implementation Details

In all the experiments, we use stochastic gradient descent (SGD) to train our network. For initializing the weights of our network, we use the AlexNet pre-trained network for the task of large-scale object recognition (ILSVRC2012) provided by the BVLC Caffe website [98]. We fine-tune the last two fully connected layers (fc6, fc7) and the last convolutional layer (conv5) unless stated otherwise. We have also used the double margin loss function introduced in Equation 5.3 with $m_P = 0.2, m_N = 0.4$ which we empirically found to give the best results in a held-out validatation set. The effect of using a single margin vs. double margin loss function is also investigated in section 5.4.4.

5.4.2 Analogy Question Answering Using 3D Chairs

We use a large collection of 1,393 models of chairs with different styles introduced in [11]. To make the dataset, the CAD models are download from Google/Trimble 3D Warehouse and each chair

style is rendered on white background from different view points. For making analogy quadruples, we use 31 different view points of each chair style which results in 1,393*31 = 43,183 synthesized images. In this dataset, we treat different styles as different *categories* and different view points as different *properties* of the images according to the explanations given in section 5.3.2. We randomly select 1000 styles and 16 view points for training and keep the rest for testing. We use the rest of 393 classes of chairs with 15 view points (which are completely unseen during the training) to build unseen analogy questions that test the generalization capability of our network at test time. To construct an analogy question, we randomly select two different styles and two different view points. The first part of the analogy quadruple (I_1, I_2) contains two images with the same style and with two different view points. The images from the second half of the analogy quadruple (I_3, I_4), have another style and I_3 has the same viewpoint as I_1 and I_4 has the same view point as I_2. Together, I_1, I_2, I_3 and I_4 build an analogy question $(I_1 : I_2 :: I_3 :?)$ where I_4 is the correct answer. Using this approach, the total number of positive analogies that could be used during training is $\binom{1000}{2} \times \binom{16}{2} \times 4 = 999,240$.

To train our network, we uniformly sampled 700,000 quadruples (of positive and negative analogies) and initialized the weights with the AlexNet pre-trained network and fine-tuned its parameters. Figure 5.4 shows several samples of the analogy questions (left column) used at test time and the top-4 images retrieved by our method (middle column) compared with the baseline (right column). We see that our proposed approach can retrieve images with a style similar to that of the third image and with a view-point similar to the second image while the baseline approach is biased towards retrieving chairs with a style similar to that of the first and the second image. To quantitatively compare the performance of our method with the baseline, we randomly generated 10,000 analogy questions using the test images and report the average recall at top-k retrieval while varying the number of irrelevant images (D) in the distractor set. Note that, since there is only one image corresponding to each (style , view-point), there is only one positive answer image for each question. The performance of chance at the top-kth retrieval is $\frac{k}{n}$ where n is the size of D. The images of this dataset are synthesized and do not follow natural image statistics. Therefore, to be fair at comparing the results obtained from our network with that of the baseline (AlexNet), we

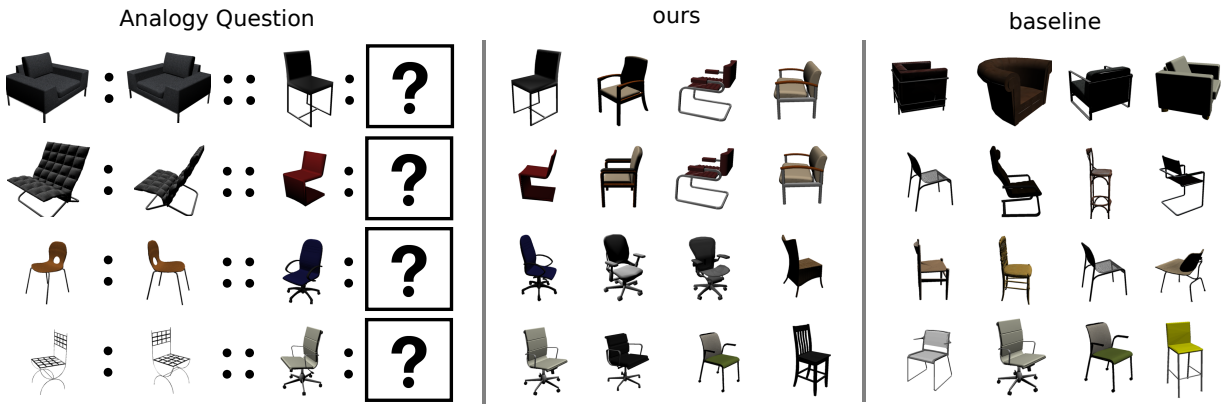

Figure 5.4: Left: Several examples of analogy questions from the 3D chairs dataset. In each question, the first and second chair have the same style while their view points change. The third image has the same view point as the first image but in a different style. The correct answer to each question is retrieved from a set with 100 distractors and should have the same style as the third image while its view point should be similar to the second image. Middle: Top-4 retrievals using the features obtained from our method . Right: Top-4 retrievals using AlexNet features. All retrievals are sorted from left to right

fine-tune all layers of the AlexNet via a soft-max loss for categorization of different view-points and using the set of images seen during training. We then use the features obtained from the last fully connected layer (fc7) of this network to solve analogy questions. As shown in Figure 5.3, fine-tuning all layers of AlexNet (the violet curve referred to as 'AlexNet,ft' in the diagram) helps improve the performance of the baseline. However, the recall of our network still outperforms it with a large margin.

5.4.3 Analogy Question Answering using VAQA Dataset

As explained in section 5.3.2, to construct a natural image analogy dataset we need to have images of numerous object categories with distinguishable properties. We also need to have these properties be shared amongst object categories so that we can make valid analogy quadruples using the (c, p) labels. In natural images, we consider the *property* of an object to be either the action that it is doing (for animate objects) or its attribute (for both animate and non-animate objects). Unfortunately, we found that current datasets have a sparse number of object properties per class, which restricts the number of possible analogy questions. For instance, many action datasets are human centric, and do not have analogous actions for animals. As a result, we collected our own

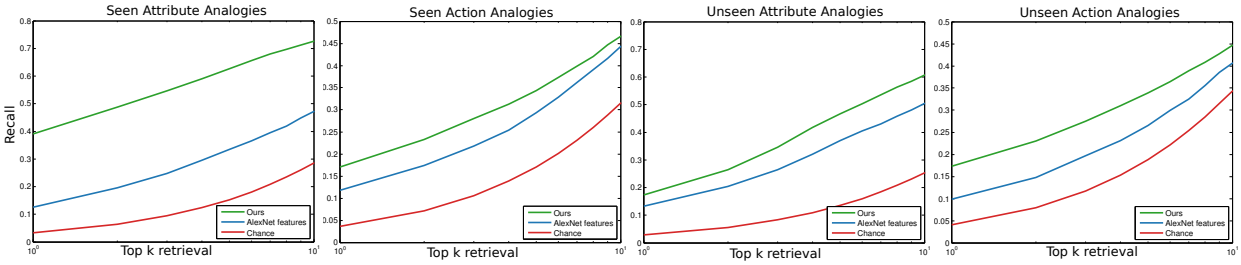

Figure 5.5: Quantitative evaluation (log scale) on the VAQA dataset using 'attribute' and 'action' analogy questions. Recall as a function of the number (k) of images returned (Recall at top-k). For each question the recall at top-k is averaged over 10,000 questions. The size of the distractor set is fixed at 250 in all experiments. Results shown for analogy types seen in training are shown in the left two plots, and for analogy types not seen in training in the two right plots.

dataset VAQA for solving visual analogy questions.

Data collection: We considered a list of 'attributes' and 'actions' along with a list of common objects and paired them to make a list of (c, p) labels for collecting images. Out of this list, we removed (c, p) combinations that are not common in the real world (e.g. (horse,blue) is not common in the real world though there might be synthesized images of 'blue horse' in the web). We used the remaining list of labels to query Google Image Search with phrases made from concatenation of word c and p and downloaded 100 images for each phrase. The images are manually verified to contain the concept of interest. However, we did not pose any restriction about the view-point of the objects. After the pruning step, there exists around 70 images per category with a total of 7,500 images. The VAQA dataset consists of images corresponding to 112 phrases which are made out of 14 different categories and 22 properties. Using the shared properties amongst categories we can build 756 types of analogies. In our experiments, we used over 700,000 analogy questions for training our network.

Attribute analogy: Following the procedure explained in Section 5.3.2 we build positive and negative quadruples to train our network. To be able to test the generalization of the learned embeddings for solving analogy question types that are not seen during training, we randomly select 18 attribute analogy types and remove samples of them from the training set of analogies. Using the remaining analogy types, we sampled a total of 700,000 quadruples (positive and negative) that are used to train the network.

Action analogy: Similarly, we trained our network to learn action analogies. For the generalization test, we remove 12 randomly selected analogy types and make the training quadruples using the remaining types. We sampled 700,000 quadruples (positive and negative) to train the network.

Evaluation on VAQA: Using the unseen images during the training, we make analogy quadruples to test the trained networks for the 'attribute' and 'action' analogies. For evaluating the specification and generalization of our trained network we generate analogy quadruples in two scenarios of 'seen' and 'unseen' analogies using the analogy types seen during training and the ones in the withheld sets respectively. In each of these scenarios, we generated 10,000 analogy questions and report the average recall at top-k. For each question $[I_1 : I_2 :: I_3 :?]$, images that have property p equal to that of I_2 and category c equal to I_3 are considered as correct answers. The result is around 4 positive images for each question and we fix the distracter set to have 250 negative images for each question. Given the small size of our distracter set, we report the average recall at top-10. The obtained results in different scenarios as summarized in Figure 5.5. In all the cases, our method outperforms the baseline.

Other than training separate networks for 'attribute' and 'action' analogies, we trained and tested our network with a combined set of analogy questions and obtained promising results with a gap of 5% compared to our baseline on the top-5 retrievals of the seen analogy questions. Note that our current dataset only has one property label per image (either for 'attribute' or 'action'). Thus, a negative analogy for one property may be positive for the other. A more thorough analysis would require multi-property data, which we leave for future work.

