Learning Multi-Modal Navigation for Unmanned Ground Vehicles

Arne J. Suppé
March 30, 2018

Proposal Committee:
Martial Hebert, Chair
Kris Kitani
Jean Oh
Junsong Yuan, State University of New York at Buffalo

Copyright © 2018 Arne J. Suppé. All rights reserved.
Abstract

A robot that operates efficiently in a team with a human in an unstructured outdoor environment must be able to translate commands from a modality that is intuitive to its operator into actions. This capability is especially important as robots become ubiquitous and interact with untrained users. For this to happen, the robot must be able to perceive the world as humans do, so that the nuances of natural language and human perception are appropriately reflected in the actions taken by the robot. Traditionally, this has been done with separate perception, language processing, and planning blocks unified by a grounding system. The grounding system relates abstract symbols in the command to concrete representations in perception that can be placed into a metric or topological map upon which one can execute a planner. These modules are trained separately, often with different performance specifications, and are connected with restrictive interfaces to ease development and debugging (i.e., point objects with discrete attributes), but which also limit the kinds of information one module can transfer to another.

The tremendous success of deep learning has revolutionized traditional lines of research in computer vision, such as object detection and scene labeling. The latest work goes even further, bringing together state of the art techniques in natural language processing with image understanding in what is called visual question answering, or VQA. Symbol grounding, multi-step reasoning, and comprehension of spatial relations are already elements of these systems, all contained in a single differentiable deep learning architecture, eliminating the need for well-defined interfaces between modules and the simplifying assumptions that go with them.

Building upon this work, we introduce a technique to transform a natural language command and a static aerial image, into a cost map suitable for planning. With this technique, we take a step towards unifying language, perception, and planning in a single, end-to-end trainable system. Further, we propose a synthetic benchmark based upon the CLEVR dataset, which can be used to compare the strengths weakness of the comprehension abilities of various planning algorithms in the context of an unbiased environment with virtually unlimited data. Finally, we propose some extensions to the system as steps towards practical robotics applications.
## Contents

1 Introduction .............................. 1
   1.1 Basic Problem Statement .............................. 2
   1.2 Anticipated Contributions .............................. 3
   1.3 Proposal Organization .............................. 4
   1.4 Proposed Timeline .............................. 6

2 Commanding Robots with Language .............................. 7
   2.1 Prior Work ........................................ 7
   2.2 RCTA Architecture ..................................... 9
   2.3 Improving existing architectures .............................. 11
   2.4 Conclusion ....................................... 12

3 Attribute Sensitive Cost Maps ...................................... 14

4 Navigating with Natural Language Commands and Static Aerial Imagery ...................................... 15
   4.1 Background ......................................... 16
      4.1.1 Basic VQA Architectures .............................. 16
      4.1.2 Attention Models ................................... 18
   4.2 Adapting a VQA System for Navigation .............................. 22
      4.2.1 Modifying Visual7W .................................. 24
      4.2.2 Modifying SAN .................................... 26
      4.2.3 Improving Map Resolution .............................. 26
      4.2.4 Training ........................................ 27
      4.2.5 Modifications to the CLEVR Dataset .............................. 27
   4.3 Procedural Attention Models .............................. 28
   4.4 Experiments ........................................ 28
   4.5 Conclusion ....................................... 29
   4.6 Timeline ........................................ 29

5 Navigating in Uncertain and Partially Observable Environments ...................................... 31
   5.0.1 Integrating Perspective Images, Memory and Planning .............................. 33
   5.0.2 Other Goals ....................................... 34
   5.1 Proposed Simulation Experiments .............................. 35
      5.1.1 Task 3A: Navigation with Mapping and Exploration .............................. 35
      5.1.2 Navigation with Perspective Images .............................. 37
   5.2 Conclusion ....................................... 37
   5.3 Timeline ........................................ 38