Qualitative Analysis: Figure 5.6, shows examples of attribute analogy questions that are used for evaluating our network along with the top five retrieved images obtained from our method and the baseline method. As explained above, during the data collection we only prune out images that do not contain the (c, p) of interest. Also, we do not pose any restriction for generating positive quadruples such as restricting the objects to have similar pose or having the same number of objects of interest in the quadruples. However, as can be seen in Figure 5.6 our network had been able to *implicitly* learn to generalize the *count* of objects. For example, in the first row of Figure 5.6, an image pair is ['dog swimming' : 'dog standing'] and the second part of the analogy has an image

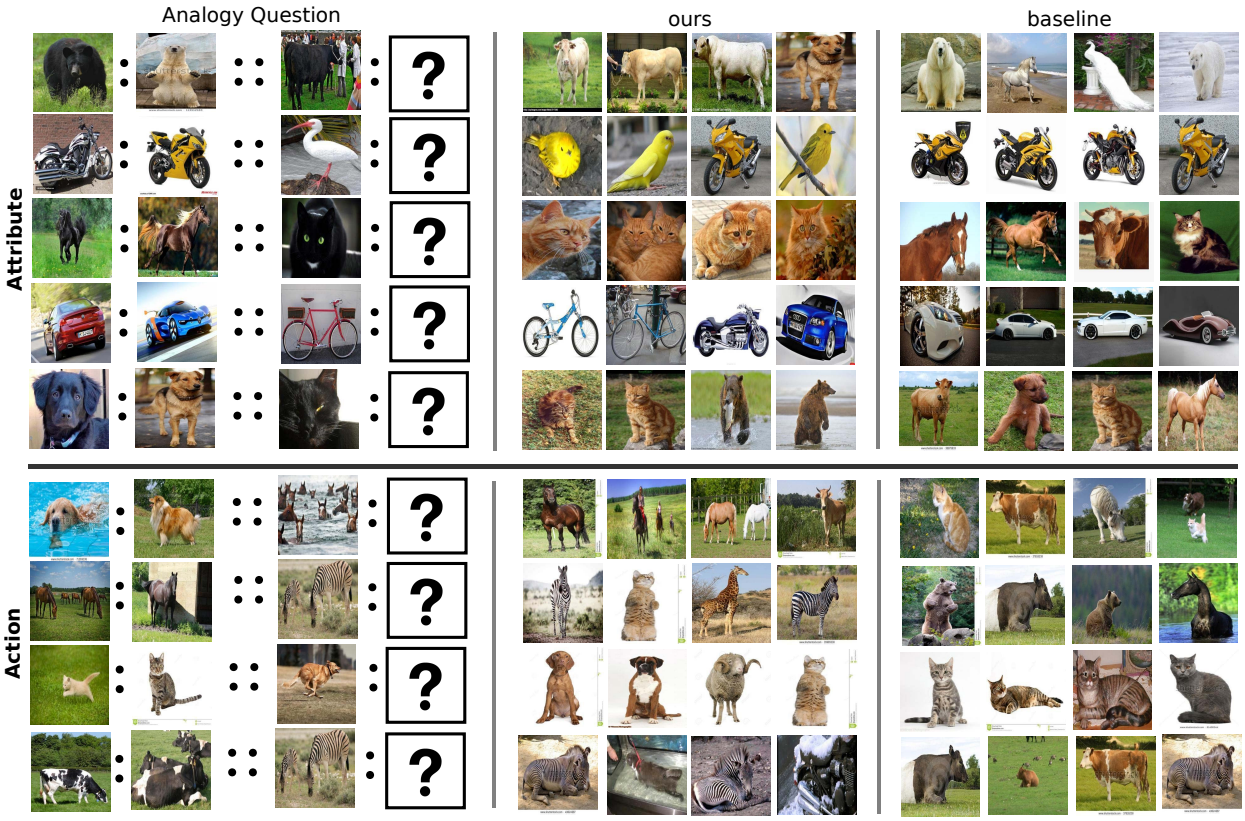

Figure 5.6: Left: Samples of test analogy questions from VAQA dataset. Middle: Top-4 retrievals using the features obtained from our method. Right: Top-4 retrievals using AlexNet features.

of 'multiple horses swimming'. Given this analogy question as input, our network has retrieved images with multiple 'standing horses' in the top five retrievals.

5.4.4 Ablation Study

In this section, we investigate the effect of training the network with double margins (m_P, m_N) for positive and negative analogy quadruples compared with only using one single margin for negative quadruples. We perform an ablation experiment where we compare the performance of the network at top-k retrieval while being trained using either of the loss functions explained in Section 5.4. Also, in two different scenarios, we either fine-tune only the top fully connected layers fc6 and fc7 (referred to as 'ft(fc6,fc7)' in Figure 5.7) or the top fully connected layers plus the last convolutional layer c5 (referred to as 'ft(fc6,fc7,c5)') in Figure 5.7). We use a fixed training

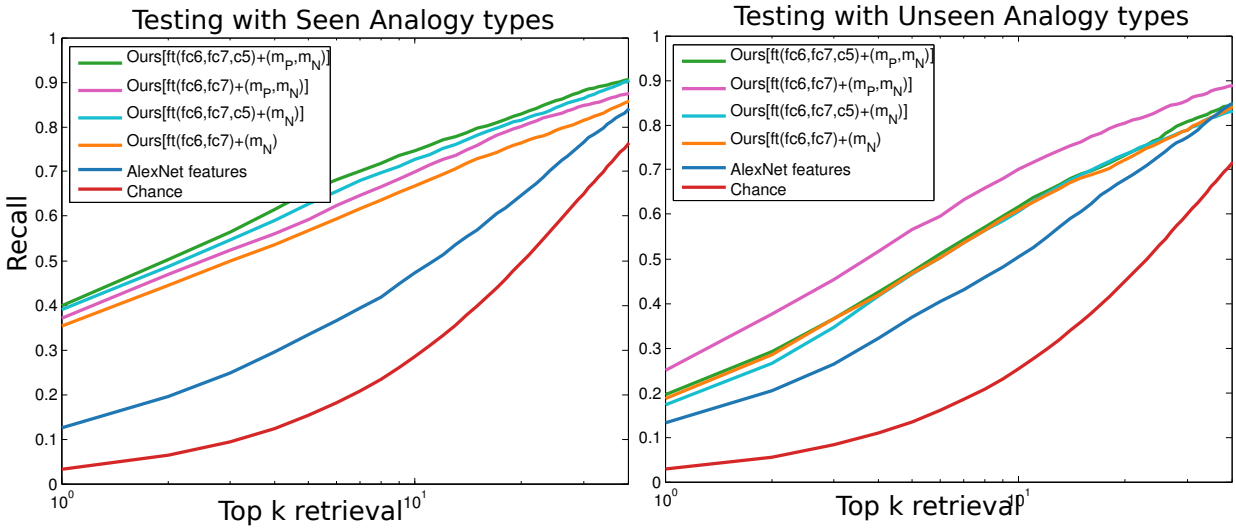

Figure 5.7: Quantitative comparison for the effect of using double margin vs. single margin for training the VISALOGY network.

sample set consisting of 700,000 quadruples generated from the VAQA dataset in this experiment. In each case, we test the trained network using samples coming from the set of analogy questions whose types are seen/unseen during the training. As can be seen from Figure 5.7, using double margins (m_P, m_N) in the loss function has resulted in better performance in both testing scenarios. While using double margins results in a small increase in the 'seen analogy types' testing scenario, it has considerably increased the recall when the network was tested with 'unseen analogy types'. This demonstrates that the use of double margins helps generalization.

Chapter 6

VIEWPOINT INVARIANT VISUAL SERVOING BY RECURRENT CONTROL

"There's a different point of view awaiting you...if you just look UP "

Mary Poppins

Humans are remarkably proficient at controlling their limbs and tools from a wide range of viewpoints. In robotics, this ability is referred to as visual servoing: moving a tool or end-point to a desired location using primarily visual feedback. In this chapter, we propose learning viewpoint invariant visual servoing skills in a robot manipulation task. We train a deep recurrent controller that can automatically determine which actions move the end-effector of a robotic arm to a desired object. This problem is fundamentally ambiguous: under severe variation in viewpoint, it may be impossible to determine the actions in a single feedforward operation. Instead, our visual servoing approach uses its memory of past movements to understand how the actions affect the robot motion from the current viewpoint, correcting mistakes and gradually moving closer to the target. This ability is in stark contrast to previous visual servoing methods, which assume known dynamics or require a calibration phase. We learn our recurrent controller using simulated data, synthetic demonstrations and reinforcement learning. We then describe how the resulting model can be transferred to a real-world robot by disentangling perception from control and only adapting the visual layers. The adapted model can servo to previously unseen objects from novel viewpoints on a real-world Kuka IIWA robotic arm. For supplementary videos, see: https://www.youtube.com/watch?v=oLgM2Bnb7fo [1].

[1]This work is also summarized in [159]: https://ai.googleblog.com/2018/06/teaching-uncalibrated-robots-to_22.html

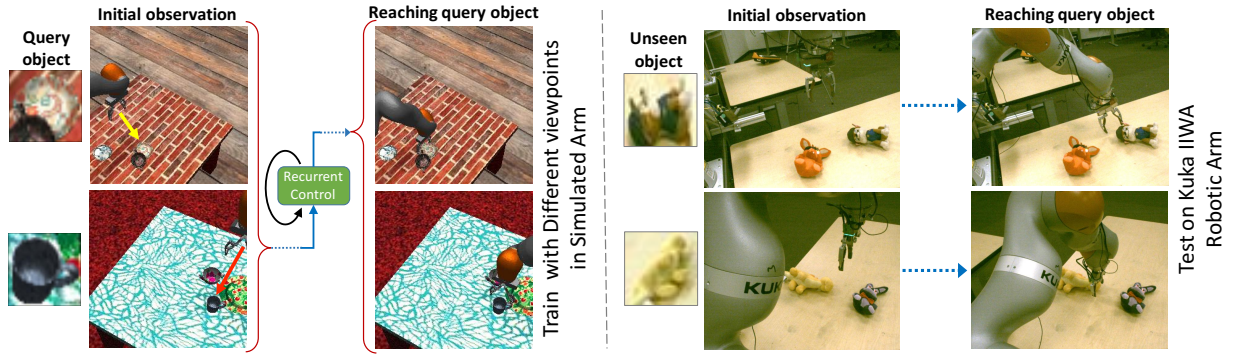

Figure 6.1: Illustration of our learned recurrent visual servoing controller. Training is performed in simulation (left) to reach varied objects from various viewpoints. The recurrent controller learns to implicitly calibrate the image-space motion of the arm with respect to the actions issued in the unknown coordinate frame of the robot. The model is then transferred to the real world by adapting the visual features, and can reach previously unseen objects from novel viewpoints (right).

6.1 Overview

Humans and animals can quickly recognize the effects of their actions through visual perception: when we see ourselves in a mirror, we quickly realize that the motion of our reflected image is reversed as a function of our muscle movements, and when we observe ourselves on camera (e.g., a security camera in a grocery store), we can quickly pick ourselves out from a crowd simply by looking for the motions that correlate with our actions. We can even understand the effects of our actions under complex optical transformations, such as in the case of a surgeon performing a procedure using a laparoscope. In short, we can quickly discover our own "end-effector" (either our own hand, or even a tool) and visually guide it to perform a desired task.

The ability to quickly acquire visual servoing skills of this sort under large viewpoint variation would have substantial implications for autonomous robotic systems: if a robot can learn to quickly adapt to any viewpoint, it can be dropped without any calibration into novel situations and autonomously discover how to control its joints to achieve a desired servoing goal. However, this poses a substantial technical challenge: the problem of discovering how the controllable degrees of freedom affect visual motion can be ambiguous and under-specified from a single frame. Consider the two scenes shown on the right side of Figure 5.1. Which way should the robot move its end-effector in order to reach the unseen query object? In the two settings, the action in the robot's

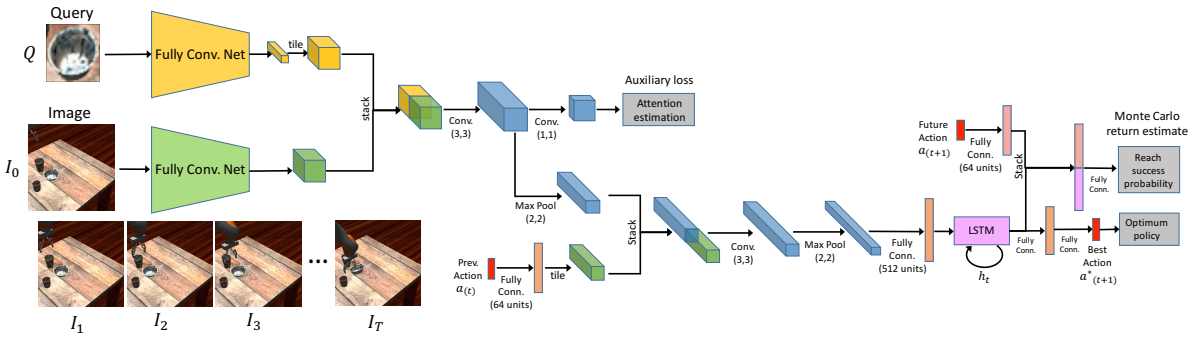

Figure 6.2: The input to our network is a query image (top-left) and the observed image at step t (left). The images are processed by separate convolutional stacks; their features are concatenated and are fed into an LSTM layer. The output is the policy (bottom right) which is an end-effector movement in the frame of the robot . The previously selected action is provided to LSTM, enabling it to implicitly calibrate the effects of actions on image-space motion. **Value prediction:** a separate head (top right) predicts the Q-value of the action trained with Monte Carlo return estimates. **Auxiliary loss:** An auxiliary loss function minimizes the localization error for the query object in the observed image. Also used in order to adapt the convolutional layers with a few labeled real images.

(unknown) coordinate frame has almost the opposite effects on the observed image-space motion. After commanding an action and observing the movement, it is possible to deduce this relationship. However, identifying the effect of actions on image-space motion and successfully performing the servoing task requires a robust perception system augmented with the ability to maintain a memory of past actions.

In this chapter, we show that view invariant visual servoing skills can be learned by deep neural networks, augmented with recurrent connections for memory. In classical robotics, visual servoing refers to controlling a robot in order to achieve a positional target in image space, typically specified by positions of hand-designed keypoint features [201, 89]. We instead take an open-world approach to visual servoing: the goal is specified simply by providing the network with a small picture of the desired object, and the network must select the actions that will cause the robot's arm to reach that object, without any manually specified features, and in the presence of severe viewpoint variation. This mechanism automatically and implicitly learns to identify how the actions affect image-space motion, and can generalize to novel viewpoints and objects not seen during training.

The main contribution of our work is a novel recurrent convolutional neural network controller that learns to servo a robot arm to previously unseen objects while it is invariant to the viewpoint. To learn perception and control for such viewpoint invariant servoing, we propose a training

procedure that uses automatically generated demonstration trajectories in simulation as well as reinforcement learning (RL) policy evaluation. We train our viewpoint invariant controller primarily in a randomized simulation setup where we generate diverse scenes. Using a small amount of real-world images, we adapt the visual features to enable successful servoing on a real robotic arm while the overwhelming majority of training data is generated in a randomized simulator. Our experimental results evaluate the importance of recurrence for visual servoing on an extensive simulated benchmark and show that incorporating the value prediction function improves the results. We also evaluate the effectiveness of our method in several real-world servoing scenarios both quantitatively and qualitatively.

6.2 Related Work

Visual servoing has a long history in computer vision and robotics [34, 89]. Our proposed visual servoing method aims to address a similar problem, but differs in several important aspects of the visual servoing problem formulation, with the aim of producing a method that is more general and practical for open-world settings. We depart from a common assumption that the camera intrinsics and extrinsics are calibrated [132, 54], and make no assumptions about the 3D structure of the scene [54, 132, 41]. Several prior visual servoing methods also address servoing with uncalibrated cameras [207, 95, 124], but all of them address an "eye-in-hand" setting, where the goal is to servo the camera toward a target view by using previously known geometric features and estimate the image Jacobian within an image based visual servoing setup. In contrast, our visual servoing setting involves servoing a robotic arm to a visually indicated target, provided via a query image while the camera viewpoint is unknown and changes between trials. In contrast to the the eye-in-hand setup, the query image is *not* the desired image that the camera should see, but rather an object that arm should reach while the camera observes the scene from an unknown viewpoint. This requires the servoing mechanism to learn to match visual features between the query object and current observation, recognize the motion of the arm, and account for differences in viewpoint between trials.

Specifying the target by providing an image of the query object, instead of specifying low-level

keypoints, is most similar to photometric visual servoing [30]. However, while photometric visual servoing aims to match a target image (e.g., by moving the camera), our method aims to direct the arm to approach the visually indicated object. The query image provides no information about *how* to approach the object, just which object is desired. Our model must therefore both localize the object and direct the robot's motion.

Similarly to recent work on self-supervised robotic learning [113, 114, 6, 65, 144], our method uses observed images and self-supervision to train deep convolutional networks to predict action-conditioned task outcome. In contrast to these prior methods, our camera viewpoint is not fixed and can change drastically from one episode to another. Our approach incorporates fast adaptation via recurrence to adapt the visual servo to a novel viewpoint within a single episode of task execution, in addition to an outer-level self-supervised learning process performed with conventional gradient descent.

The use of recurrent networks for control has previously been used in a number of works on reinforcement learning, including methods for visual navigation [130, 137], continuous control [78, 214], and physics simulation to real world transfer [142] without visual observations. However, to our knowledge, no prior method has demonstrated that recurrence can be used to enable real-world robotic visual servoing from novel viewpoints. The closest work to this topic has taken a system identification approach for unknown physical parameters, such as masses and friction parameters [210] and does not use either of image observation or recurrence.

We use randomized simulated experience to train our visual servoing system. In order to use it for visual servoing in the real world, we also introduce an adaptation procedure based on finetuning of the visual features with an auxiliary objective. Most prior approaches to domain adaptation either train domain invariant representation [120, 24, 66, 23], learn a representation transformation from one domain to another via transfer learning [72, 195, 158], or employ domain randomization in simulation [162, 188] which produces robust models that can generalize broadly and can directly be deployed in the real world. Our approach combines domain randomization with transfer learning: we learn the controller entirely in a randomized simulation environment and then finetune only the visual features with a small amount of real world images, effectively transforming the model's

Figure 6.3: Our randomized simulated scenes and viewpoint randomization to learn viewpoint invariant visual servoing skills.

representation into the real-world domain. We show that our final finetuning procedure produces an effective visual servoing mechanism in the real world, even though the recurrent motor control layers are not finetuned on the real-world data.

6.3 Recurrent View Invariant Visual Servo

Our aim is to learn visual servoing policies that can generalization to new viewpoints. To this end, our policy network should implicitly learn to "self-calibrate" and discover the relationship between motor commands and motion in the image. We set up visual servoing scenarios where a robot arm must reach objects using monocular camera observations captured from an arbitrary viewpoint. The reaching target is indicated by an image of the query object, and the network must figure out where this object is in the image and how to actuate the robot arm in order to reach it.

The principal challenge in this problem setup comes from the inherent ambiguity over the motion of the arm in the image in terms of the actions. Most standard visual servoing methods assume knowledge of the Jacobian – the relationship between actions and motion of desired visual features. Our approach not only has no initial knowledge of the Jacobian, but it does not even have any prior visual features, and must learn everything from data. Determining the right actions from a single image is generally not possible. Instead, we must incorporate temporal context, using the outcomes of past actions to inform future ones. To that end, we use a recurrent neural net (RNN), whose internal state is trained to extract and capture knowledge about hand-eye coordination during the course of a single episode. Our network must take a few initial actions, observe their outcomes, and implicitly "self-calibrate" to understand how actions influence image-space motion. We train this recurrent controller over a large number of randomized scenes and show that it generalizes to novel scenes, with variability in both viewpoint and object appearance.

Another challenging aspect of our problem is the visual scene complexity. We generate diverse simulated scenes for each episode by randomly selecting a query object and distractor objects from a set of 3D shapes and rendering scene components(table, plane, objects) with random textures and under varied lighting conditions. The objects are placed on the table with random location and orientation. This setup enforces the model to learn to distinguish between objects and as such it needs to implicitly perform object localization in 3D.