Bibliography ...................................... 39
List of Figures

1.1 Simplified system schematic ................................................. 3
1.2 Timeline for proposed work ................................................ 6
2.1 Technical details for Husky robots used in our experiments .......... 9
2.2 Example of perception data flow in RCTA system ....................... 10
2.3 Tactical Behavior Specification (TBS) .................................... 11
4.1 Basic VQA architecture adapted from [55] ............................... 17
4.2 VQA architecture with LSTM text encoding adapted from [1] ...... 17
4.3 Simplified diagram of the Visual7W VQA network adapted from Zhu et al. [57] .... 19
4.4 Simplified diagram of a Stacked Attention Network adapted from Yang et al. [53] ... 21
4.5 Example image, question, and partial scene graph from CLEVR dataset [20]. .... 22
4.6 Simplified diagram of the Visual7W VQA network modified for navigation with cost map encoding mechanism 1 .................................. 24
4.7 Simplified diagram of the Visual7W VQA network modified for navigation with cost map encoding mechanism 2 .................................. 25
4.8 Simplified diagram of a Stacked Attention Network modified for navigation .......... 27
4.9 Timeline for task 2 .............................................................. 30
5.1 Simulation environments for task 3 experiments ......................... 36
5.2 Timeline for task 3 .............................................................. 38
List of Tables

4.1 State-of-the-art VQA reference implementations considered in this work. Accuracy as measured against CLEVR. ............................................................... 23
Chapter 1

Introduction

A robot that operates seamlessly in tandem with a human must be able to translate text or verbal commands into the expected actions. In scenarios as disparate as disaster recovery, combat, and elder care, it can not be assumed that the operator has the time or the training to generate detailed instructions using specialized interfaces. In short, the ideal robot would follow instructions as well as any other human teammate and make appropriate assumptions to clarify ambiguous commands or perception information.

This is a challenging problem that lays at the intersection of perception, natural language processing, and artificial intelligence. In this thesis proposal, we plan to address just one example of this much more significant problem; how to traverse an aerial image, given a starting location and a spoken or written command. Even in this restricted context, we face the same difficulty that makes this an area worthy of continuing research; that both language and perception are often ambiguous, with a large amount of implicit knowledge underlying our everyday communications with each other.

Almost as soon as the first mobile robots were made, researchers have been looking towards natural languages as an ideal way to command them. Shakey the Robot (Nilsson [33]), used a command language that was directly translated into a first-order predicate calculus statement. It operated in an office environment specially constructed to make the perception problem as simple as possible. Even Shakey’s world was distilled into an array of predicate calculus statements. A plan was a sequence of elementary actions that manipulated the world model into the desired state. In Shakey’s case, perception was assumed to directly yield discrete symbols that are perfect or near perfect, allowing the use of classical artificial intelligence techniques such as logical inference. Indeed, perception was explicitly not a design objective of this system to simplify an already complex problem.

Unlike SONAR or LADAR, which directly measure the presence of an obstacle, vision is uniquely difficult to model, perhaps because we expect it to generate far more abstract output than it usually does. For example, SONAR or LADAR measurement properties are modeled in algorithms such as occupancy grids (Moravec [31]), particle filter localizers, (Thrun [48]) and SLAM systems
CHAPTER 1. INTRODUCTION

(Thrun et al. [49]). These systems have been successfully integrated into the intelligence/navigation architectures of robots that operate in many different environments, like CARMEL (Congdon et al. [6]), Boss (Urmson et al. [50]) and others. While vision has been used for similar purposes, it is often at the level of landmark detection and triangulation, and not scene parsing. The problem remains of how to combine higher-level vision into a robot navigation system in a way that both renders meaningful, abstract symbols and is also mathematically tractable.