We denote our controller model as π_θ. This model is a function, with parameters θ, that accepts as input the current image observation o_t, the query image q as well as the previous internal state h_{t-1} representing the memory. The previously chosen action a_{t-1} is also provided, so that it can infer how actions affect image-space motion. The output consists of the action a_t and the new internal state h_t, such that the policy is defined as

$$a_{t+1}, h_{t+1} = \pi_\theta(o_t, a_{t-1}, q, h_t) \tag{6.1}$$

We implemented recurrence in our controller using an LSTM, and the action is defined as a displacement $a = (\partial x, \partial y, \partial z)$ of the end-effector of the arm in the robot's frame of reference (which is not known to the model). For the purpose of exploration, the policy that is used to collect expe-

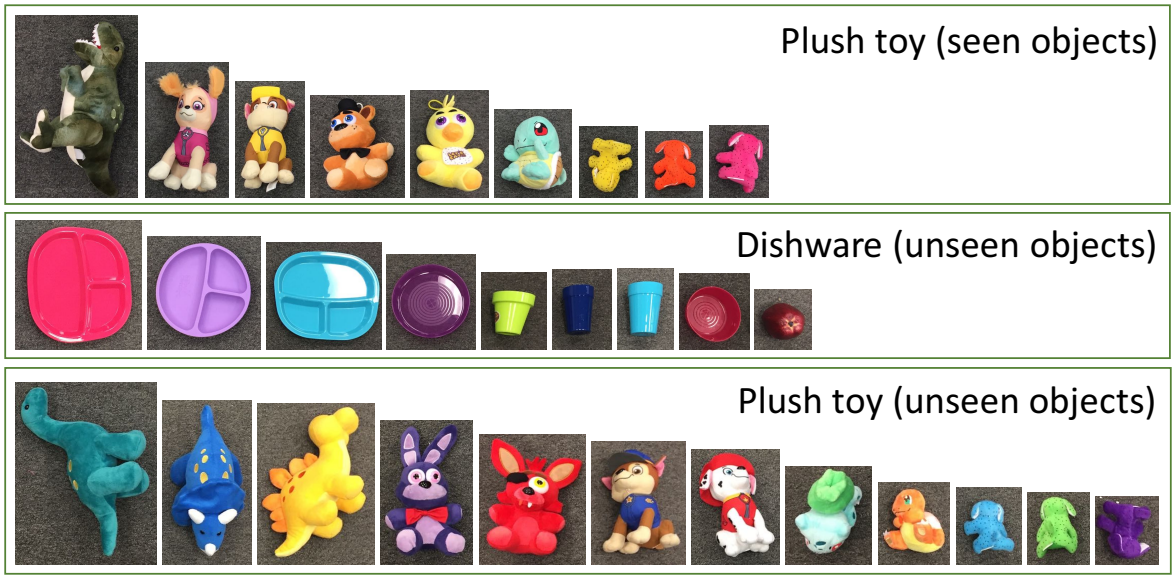

Figure 6.4: The set of objects used in the real-world experiments. The seen plush toys are used for adapting the visual layers to natural images, while the unseen objects are used for testing.

rience during training is stochastic, and corresponds to a Gaussian with mean given by the model output. When the model is used to select actions for T steps, it produces a sequence of observation and actions $\tau = (o_1, a_2, \cdots, o_T, a_T)$. As illustrated in Fig. 6.2, the observation o_t and query image q are processed with separate convolutional stacks based on the VGG16 architecture [178], with o_t having an input size of 256×256 and q resized to 32×32. These networks are trained from scratch, without pretraining and produce vector representations of both images. The previous action a_{t-1} is transformed via a fully connected ReLU layer into a 64-dimensional feature vector. The observation embedding at step t, the query embedding and the action embeddings are concatenated in one vector as input to the recurrent motor control system, which uses a single-layer LSTM with 512 units [84]. The state of this LSTM corresponds to the memory h_{t-1}, which allows the model to capture information from past observations needed to perform implicit calibration.

6.4 Training

We train our visual servoing model with a combination of supervised learning, which is analogous to learning from demonstration, and outcome prediction, which corresponds to RL objective. In our implementation, the model is trained entirely in simulation (see Sec. 6.5), which provides full

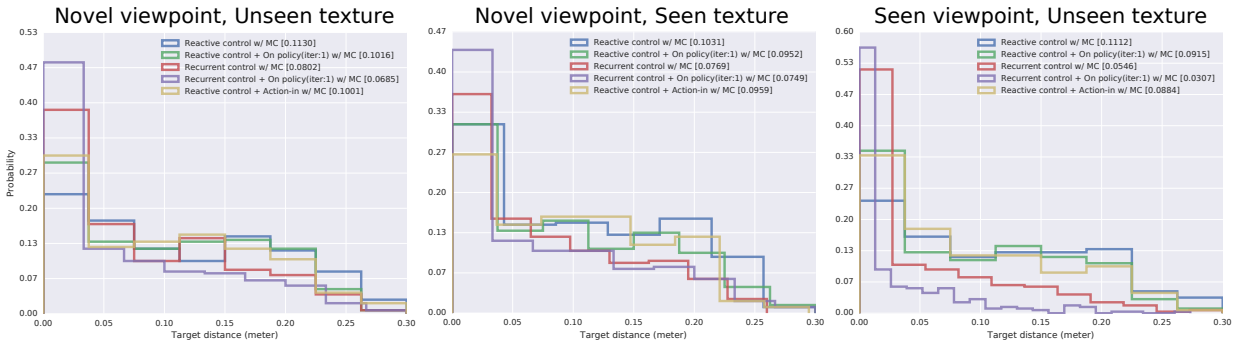

Figure 6.5: Comparing recurrent vs reactive control in test scenarios with different levels of difficulty and three random objects.

access to object locations and robot states. This allows us to produce supervision that corresponds to ground truth actions or synthetic "demonstrations." These demonstrations directly supervise the action output a_t. However, this supervision does not directly teach the long term effects of an action to the network. We found it beneficial to also augment the training process with a value prediction loss for RL. This loss trains the model to also predict the state-action value function associated with each action using multi-step Monte Carlo policy evaluation, which is the reward that the model expects to obtain for the entire episode by following its policy to take actions. As shown in our experiments, this RL loss leads to improved performance, since the resulting internal representations become better adapted to the long-term goal of the task.

6.4.1 Learning from Synthetic Demonstration

We synthesize demonstration trajectories by generating a large set of episodes with varied camera locations, objects, and textures as described in Sec. 6.5. Each episode contains ground truth actions that servo the arm to the query object, perturbed by Gaussian noise to provide some degree of exploration, which we found beneficial for producing robust polices. The training loss corresponds to the sum of squared Euclidean distances between the output action and the vector from the end-effector to the target object and can be written as

$$Loss = \sum_{t=1}^{T} || \frac{y - x_t}{||y - x_t||} - a_t ||^2. \tag{6.2}$$

To keep the action magnitudes within a bound, we learn normalized action direction vectors and use constant velocity so that a_t is independent of the number of steps. Here, we are not proposing a planning method for arbitrary tasks (e.g. presence of obstacles, etc.), but specifically aim to solve a visual servoing task, where the robot should move *directly* toward an object. Therefore, at each time t, the optimum action is the normalized direction vector towards the object and is computed by subtracting the end-effector position x_t from the object location y. The sampled trajectories provide starting points of the arm and past actions from which the model needs to recover. After unrolling the policy and formulating the above loss we use stochastic gradient descend over the parameters θ to minimize $Loss$. Following the DAgger framework [154], once our model converges, we generate additional on-policy trajectories by running the current policy and label them with the ground truth actions. This new data is then used to retrain the model and we repeat this procedure for two iterations.

6.4.2 Learning the Value Function

The above supervised learning procedure can quickly lead to a reasonable policy, but it is also myopic, in the sense that it does not consider the long term effects of an action. We found that the final performance of our model could be improved by also incorporating an RL objective, in the form of state-action value function prediction, also known as the Q-function [183]. This allows us to then select the action that minimizes the predicted long term reward. We formulate a reward function that indicates whether the arm has reached the target at the end of the episode, such that $r(s_t, a_t) = 1$ and $r(s_t, a_t) = 0$ otherwise. Here, we use s_t to denote the (unobserved) underlying state of the system, which includes the arm pose and query position. The target Q-values are then computed according to

$$Q(s_t, a_t) = r(s_t, a_t) + E_{\tau \sim \pi_\theta} \left[\sum_{t'=t+1}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'}) \right]$$

where γ is a discount factor. These target values are used with a squared error regression loss applied to a second head of the model (see Fig. 6.2). The rewards and target Q-values are computed

along trajectories sampled by running the policy. In practice, we found it beneficial to unroll the policy multiple times from each visited state and action and average together the corresponding returns to estimate the expectation with respect to π_θ. This corresponds to multi-step Monte Carlo policy evaluation [183], which provides a simple and stable method for learning Q-functions [162]. We optimize with respect to the input action a_t to choose actions according to this Q-function. In our implementation, we use cross-entropy method (CEM) [156] to perform this optimization, which provides a simple gradient-free optimization procedure for choosing actions that maximize predicted Q-values.

We implement our models in TensorFlow [1]. We use a buffer of one million unrolls for each policy learning round and deploy the Adam optimizer [102] with a learning rate of $1.5e-5$ and an exponential decay schedule. Each round of training converges after one million steps.

6.5 Simulated Training and Transfer

One of the main challenges in robot learning is data collection. While deep models are shown to work well on huge amount of data, large scale robot data collection is infeasible in many scenarios and results in challenges for training models with high generalization. To address this challenge, we train our controller in simulation where we can generate a large, diverse range of scenes captured from various viewpoints and obtain the supervision needed to train our model efficiently. We use domain randomization [162] to learn robust visual features and boost our performance in the real world by also incorporating visual adaptation with small amount of real world images.

Simulated Environment: We use the Bullet physics engine simulator [53], with a simulated 7 DoF Kuka IIWA arm and a variety of objects placed on a table in front of the arm. The objects are randomly selected from a set of 50 scanned objects of various dishware – plates, mugs, cups, bowls, teapots and etc. The objects are dropped on the table, so that their pose and location is randomized. We also randomize textures, lighting, and the appearance of the table and ground plane. This randomization procedure serves two important purposes: first, it forces the controller to learn the underlying geometric patterns that are actually important to the task, rather than picking up on extraneous aspects of object appearance that might correlate with actions, and second, it serves

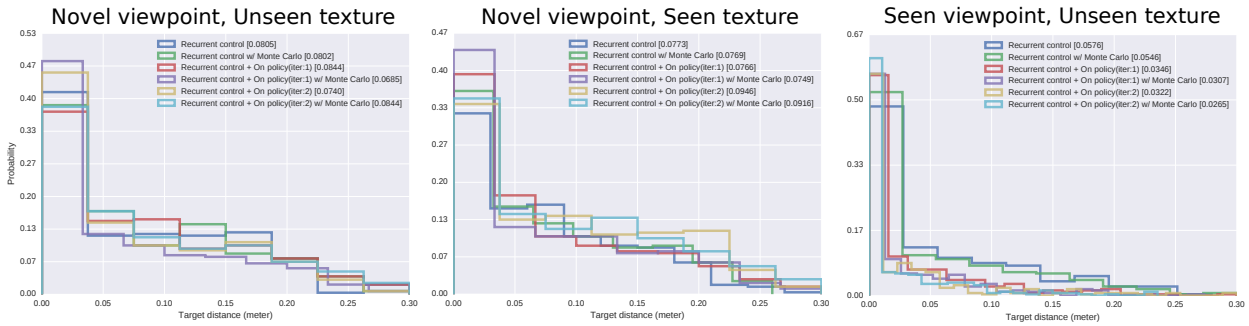

Figure 6.6: Comparison for different iterations of on-policy data collection, and the benefit of value prediction objective by using Monte-Carlo policy evaluation with three object test scenarios. Test scenarios with different levels of difficulty from left to right.

to enable the model to generalize more easily to real-world scenes, by essentially forcing it to solve a harder generalization task (widely different appearances), as discussed in prior work [162]. Each simulated trial consists of a random camera viewpoint, up to three randomly selected objects and randomized appearance parameters. Fig. 6.3 shows examples of our randomized simulation environment from various robot camera viewpoints.

Adaptation to the Real World: To perform visual servoing with a real robotic arm the model parameters should be able to generalize to real world. The randomization procedure described in the previous section already provides some degree of generalization [188, 162]. We also found that an additional adaptation step can improve generalization to real world scenarios. Obtaining ground truth actions and rewards in the real world requires costly manual labeling. Instead, we leverage the fact that the motor control portion of our model can remain largely unchanged between simulation and the real world, and only the visual features can be adapted using a weaker form of supervision to finetune only the convolutional layers of the model.

To that end, we use the auxiliary adaptation loss at the last layer of the visual stack (see Fig. 6.2, top-left), which predicts the presence or absence of the query object on a 8×8 grid overlaid on the image, by using computed logits at the last fully convolutional layer. These logits are fed into a cross entropy loss to finetune the vision stack. We use 22 sequences of the arm executing random actions, and we annotate the first frame in each video with bounding boxes for the objects that are present on the table which resulted in a total of 76 bounding boxes. Some of these scenes are

Table 6.1: Average distance to target in meter for various test settings with two and three objects scenes. (VP: Viewpoint, T: Texture)

	Three objects			Two objects		
	Novel VP Unseen T	Novel VP Seen T	Seen VP Unseen T	Novel VP Unseen T	Novel VP Seen T	Seen VP Unseen T
Reactive w/ MC	0.1130	0.1031	0.1112	0.1062	0.1067	0.1051
Reactive + On policy w/ MC	0.1016	0.0952	0.0915	0.0935	0.0909	0.0950
Reactive+Action-in w/ MC	0.1001	0.0959	0.0884	0.0990	0.0938	0.0898
Recurrent w/ MC	0.0802	0.0769	0.0546	0.0730	0.0757	0.0461
Recurrent + On policy w/ MC	0.0685	0.0749	0.0307	0.0678	0.0741	0.0226

Table 6.2: Average distance to target in meter for evaluating the effect of the value prediction loss by using Monte-Carlo policy evaluation (MC) and on-policy data. (VP: Viewpoint, T: Texture)

	Three objects			Two objects		
	Novel VP Unseen T	Novel VP Seen T	Seen VP Unseen T	Novel VP Unseen T	Novel VP Seen T	Seen VP Unseen T
Recurrent	0.0805	0.0773	0.0576	0.0729	0.0819	0.0437
Recurrent w/ MC	0.0802	0.0769	0.0546	0.0730	0.0757	0.0461
Recurrent + On policy	0.0844	0.0766	0.0346	0.0747	0.0751	0.0235
Recurrent + On policy w/ MC	0.0685	0.0749	0.0307	0.0678	0.0741	0.0226
Recurrent + On policy(iter:2)	0.0740	0.0946	0.0322	0.0883	0.0863	0.0270
Recurrent + On policy(iter:2) w/ MC	0.0844	0.0916	0.0265	0.0865	0.0903	0.0229

shown in Fig. 6.10. Since the episodes remain stationary during each episode, we can propagate the labels automatically through each sequence. The actual loss is constructed by sampling a batch of sequences, and for each sequence sampling one object to use as the query object by cropping out one bounding box. To make our localization robust against object poses, for each sequence we randomly select the query image from a pool of query images of the same query object category. The loss then describes the error in localizing the query object in the spatial image frame.

6.6 Experiments

Our experiments consists of detailed evaluations in simulation as as well as real-world evaluation with Kuka IIWA arm to study generalization to real-world. Prior visual servoing methods are not directly applicable to our problem setting, since we do not assume knowledge about the camera position or the action to image Jacobian. Also our target is specified by a picture of the object that

the arm should reach for. Therefore, we compare to ablated variants of our method. To evaluate the importance of memory for learned implicit self-calibration, we compare to non-recurrent visual servo architectures trained in exactly the same way as our method. We also analyze the importance of combining supervised learning from demonstrated trajectories with RL value prediction.

6.6.1 Simulated Reaching

In our simulation experiments, we aim to answer the following questions: (1) How effective is our proposed recurrent controller compared to a feedforward reactive policy? (2) How robust is our model to viewpoint variation and visual diversity? (3) What is the benefit of incorporating on-policy data? (4) How beneficial is the value prediction objective?

Model setup: In addition to the recurrent controller, we also train two reactive non-recurrent policies which may or may not take the previous action as input and use a feed-forward network which has same layers as in Fig. 6.2, but has two fully connected layers instead of the recurrent LSTM layer. We call these baseline architectures as reactive+action-in and reactive policies, respectively.

Test setup: For testing, we generate new simulated scenarios using the randomization procedure described in Sec. 6.5. We generate scenes with two or three novel objects not seen during training. The test viewpoints are sampled uniformly at random in a region around the workspace, with the orientation chosen to always point toward the table, such that the query object is visible. We randomly select of the viewpoints for training episodes, while keeping a held-out set of test viewpoints to test generalization. We evaluate the performance of our method on test scenarios with different levels of difficulty: (a) Novel textures and novel viewpoints. (b) Previously seen textures and novel viewpoints. (c) Novel textures and previously seen viewpoints.

Evaluation criteria: In each simulation experiment, we run the policy for 300 trials, each with a fixed length of 10 steps. At the end of each trial, we compute the Euclidean distance of the robot's end-effector to the query object, using the closest points on the arm and the object mesh. This metric is in meters, and is zero when the arm touches the object. We report the average distance to the query object is the last time step over 300 test trials. We visualize the distribution of the final distances attained at the end of the trials in Figure 6.5, and report the average distances in

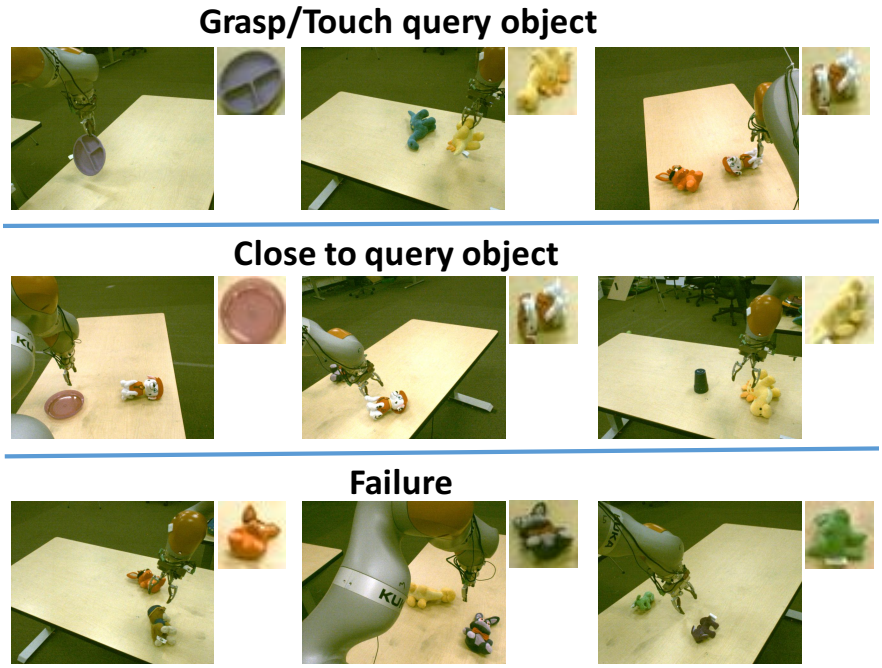

Figure 6.7: A successful reach occurs if the gripper touches the query object or gets very close to it. Failure examples include cases where the gripper ends up with a far distance from the query object or approaching the wrong object.

Table 6.1.

Reactive vs. recurrent policies: According to the final distance distributions illustrated in Figure 6.5, both reactive policies are substantially less proficient at reaching the query objects. When the testing scenario is the most challenging, with novel textures and novel viewpoints, and without any on-policy data collection, the average final distance obtained by the reactive policies are 0.10m and 0.11m, while the recurrent policy reaches an average final average distance of 0.08m. Incorporating on-policy data for training our proposed approach results in a final distance of 0.07m in the novel viewpoint and unseen texture condition. The results also indicate that the novel camera viewpoints are indeed more challenging when it comes to generalization. According to Table 6.1 there is $\sim$ 4-6 cm difference between reactive and recurrent controller performance which is 57%-86% of the open gripper (with width of 7cm), respectively. This is significant for robotic applications.

Random policy: We compared our performance with random walk policy for commanding the end-effector. The random policy obtains average euclidean distance of 0.169m to the target object in 300 trials of 10 steps which validates the promising performance of our learned recurrent policy.

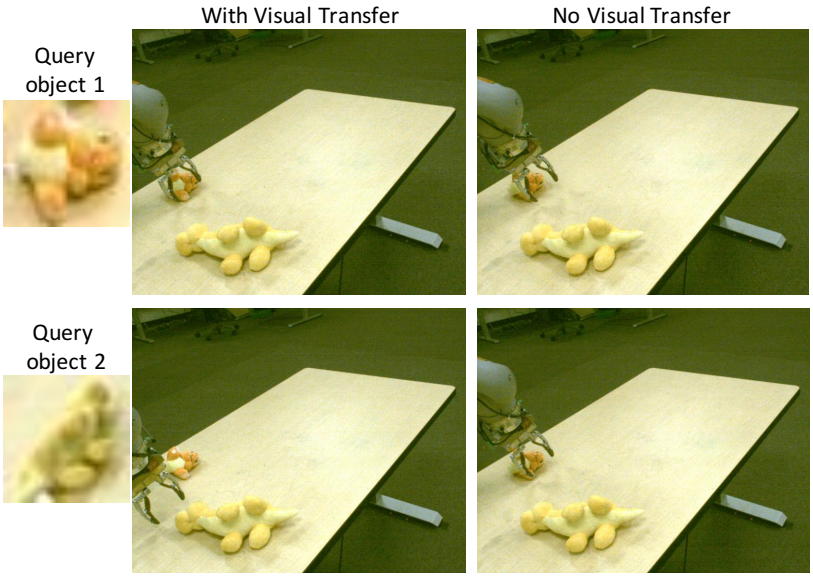

Figure 6.8: The network trained with only simulated data becomes confused between two objects with similar color and fails in the reaching task, while the visually adapted network can distinguish between the two object.