Our own recent experience with outdoor field robotics (Oh et al. [34]) points to the continuing difficulty in fully integrating vision with intelligence tasks. At higher levels of reasoning, vision can be used, for example, as an object detector (Zhu et al. [56]) or semantic scene labeler (Munoz [32]). However, simplifications are necessary to condense complex visual information into a model suitable for abstract reasoning. These models may be carefully engineered for a particular task (such as the exemplar 3D models in Zhu et al. [56]), or may be extremely general, such as a bounding box or cylinder used to define vehicles and humans. As a model can only convey the data its designers deem appropriate, it is likely that essential but non-obvious information is lost. Was the car occluded? Was it located in an unusual place or parked at a strange angle? It is clear that it is not possible to maintain a set of models for all objects of interest in most human environments.

Computer vision has recently seen incredible development. The deformable parts model (DPM) (Felzenswabl et al. [9]), the leading object detector only a few years ago, has been overtaken by deep learning approaches that require large amounts of data but also generalize better. Further, many deep learning techniques are shared between state of the art image understanding and natural language processing architectures (e.g., LSTM, RNN), making a fusion of these domains into a single deep learning framework quite natural. This has already occurred with image captioning (Chen et al. [5] and others) and visual question answering (Johnson et al. [21] and others). Moreover, the most recent examples all have some form of symbol grounding, the core problem of relating the meaning of the text to pixels in an image, which is also essential to the robot navigation problem in which we are interested.

Therefore we choose to approach the problem of integrating perception, natural language processing, and planning from the point of view of deep learning. Different from more current lines of research, such as Oh et al. [34], the grounding mechanism is not explicitly modeled. Instead, we propose to learn the desired trajectory directly, obviating the need for interfaces between modules, and allowing for the simultaneous training of language, perception, and planning elements against a single loss function. This single loss function is the same loss metric against which the user would ultimately measure the robot’s performance. Further, we propose to build this system from existing image captioning and visual question answering architectures.

Basic Problem Statement

We are given a set of aerial orthographic image $I_i$, for $i \in 1...N$ a natural language command $Λ_{ij}$, for $j \in 1...M$, a set of starting points $X_{ij}$, and a set trajectories $µ_{ijk}, k \in 1...L$ that satisfy the commands, as deemed by $L$ experts. (Figure 1.1) Having learned from the labeled data, our goal is
\( \mathcal{I}_i = \) "Plan a covert path to the front of the building to the south."

\( \Lambda_j = \) "Plan a covert path to the front of the building to the south."

Deep Network Cost Map
\[ F(\mathcal{I}_i, \Lambda_j) \]
Path Planner Prediction
\[ \mu_{ijk} \]
Expert Path
\[ \mu^*_{ijk} \]
\[ \frac{\partial L}{\partial F} \]

Figure 1.1: Simplified system schematic

to interpret a new command \( \Lambda' \), image \( \mathcal{I}' \), and starting location \( X' \), and generate a trajectory \( \mu' \) that conforms to what an expert would have drawn.

We simplify the enormous problem of robot planning and perception to a more constrained problem that interests us; how to unify command parsing, scene understanding, and path planning into a single, fully differentiable deep learning process. We assume the world is fully observable and static.

Even with these simplifications, we retain many exciting problems to study. For example, a mountain range in the distance may affect the local cost of heading west versus heading north. Will the deep network be able to determine how particular local image features far from the starting point affect global path decisions? What kind of directional concepts can the architecture understand? Can it make appropriate assumptions given ambiguous commands that a human could follow?

Later, we will expand this definition to include a ground vehicle’s perspective. This simple change introduces a host of new and interesting problems because the world is no longer fully observable, even if it remains static. The navigation system must learn to build a map and explore the world to find the feature relevant to the given command.

**Anticipated Contributions**

At the conclusion of this thesis project, we foresee the following contributions:

- An architecture for unifying natural language processing, image understanding, symbol grounding, and path planning into a single deep learning framework with a single loss metric.
- An evaluation of several modified state-of-the-art VQA systems, with analysis of their strengths and weaknesses in this new context, along with concepts on how to make such systems more suitable for our problem.
- Extensions to these architectures to handle images from a ground robot’s perspective, map building, and partial grounding in ambiguous situations.
CHAPTER 1. INTRODUCTION

• A freely available synthetic dataset for testing the reasoning abilities of this and other similar systems in an unbiased environment with virtually unlimited data, based on the CLEVR dataset (Johnson et al. [20]), but re-targeted for our problem space.

Proposal Organization

In Chapter 2, we introduce prior work on how spoken or written commands and perception are integrated to generate plans for field robots. We examine their strengths and weaknesses to motivate the fundamental contribution of this thesis; that we can build useful robotic planning systems by unifying state of the art techniques in natural language processing and image understanding with path planning into a single framework.

We do this without having to invent radical new concepts in machine learning or computer vision, but instead, we observe that Visual Question Answering systems are now addressing many of the same problems that text-perception-planning systems have had to solve in the past, in particular, grounding.

In Chapter 3 we introduce some fundamental techniques for combining deep networks with path planning using Inverse Optimal Control (IOC) to generate attribute sensitive cost maps; cost maps that are learned from aerial images annotated with expert path demonstrations, which implicitly capture the features necessary for planning a path, given simple attributes such as “directly” or “covertly.” Some of these concepts are easy to code, while others, such as “covertly” require a more subtle understanding that can not be captured so easily, for which deep learning is useful. While the IOC learning techniques are not new, they are not commonly used in the deep learning community for image processing tasks. The purpose of this initial work is to demonstrate their suitability for following tasks. We offer demonstrations on synthetic image data and real aerial images.

In Chapter 4 we propose to apply the techniques to existing, state of the art Visual Question Answering (VQA) architectures to demonstrate a complete system that takes text commands and a static image and produces a cost map from which we can generate a path. As demonstrated by results in Hu et al. [19] and Lu et al. [26] among many others, different VQA systems show different strengths and weaknesses based on the type of question posed. This is often due to limitations in the kinds of grounding strategies the underlying model can express. We expect similar variation in the planning ability of these systems when applied to our problem. We propose to construct a diagnostic data set, based on CLEVR (Johnson et al. [20]), that highlights the strengths and weakness of any such system in a controlled, unbiased environment, and to develop strategies to address these weaknesses. Finally, we propose to collect expertly labeled satellite imagery to demonstrate this concept on real data.

In Chapter 5 we propose to extend one of the navigation systems we develop in Chapter 4 to make it more practical for ground vehicles. Initially, we only consider a world that is fully observable, but in the real world and from a ground vehicle’s perspective, objects may lay behind an obstruction or beyond the horizon. This means that the referents in a command may not all be visible, and the command is therefore ambiguous. To plan in this, more realistic, model of the world requires
reasoning in a topological space, not just a metric one, with grounding that may not be directly tied to perception. Finally, we also seek to address problems where the robot must build a map over time, integrating new knowledge as it travels, updating its plan accordingly as it efficiently explores the world to disambiguate a command.
## Proposed Timeline

<table>
<thead>
<tr>
<th>Task Group 1: Attribute sensitive cost maps</th>
<th>2018</th>
<th>2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improve Max Margin Solution</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Test w/ synthetic/existing data</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Code MaxEnt IRL Solution</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Label and test w/ aerial images</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Paper writing</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Submission to NIPS 2018</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task Group 2: Navigating w structured commands</th>
<th>2018</th>
<th>2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formulation of several architectures</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Generate modified CLEVR dataset</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Test w/ modified SAN</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Test w/ modified Visual7W</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Test w/ modified IEP</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Test w/ modified N2NM</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Label and test all w/ aerial images</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Paper writing</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Submission to AAAI 2019</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 3A: Navigation w/ mapping and exploration</th>
<th>2018</th>
<th>2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple loop navigation and strategic exploration</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Integrated memory</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Experiments with CLEVR-based simulator</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Paper writing</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Submission to IJCAI 2019</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 3B: Navigation with Perspective Images</th>
<th>2018</th>
<th>2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code development</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Testing with AirSim</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Paper writing</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Submission to IROS 2019</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Thesis writing</th>
<th>2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thesis Defense</td>
<td>☐</td>
</tr>
</tbody>
</table>
Chapter 2