The effect of using on-policy data: The effect of using different numbers of iterations of on-policy training is shown in Figure 6.6 and also summarized in Table 6.2. Performing two iterations of retraining with on-policy data produces the best performance on the seen viewpoints and unseen texture scenarios, while resulting in poorer performance in the scenarios with novel viewpoints. On the other hand, using one iteration of retraining with on-policy data improves the performance in all scenarios. This result suggests that one iteration of on-policy data collection can address the distribution mismatch problem, though additional iterations can potentially result in becoming more specialized to the training viewpoints. Therefore, if the task emphasis is on mastery, using more on-policy data can improve performance.

The effect using Monte Carlo policy evaluation for value prediction : To evaluate the effectiveness of the value prediction loss which uses multi-step Monte-Carlo (MC) policy evaluation, we conducted simulated experiments with and without the value prediction loss. When the value prediction loss is used, it is denoted by w/Monte Carlo in Figure 6.6 and Table 6.2. When using w/Monte Carlo, we compute the action based on the action prediction output of the model, at each time step. We then use CEM to perturb this action with Gaussian noise with standard deviation $\sigma = 0.003$ to generate 150 candidate actions and evaluate them via the value prediction head. The

Table 6.3: Real world reaching task results with novel viewpoints (Percentage of successful trials).

	Simulation only controller			Visually adapted controller		
	Success rate	Grasp/Touch query object	Close to query object	Success rate	Grasp/Touch query object	Close to query object query object
One object	88.9	55.6	33.3	94.4	61.1	33.3
Two objects	54.1	33.3	20.8	70.8	25.0	45.8

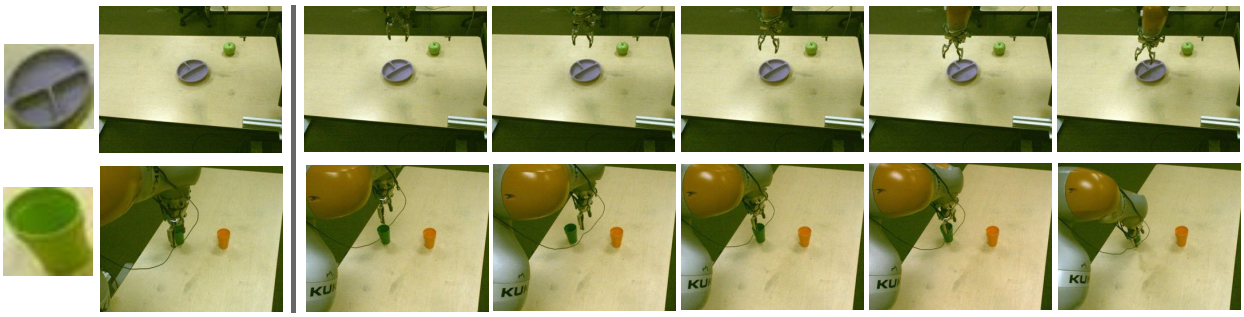

Figure 6.9: In both scenarios, the arm successfully reaches the object. Note that, in the second sequence, the arm first moves to the right, and then observes the effect of this action and corrects, moving toward the query object. This suggests that the controller can observe action outcomes and incorporate these observations to correct servoing mistakes.

executed action is sampled at random from the top 5 actions with highest values. The results in Figure 6.6 shows that, in most conditions, incorporating the value prediction head which is trained using Monte-Carlo return estimates results in improved performance.

6.6.2 Real-World Robotic Reaching

We evaluated the generalization capability of our viewpoint invariant visual servoing model on a real 7DoF Kuka IIWA robotic arm. We used two sets of novel objects for the test experiments. The test objects include plush toys with different colors and shapes, as well as different dishware objects, such as cups, plates, and bowls. These objects are shown in Figure 6.4. In the experiments, we placed the camera at various locations and arranged the table with objects at arbitrary locations and poses.

Quantitative Results: In our real-world evaluation, we used different objects placed at arbitrary locations on a table, and placed the camera at various locations. Our experiments compare our recurrent controller with adapted visual layers to one that was trained entirely in simulation without

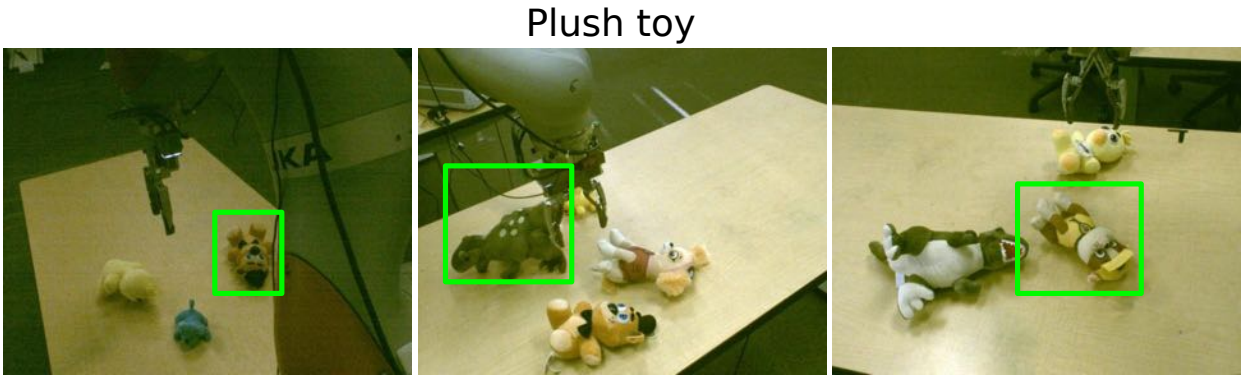

Figure 6.10: Examples of real-world scenes used for testing.

any additional adaptation. Here, we compare our recurrent controller with adapted visual layers to one that was trained entirely in simulation without any additional adaptation. The two models were compared head-to-head on each viewpoint and object arrangement, to provide a low-variance comparison. The tests were divided into scenarios with either one or two objects on the table. Table 6.3 summarizes the performance on the real-world reaching task. We performed a total of 42 trials, 18 with single objects, and 24 with two objects. These trials were recorded from a total of 18 camera viewpoints. We count the number of times the arm moves towards the right query object and reaches it. A successful reach occurs if the gripper touches the object or gets very close to it. If the arm moves in the wrong direction or is confused between the two objects, we count the trial as a failure. We added a fixed procedure at the end of each robot trial to model a pointing action. In this procedure, the gripper is first closed and the arm is pulled up. Then the arm is moved downward and the gripper is opened. Note that, while our model is trained for reaching, and not particularly for the grasping task, using the aforementioned procedure can sometimes result in a successful grasp.

Figure 6.8 illustrates examples of successful and unsuccessful reaching attempts. The first row of this figure shows trials where the gripper touches the object or grasps it. The second row shows successful reach attempts where the gripper gets very closed to the query object, and is far from the distractor object. The last row shows several failure cases, where the controller is confused between two objects and the trial ends with the gripper at a considerable distance from the query object.

The single object scenarios provide a simpler test setting, where success is mainly dependent on the ability of the method to determine which actions move the arm toward the object. On the other hand, the two-object scenarios require the model to both generalize to a novel viewpoint and distinguish the query object from the distractor. This is significantly more challenging, especially since the test objects differ significantly from the simulated objects seen during training. As seen in Table 6.3, adapting the visual features with a small amount of real-world data substantially improves the performance of the network in both scenarios, with a success rate of 70.83% in the harder two-object setting. Table 6.3 summarizes the outcome of successful trials in detail and outlines the percentage of trials that result in the gripper touching or grasping the object.

Qualitative Results: We visualize two interesting reaching sequences. In Figure 6.9 we see successful reaches with exploratory motions, where the arm first moves in the wrong direction, then observes the image-space motion and corrects. In Figure 6.8, we observe that the network that is entirely trained in simulation makes more mistakes when the query object and distractor object are visually similar. The network after adaptation is more robust to these kinds of visual ambiguities. For supplementary videos with more qualitative results, see:
https://fsadeghi.github.io/Sim2RealViewInvariantServo.

6.7 Discussion

In this chapter, we described a learning-based visual servoing approach which can automatically and implicitly "self-calibrate" a robot in the process of a manipulation task from an unseen viewpoint. Our method is based on training a deep convolutional recurrent neural network that can control a robot to reach user-specified query objects, implicitly learning to identify the effects of actions in image-space from the past history of observations and actions. The network is trained primarily in simulation, where supervised demonstrated data is easy to obtain automatically, and a novel adaptation procedure is used to adapt the visual layers of this model to the real world, using only a small number of labeled images. An exciting direction to explore in future work is how more complex manipulation skills can be performed from any viewpoint using a similar approach as well as incorporating meta learning for fast adaptation.

Chapter 7

CONCLUSION

"If you ask one question, it will lead you to another, and another, and another..."

Lemony Snicket, *The End*

In this thesis, we started by working on visual semantic reasoning where we incorporated visual cues obtained from detecting objects on RGB images. We observed that object presence in images can be used as a strong source of information for high level semantic reasoning on the spatial 2D image space [161]. With this background, we then worked on our core idea of using visual semantics for learning highly generalizable vision-based robot policies. Our goal was to devise techniques that enable robots to incorporate semantics and visual feedback for learning versatile policies that can generalize to diverse real world situations. To this end, we first explored how to learn vision-based policies in simulation that can directly be applied in the real world. We showed that by highly randomizing the rendering settings of simulation during training, we can bridge the sim-to-real gap and learn policies that generalize to new unseen environments with substantially different appearance from the training scenarios as well as real world. Our simulation randomization work, CAD^{2}RL, introduces the domain randomization technique as an orthogonal approach to conventional transfer learning approaches for learning policies in simulation such that can generalize to real world and we studied the applicability of this technique for several problems: (a) Indoor robot navigation with a drone, (b) semantic goal reaching with mobile robots, and (c) object manipulation with a robotic arm [162, 160, 164].

After introducing simulation randomization (domain randomization), we worked on deploying semantics into vision-based robot policies. We proposed a new method for learning goal-oriented semantic navigation of mobile robots in simulation that can be directly transferred to various real

world environments. Our Domain Invariant Visual Servoing (DIViS) approach incorporates visual semantics from static images into policy learning and thus provides a data efficient approach for robotic learning. We showed that DIViS can learn domain invariant visual servoing policies entirely in simulation and is capable of generalization to a variety of unseen and unstructured real world environments [160].

We then studied how pure visual feedback can be used for learning adaptive behavior by presenting a new problem of viewpoint invariant visual servoing with a robotic arm. We devised a visual servoing method that maintains past history of observations and actions for learning vision-based adaptive behavior in object reaching with a robotic manipulator [164]. This approach was inspired from visual analogies over a sequence of images [165].