Commanding Robots with Language

In this chapter, we present a brief summary of recent work in robotics where language and perception are combined to plan robot actions. Second, we present our practical experience with a robot that was tested in an outdoor environment. We do this to understand what kinds of problems these systems solve, and to understand what opportunities exist for improvement.

Prior Work

Symbol grounding is the process of relating symbols in the form of words to concrete representations in perception data first formulated in Harnad [16]. This process is essential for planning robot trajectories with language commands. Shakey (Nilsson [33]) did this by directly translating the instruction into the same predicate calculus statements that comprised the world model. Perception was vastly simplified to minimize clutter and false detections by engineering the environment in which the robot operated. These discrete and unambiguous object detections were translated into yet more predicate calculus symbols in the world model. Therefore, there was no ambiguity in understanding the meaning of the command or the state of the world.

The real world is not nearly so ideal. Commands may be ambiguous because of information they leave out or even because one person’s understanding of a phrase may differ from another. (Oh et al. [34]) Objects may be in a cluttered environment, partially obstructed, have multiple or false detections, or may have an appearance that does not exactly match the exemplar object. For example, what one person may call a bench another may call a chair.

One method for dealing with ambiguity and error is to develop appropriate fallback and error recovery routines. In MacMahon et al. [27], the authors assume that only the instruction may be ambiguous, and not perception. The artificial world in which they run human and automated trials is a maze of corridors with sparse features and a very limited set of actions at each intersection.
CHAPTER 2. COMMANDING ROBOTS WITH LANGUAGE

The core contribution is a method by which a natural language instruction is parsed into a syntax tree, and then translated into a robust set of commands for the robot to follow, using hard-coded knowledge to fill in information that is implicit to the natural language instruction.

This reduction of the world to a maze of corridors with discrete decision points is common in literature and is appropriate for office-like environments, but does not work well in open space environments where one may arrive at the same goal with many solutions. Further, this method is not robust to sensor noise, which is more much common in natural environments.

Kollar et al. [22] is an instructive example of how probabilistic reasoning can be used to solve the grounding problem. We expand upon this example because more easily summarized compared to more advanced systems. However, these more advanced systems still must codify relational concepts from language (such as “left of” or “right of”) as probabilistic equations.

In Kollar et al. [22], the authors show that it is possible to translate more than 90 percent of human-generated instructions used to describe paths in a real office environment using a formalism called spatial description clauses (SDC), allowing for natural language communication, with the exception of occasional parsing errors. Each command is parsed into SDCs, and each SDC includes an object, a verb, a landmark, and a spatial relationship. By translating the command into discrete units, they effectively translate a natural language command into a sequence of grounded instructions that the robot can follow.

Given path \( P \), a sequence of SDCs \( S \), and detected objects \( O \), the path which best satisfies the sequence of relations specified in the SDCs is:

\[
P^* = \arg \max_P p(P, S|O) = \arg \max_p p(S|P,O)p(P|O)
\]

(2.1)

Already, it is apparent that perception is not modeled since the objects, \( O \), are given. Second, the algorithm only returns the most probable path. The next simplification is to decompose the SDCs and the path. In particular, decomposing the path into discrete viewpoints \( v_i \) implies that the algorithm operates over a topological map, with each SDC causing a change in viewpoint.

\[
p(P, S|O) \approx p(sdc_1, \ldots, sdc_M|v_1, \ldots, v_{M+1}, O)p(v_{M+1}, O)\]

\[
\approx \prod_{i=1}^{M} p(sdc_i|v_i, v_{i+1}, O) \times \prod_{i=1}^{M} p(v_{i+1}|v_i, \ldots, v_1) \quad p(v_1)
\]