In addition to that, we showed how having full control over the simulation environment can allow us collect unlimited number of robot trials which is especially beneficial for complex tasks that usually need data collected by human demonstration. We harnessed the simulation capabilities to generate synthetic demonstrations and combined reinforcement learning objectives to learn a robust controller for a robotic arm. Also, we developed a new deep network architecture for learning vision-based robotic policies which disentangles perception from control and facilitates policy transfer from simulation to real with minimal amount of real world data [164].

The work we presented in this thesis is the first step towards tackling the problem of learning highly generalizable and domain invariant vision-based robot policies through incorporating diverse 3D simulations as well as deploying visual semantics. Here, we discuss several directions that we believe are compelling for future work:

Simulation to Real Policy Transfer for Complex Skills: In this thesis, we took the first step towards developing techniques for direct simulation to real transfer of robot policies in the context of simpler tasks such as short range navigation and object reaching. Considering complex and long horizon tasks is a next step towards devising better algorithms for simulation to real world transfer. An example of such long horizon tasks can be long range multi-room navigation or constructing complex object arrangement with a robotic manipulator.

Generalization in Dynamic Environments: We proposed several approaches for learning vision-

based robot policies with high generalization. While we showed high generalization in unstructured environments that were not seen at the training time, the target scene and objects evolved were mostly stationary and did not dynamically change during the course of each robot trial. In a more challenging setup, the environment itself can exhibit dynamic behavior and exploring techniques for learning generalizable policies in highly dynamic environments is an exciting direction for future research.

Lifelong Learning: Just like humans, intelligent machines should be able to digest high volumes of sensory input and reason how to act when multiple control problems are involved in complex tasks. We showed how to provide large volumes of cheap data in simulation for policy learning. The next key ingredient to mimic human learning is to simulate a lifelong learning experience for intelligent agents. Also, if robots are capable of continuing to learn they will be able to increment their skills as well as adapting to unseen situations. In particular, the paradigm of lifelong and continual learning combined with the domain invariant policy learning can provide promising directions for evolving robots to perform large sets of skills that can generalize and adapt to new situations.

BIBLIOGRAPHY

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

[3] P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *IJRR*, 2010.

[4] P. Abbeel, A. Coates, M. Quigley, and A. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *NIPS*, 2006.

[5] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in neural information processing systems*, pages 1–8, 2007.

[6] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, 2016.

[7] A. Akbik and T. Michael. The weltmodell: A data-driven commonsense knowledge base. In *LREC*, 2014.

[8] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2018.

[9] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[10] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual question answering. In *ICCV*, 2015.

[11] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *CVPR*, 2014.

[12] A. Bachrach, R. He, and N. Roy. Autonomous flight in unstructured and unknown indoor environments. In *EMAV*, 2009.

[13] R. Baillargeon and J. DeVos. Object permanence in young infants: Further evidence. *Child development*, 62(6):1227–1246, 1991.

[14] M. Banko et al. Open information extraction from the web. In *IJCAI*, 2007.

[15] D. M. Barbella and K. D. Forbus. Analogical dialogue acts: Supporting learning by reading analogies in instructional texts. In *AAAI*, 2011.

[16] M. Baroni and A. Lenci. Distributional memory: A general framework for corpus-based semantics. *Comput. Linguist.*, 2010.

[17] A. J. Barry and R. Tedrake. Pushbroom stereo for high-speed navigation in cluttered environments. In *ICRA*. IEEE, 2015.

[18] R. Basri, E. Rivlin, and I. Shimshoni. Visual homing: Surfing on the epipoles. *International Journal of Computer Vision*, 1999.

[19] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, 2013.

[20] C. Bills, J. Chen, and A. Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *ICRA*, 2011.

[21] Blender Community. Blender: Open Source 3D modeling suit. `http://www.blender.org`.

[22] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[23] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *arXiv preprint arXiv:1709.07857*, 2017.

[24] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *NIPS*, 2016.

[25] J. G. Bremner. *Infancy*. Blackwell Publishing, 1994.

[26] J. G. Bremner, A. M. Slater, and S. P. Johnson. Perception of object persistence: The origins of object permanence in infancy. *Child Development Perspectives*, 9(1):7–13, 2015.

[27] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[28] A. Carlson, J. Betteridge, E. R. Hruschka, Jr., and T. M. Mitchell. Coupling semi-supervised learning of categories and relations. In *NAACL*, 2009.

[29] A. Carlson et al. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[30] G. Caron, E. Marchand, and E. M. Mouaddib. Photometric visual servoing for omnidirectional cameras. *Autonomous Robots*, 2013.

[31] K. Celik, S. Chung, M. Clausman, and A. Somani. Monocular vision SLAM for indoor aerial vehicles. In *IROS*, 2009.

[32] M. D. Chang and K. D. Forbus. Using analogy to cluster hand-drawn sketches for sketch-based educational software. *AI Magazine*, 2014.

[33] M. D. Chang, J. W. Wetzel, and K. D. Forbus. Spatial reasoning in comparative analyses of physics diagrams. In *Spatial Cognition IX*, 2014.

[34] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 2006.

[35] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[36] X. Chen, A. Shrivastava, and A. Gupta. NEIL: Extracting visual knowledge from web data. In *ICCV*, 2013.

[37] Y. F. Chen, M. Liu, M. Everett, and J. How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. *arXiv preprint arXiv:1609.07845*, 2016.

[38] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[39] P. Clark, P. Harrison, and N. Balasubramanian. A study of the knowledge base requirements for passing an elementary science test. In *AKBC*, 2013.

[40] C. Collewet and E. Marchand. Photometric visual servoing. *IEEE Transactions on Robotics*, 2011.

[41] C. Collewet, E. Marchand, and F. Chaumette. Visual servoing set free from image processing. In *ICRA*, 2008.

[42] P. I. Corke. Visual control of robot manipulators–a review. World Scientific, 1993.

[43] P. I. Corke and S. A. Hutchinson. Real-time vision, tracking and control. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. IEEE, 2000.

[44] M. Cutler and J. P. How. Efficient reinforcement learning for robots using informative simulated priors. In *ICRA*, 2015.

[45] M. Cutler, T. J. Walsh, and J. P. How. Reinforcement learning with multi-fidelity simulators. In *ICRA*. IEEE, 2014.

[46] M. Cutler, T. J. Walsh, and J. P. How. Real-world reinforcement learning via multifidelity simulators. *IEEE Transactions on Robotics*, 2015.

[47] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[48] S. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014.

[49] C. Doersch, A. Gupta, and A. A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*. 2013.

[50] A. Dosovitskiy and V. Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

[51] D. Downey, O. Etzioni, and S. Soderland. A probabilistic model of redundancy in information extraction. In *IJCAI*, 2005.

[52] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014.

[53] Y. B. Erwin Coumans. pybullet, a python module for physics simulation in robotics, games and machine learning. `http://pybullet.org/`, 2016–2017.

[54] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *ieee Transactions on Robotics and Automation*, 1992.

[55] M. Everingham et al. The PASCAL Visual Object Classes (VOC) challenge. In *IJCV*, 2010.

[56] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

[57] A. Fader, L. Zettlemoyer, and O. Etzioni. Open question answering over curated and extracted knowledge bases. In *KDD*, 2014.

[58] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009.

[59] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009.

[60] O. Faugeras and O. A. FAUGERAS. *Three-dimensional computer vision: a geometric viewpoint*. MIT press, 1993.

[61] P. Felzenszwalb et al. Object detection with discriminatively trained part based models. *PAMI*, 2010.

[62] K. Forbus, J. Usher, A. Lovett, K. Lockwood, and J. Wetzel. Cogsketch: Sketch understanding for cognitive science research and for education. *Topics in Cognitive Science*, 2011.

[63] K. D. Forbus, J. M. Usher, and E. Tomai. Analogical learning of visual/conceptual relationships in sketches. In *AAAI*, 2005.

[64] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell. Speaker-follower models for vision-and-language navigation. *arXiv preprint arXiv:1806.02724*, 2018.

[65] D. Gandhi, L. Pinto, and A. Gupta. Learning to fly by crashing. *arXiv preprint arXiv:1704.05588*, 2017.

[66] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 2016.

[67] D. Geman, S. Geman, N. Hallonquist, and L. Younes. Visual turing test for computer vision systems. *PNAS*, 2015.

[68] D. Gentner, K. J. Holyoak, and B. N. Kokinov. *The analogical mind: Perspectives from cognitive science*. MIT press, 2001.

[69] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[70] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[71] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 2016.

[72] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *ICCV*, 2011.

[73] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013.

[74] P. Grice. Logic and conversation. In *Speech Acts*, 1975.

[75] A. Gupta, A. Kembhavi, and L. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. In *PAMI*, 2009.

[76] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920*, 3, 2017.

[77] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[78] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

[79] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *International Journal of Robotics Research*, 2012.

[80] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *SIGGRAPH*. ACM, 2001.

[81] J. Hill. Real time control of a robot with a mobile camera. In *9th Int. Symp. on Industrial Robots, 1979*, 1979.

[82] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 2012.

[83] B. Hixon, P. Clark, and H. Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. In *NAACL*, 2015.

[84] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[85] J. Hoffman, S. Guadarrama, E. S. Tzeng, R. Hu, J. Donahue, R. Girshick, T. Darrell, and K. Saenko. Lsda: Large scale detection through adaptation. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS*. 2014.

[86] B. Horn, B. Klaus, and P. Horn. *Robot vision*. MIT press, 1986.

[87] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, 2014.

[88] J. Huang et al. Exploring web scale language models for search query processing. In *WWW*, 2010.

[89] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 1996.

[90] S. J. Hwang, K. Grauman, and F. Sha. Analogy-preserving semantic embedding for visual object categorization. In *ICML*, 2013.

[91] H. Izadinia and P. Garrigues. Viser: Visual self-regularization. *arXiv preprint arXiv:1802.02568*, 2018.

[92] H. Izadinia, F. Sadeghi, and A. Farhadi. Incorporating scene context and object layout into appearance modeling. In *CVPR*, 2014.

[93] H. Izadinia, Q. Shan, and S. M. Seitz. Im2cad. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 2422–2431. IEEE, 2017.

[94] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[95] M. Jagersand, O. Fuentes, and R. Nelson. Experimental evaluation of uncalibrated visual servoing for precision manipulation. In *IEEE International Conference on Robotics and Automation*, 1997.

[96] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *European Conference on Artificial Life*, pages 704–720. Springer, 1995.

[97] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *arXiv preprint arXiv:1707.02267*, 2017.

[98] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[99] D. A. Jurgens, P. D. Turney, S. M. Mohammad, and K. J. Holyoak. Semeval-2012 task 2: Measuring degrees of relational similarity. ACL, 2012.

[100] A. Juthe. Argument by analogy. *Argumentation*, 2005.

[101] D. K. Kim and T. Chen. Deep neural network for real-time autonomous indoor navigation. *arXiv preprint arXiv:1511.04668*, 2015.