(2.2)

(2.3)

At this point, we see how the SDC relates to the model. The SDC is expanded into the object \( f \), verb \( a \), landmark \( l \), and spatial relationship \( s \) (Equation (2.5)).

\[
p(sdc_i|v_i, v_{i+1}, O) = p(f_i, a_i, s_i, l_i|v_i, v_{i+1}, O)
\]

\[
\approx p(f_i|v_i, v_{i+1}, O_1 \ldots O_K) p(a_i|v_i, v_{i+1}) p(s_i|l_i, v_i, v_{i+1}, O_1 \ldots O_K) p(l_i|v_i, v_{i+1}, O_1 \ldots O_K)
\]

(2.4)

(2.5)
First, let us consider the action term in Equation (2.5). This is a hard-coded model relates terms such as “left” or “straight” to the physical motion required to move from $v_i$ to $v_{i+1}$. Next, consider the landmark and object terms. These terms are computed using image co-occurrence statistics gathered from Flickr. That is, given the objects $o_k$ that are visible at $v_i$, $p(l_i|...)$ is the most likely pairwise co-occurrence between $o_k$ and $l_i$. Finally, consider the spatial relationship term, which encodes phrases like “through” or “to the left of.” This is learned from human example.

In brief, perception is not modeled in any meaningful way. One of the reasons for this is that computer vision is difficult to model as anything other than pixels or discrete objects. It is hard to envision a model for something intermediate. What does it represent? Perhaps a deformable parts model (DPM, Felzenszwalb et al. [9]) would be appropriate to model appearance for some classes of objects, but the world is complex and cluttered, and it seems unlikely that a model can be constructed for the many different kinds of objects a robot is likely to encounter in a natural environment.

RCTA Architecture

We have direct experience with a field robot that used a text based command language and perception to plan paths without any a priori maps. The author has been involved with the following papers: Oh et al. [34], Oh et al. [35], Oh et al. [37], Suppé et al. [46], Oh et al. [36]. Further, the complete navigation system has been independently evaluated in Lennon et al. [23]. The robot operates in an outdoor environment and so simplifications, such as using a purely topological map, are not very appropriate.

The robot platform is based on a Clearpath™ Husky (Figure 2.1) equipped with a high dynamic range camera with a 120 degree horizontal field of view and a scanning LADAR with a 360 degree field of view and an 80 meter range. For the purposes of our experiment, the LADAR was used to detect building and walls and to position labeled image pixels in 3D.
CHAPTER 2. COMMANDING ROBOTS WITH LANGUAGE

Our image classifier was the Hierarchical Inference Machine (HIM, Munoz [32]). This pixel labeler segments the image into superpixels, and then labels the superpixels using a forest of decision trees to classify the contents of each superpixel, based on features such as dense SIFT (Lowe [25]), Local Binary Patterns (LBP, Ojala et al. [38]), and texture (Shotton et al. [44]). Neighboring superpixels are grouped into a hierarchical structures with regional features so that global as well as local information is used to produce the final labeling. The image classifier worked in parallel with an object detector (Zhu et al. [56]) and a person detector (Yang and Ramanan [52]).

Figure 2.3 shows the construction of the command language called the tactical behavior specification (TBS) used in our experiments. It is a structured language expressible in Backus-Naur form. The named objects map directly to classifier and detector categories, so there is no ambiguity in their meaning. The TBS encodes a number of spatial relationships, such as “left of” or “behind” as well as a two modal constraints: “covertly” and “quickly.” This affects the kind of path the robot will take.

The grounding process consists of two tasks. First, one must map nouns in the TBS to object detections in the world. Second, one must understand the meaning of phrases such as “behind.” The former is accomplished with a grounding graph, detailed in Duvallet [8] and the later by using Max Margin Planning (Ratliff et al. [40]) to learn cost map associated with each term, detailed in [ ] and Boularias et al. [4].