[102] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[103] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

[104] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality ACM International Symposium on*. IEEE, 2007.

[105] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 2012.

[106] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[107] J. Kua, N. Corso, and A. Zakhor. Automatic loop closure detection using multiple cameras for 3d indoor localization. In *IS&T/SPIE Electronic Imaging*, 2012.

[108] T. Lampe and M. Riedmiller. Acquiring visual servoing reaching and grasping skills using neural reinforcement learning. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8. Citeseer, 2013.

[109] T. Lan et al. From subcategories to visual composites: A multi-level framework for object detection. In *ICCV*, 2013.

[110] S. Lange, T. Gabel, and M. Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[111] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[112] A. X. Lee, S. Levine, and P. Abbeel. Learning visual servoing with deep features and fitted q-iteration. *arXiv preprint arXiv:1703.11000*, 2017.

[113] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

[114] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 2016.

[115] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[116] O. Levy and Y. Goldberg. Linguistic regularities in sparse and explicit word representations. In *CoNLL*. ACL, 2014.

[117] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*. Springer, 2014.

[118] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *CVPR*, 2015.

[119] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[120] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015.

[121] M. Malinowski and M. Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *NIPS*. 2014.

[122] M. Malinowski, M. Rohrbach, and M. Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *ICCV*, 2015.

[123] T. Malisiewicz and A. A. Efros. Beyond categories: The visual memex model for reasoning about object relationships. In *NIPS*, 2009.

[124] A. massoud Farahmand, A. Shademan, and M. Jagersand. Global visual-motor estimation for uncalibrated visual servoing. In *IROS*. IEEE, 2007.

[125] J.-B. Michel et al. Quantitative analysis of culture using millions of digitized books. In *Science*, 2010.

[126] J. Michels, A. Saxena, and A. Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *ICML*. ACM, 2005.

[127] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[128] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, 2013.

[129] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[130] 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 *ICML*, 2016.

[131] 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*, 2015.

[132] K. Mohta, V. Kumar, and K. Daniilidis. Vision-based control of a quadrotor for perching on lines. In *ICRA*. IEEE, 2014.

[133] K. Mohta, V. Kumar, and K. Daniilidis. Vision based control of a quadrotor for perching on planes and lines. In *ICRA*, 2014.

[134] M. Monajjemi. Bebop autonomy. `http://bebop-autonomy.readthedocs.io`.

[135] I. Mordatch, K. Lowrey, and E. Todorov. Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids. In *IROS*. IEEE, 2015.

[136] F. Muratore, F. Treede, M. Gienger, and J. Peters. Domain randomization for simulation-based policy optimization with transferability assessment. In *CoRL*, 2018.

[137] J. Oh, V. Chockalingam, S. Singh, and H. Lee. Control of memory, active perception, and action in minecraft. *arXiv preprint arXiv:1605.09128*, 2016.

[138] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *ICCV*, 2011.

[139] D. Parikh and K. Grauman. Relative attributes. In *ICCV*, 2011.

[140] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *ICLR*, 2018.

[141] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[142] 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.

[143] 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.

[144] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *ICRA*. IEEE, 2016.

[145] R. Pissard-Gibollet and P. Rives. Applying visual servoing techniques to control a mobile hand-eye system. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*. IEEE, 1995.

[146] D. A. Pomerleau. Alvinn, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, Computer Science Department, 1989.

[147] A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. In *ICRA*, 2015.

[148] M. L. Puterman and M. C. Shin. Modified policy iteration algorithms for discounted markov decision problems. *Management Science*, 1978.

[149] C. Richter and N. Roy. Safe visual navigation via deep learning and novelty detection. 2017.

[150] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. *arXiv preprint arXiv:1608.02192*, 2016.

[151] M. Riedmiller. Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning (ECML)*, 2005.

[152] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Van de Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

[153] A. Ritter, Mausam, and O. Etzioni. A latent dirichlet allocation method for selectional preferences. In *ACL*, 2010.

[154] S. Ross, G. Gordon, and A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research*, 15:627–635, 2011.

[155] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert. Learning monocular reactive uav control in cluttered natural environments. In *ICRA*. IEEE, 2013.

[156] R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1999.

[157] O. Russakovsky et al. Detecting avocados to zucchinis: what have we done, and where are we going? In *ICCV*, 2013.

[158] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

[159] F. Sadeghi. Teaching uncalibrated robots to visually self-adapt. `https://ai.googleblog.com/2018/06/teaching-uncalibrated-robots-to_22.html`, 2018.

[160] F. Sadeghi. DIViS: Domain invariant visual servoing for collision-free goal reaching. *arXiv preprint arXiv:1902.05947*, 2019.

[161] F. Sadeghi, S. Kumar Divvala, and A. Farhadi. VisKE: Visual Knowledge Extraction and Question Answering by Visual Verification of Relation Phrases. In *CVPR*, 2015.

[162] F. Sadeghi and S. Levine. CAD2RL: Real single-image flight without a single real image. In *Robotics: Science and Systems(RSS)*, 2017.

[163] F. Sadeghi and M. F. Tappen. Latent pyramidal regions for recognizing scenes. In *ECCV*, 2012.

[164] F. Sadeghi, A. Toshev, E. Jang, and S. Levine. Sim2real viewpoint invariant visual servoing by recurrent control. In *CVPR*, 2018.

[165] F. Sadeghi, C. L. Zitnick, and A. Farhadi. Visalogy: Answering visual analogy questions. In *Advances in Neural Information Processing Systems*, 2015.

[166] M. Sadeghi and A. Farhadi. Recognition using visual phrases. In *CVPR*, 2011.

[167] A. Sanderson. Image based visual servo control using relational graph error signal. In *Proc. Int. Conf. Cybernetics and Society, Cambridge, 1980*, 1980.

[168] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931*, 2017.

[169] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *NIPS*, 2005.

[170] K. Schmid, T. Tomic, F. Ruess, H. Hirschmller, and M. Suppa. Stereo vision based indoor/outdoor navigation for flying robots. In *IROS*, 2013.

[171] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *CoRR, abs/1502.05477*, 2015.

[172] M. J. Seo, H. Hajishirzi, A. Farhadi, and O. Etzioni. Diagram understanding in geometry questions. In *AAAI*, 2014.

[173] C. Shelley. *Multiple analogies in science and philosophy*. John Benjamins Publishing, 2003.

[174] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Vision-based state estimation for autonomous rotorcraft mavs in complex environments. In *ICRA*, 2013.

[175] Y. Shirai and H. Inoue. Guiding a robot by visual feedback in assembling tasks. *Pattern recognition*, 5(2):99–106, 1973.

[176] B. Siciliano and O. Khatib. *Springer handbook of robotics*. Springer Science & Business Media, 2008.

[177] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[178] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[179] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 190–198. IEEE, 2017.

[180] R. Speer and C. Havasi. ConceptNet 5: A large semantic network for relational knowledge. In *http://conceptnet5.media.mit.edu*, 2013.

[181] G. J. Stein and N. Roy. Genesis-rt: Generating synthetic images for training secondary real-world tasks. In *ICRA*, 2018.

[182] E. Sudderth, A. Torralba, W. T. Freeman, and A. Wilsky. Learning hierarchical models of scenes, objects, and parts. 2005.

[183] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[184] P. Talukdar, D. T. Wijaya, and T. M. Mitchell. Acquiring temporal constraints between relations. In *CIKM*, 2012.

[185] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 2009.

[186] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural computation*, 2000.

[187] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[188] J. Tobin, W. Zaremba, and P. Abbeel. Domain randomization and generative models for robotic grasping. *arXiv preprint arXiv:1710.06425*, 2017.

[189] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[190] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *arXiv preprint arXiv:1804.06516*, 2018.

[191] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustmenta modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[192] P. D. Turney. Similarity of semantic relations. *Comput. Linguist.*, 2006.

[193] P. D. Turney and M. L. Littman. Corpus-based learning of analogies and semantic relations. *CoRR*, 2005.

[194] P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 2010.

[195] E. Tzeng, C. Devin, J. Hoffman, C. Finn, P. Abbeel, S. Levine, K. Saenko, and T. Darrell. Adapting deep visuomotor representations with weak pairwise constraints. *CoRR, vol. abs/1511.07111*, 2015.

[196] E. Tzeng, C. Devin, J. Hoffman, C. Finn, X. Peng, S. Levine, K. Saenko, and T. Darrell. Towards adapting deep visuomotor representations from simulated to real environments. *arXiv preprint arXiv:1511.07111*, 2015.

[197] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015.

[198] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *SIGCHI*, 2004.

[199] L. E. Weiss, A. C. Sanderson, and C. P. Neuman. Dynamic visual servo control of robots: an adaptive image-based approach. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 662–668. IEEE, 1985.

[200] W. M. Wichman. Use of optical feedback in the computer control of an arm. Technical report, STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE, 1967.

[201] W. J. Wilson, C. W. Hulls, and G. S. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 1996.

[202] W. J. Wilson, C. W. W. Hulls, and G. S. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5), 1996.

[203] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018.

[204] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[205] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[206] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010.

[207] B. H. Yoshimi and P. K. Allen. Active, uncalibrated visual servoing. In *IEEE International Conference on Robotics and Automation*, 1994.

[208] B. H. Yoshimi and P. K. Allen. Active, uncalibrated visual servoing. In *ICRA*, 1994.

[209] L. Yu, E. Park, A. C. Berg, and T. L. Berg. Visual madlibs: Fill in the blank description generation and question answering. In *ICCV*, 2015.

[210] W. Yu, C. K. Liu, and G. Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

[211] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard. Deep reinforcement learning with successor features for navigation across similar environments. *arXiv preprint arXiv:1612.05533*, 2016.

[212] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard. Deep reinforcement learning with successor features for navigation across similar environments. *arXiv preprint arXiv:1612.05533*, 2016.

[213] J. Zhang, L. Tai, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard. Vr goggles for robots: Real-to-sim domain adaptation for visual control. *arXiv preprint arXiv:1802.00265*, 2018.

[214] M. Zhang, Z. McCarthy, C. Finn, S. Levine, and P. Abbeel. Learning deep neural network policies with continuous memory states. In *ICRA*. IEEE, 2016.

[215] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 2012.

[216] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

[217] Y. Zhu, A. Fathi, and L. Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *ECCV*, 2014.

[218] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *arXiv preprint arXiv:1609.05143*, 2016.

[219] C. Zitnick, R. Vedantam, and D. Parikh. Adopting abstract images for semantic scene understanding. *PAMI*, 2014.

[220] C. L. Zitnick and D. Parikh. Bringing semantics into focus using visual abstraction. In *CVPR*, 2013.