Our field experiments are detailed in Oh et al. [35]. The test environment was a simulated town situated on a military base that is designed for training soldiers. Of 30 TBS commands executed over 46 runs, 35 were considered successful. However, the interface between perception and planning was a common problem during the test. For example, misperception on the part of the semantic classifier often populated the robot’s map with phantom objects, causing the robot to erroneously ground the command against objects that did not exist. We believe that a more nuanced model of the detected objects may be able to mitigate these problems by also including information that would be useful to judge the quality of the detection. What does that model look like and do we
CHAPTER 2. COMMANDING ROBOTS WITH LANGUAGE

Figure 2.3: Tactical Behavior Specification (TBS). Figure adapted from Boularias et al. [3]

need to design a new one for each object in the world?

Improving existing architectures

Based on our experience and our survey of other state of the art command systems of a similar nature, we can make a few observations about how we can contribute to the field.

• **End-to-end learning:** The perception system should be trained with a loss function that is relevant to the tasks expressible in the command language.

• **Data-defined interfaces:** The interface between perception, language processing, and planning can be more flexible and expressive.

• **Task-specific planning/perception:** The perception/planning process is complex. Depending on the complexity of the command, this may even require an iterative solution with a perception attention mechanism similar to what we see in modern VQA systems.

In the RCTA architecture and many others, information flows in one direction; from perception to grounding/intelligence processes. The perception process is essentially autonomous from the rest of the robot, which makes development and testing much easier, but also presents a few problems.

First, the perception process is very likely not trained with a loss function that is relevant to the task set. For example, the HIM semantic classifier in Munoz [32] is designed to be competitive with the Stanford Background Dataset (Gould et al. [11]) and the MSRC-21 dataset (Shotton et al. [44]). Performance is measured in terms of the number of correctly labeled pixels in total, or averaged across classes. For a ground robot, each pixel in a scene will not have the same importance due to perspective effects. A pixel far away may be small, but may also be extremely relevant to the task. Worse, large regions, such as sky, are completely irrelevant to a ground robot, and regions like grass
are not as critical to the overall correctness of a task as specific objects, yet both have the same weight as any other class.

It is clear that it is important to match the performance of the classifier to the robot task, but its not obvious how to do that, in part, because it depends on the distribution of tasks the robot is commanded to perform and how sensitive each task is to errors. For example, a mislabeled grass region is not as important as a mislabeled goal, e.g., a car. Second, one may have multiple detection systems, each with their own characteristics. In our case, we had an additional object detector that was trained with rather different loss function.

The interfaces between one module and the next are similarly difficult to define. In our case, labeled pixels were mapped to 3D points with a calibrated LADAR. These point clouds were then condensed into simplified objects in the robot’s local map. These models may not be appropriate for all kinds of objects and tasks and each model must be engineered by hand. For example, color may be important for one object in a particular task, and relative orientation may be important for another object and task.

Finally, the task itself has no effect on how the perception system works. All regions of the image have equal importance, even if the region is irrelevant to the command in question. This wastes computational power, but is also invites unnecessary errors. For example, a system planning a path though an office environment may choose to look for chairs in specific regions of an image and can safely dispense with looking for vehicles, unless presented with extraordinary data. Like a human, the algorithm should have some sort of expectation as to what to find in an environment, and what things are important to look for, given a task.

One solution to the first two problems is to train the whole system with a single loss function. For a neural network based classifier, this means the system must be fully differentiable. Fortunately, language parsing and image understanding are well developed with suitable candidate algorithms already fused into a single system, thereby obviating the need for a special interface between the two. We discuss candidate systems in Chapter 4.

We need only find a differentiable path planning loss function. We introduce two candidate loss functions in Chapter 3.

Conclusion

The task of weaving together the many disparate modules that compose a robotic system is not an easy one. The choice of architecture helps to compartmentalize both design work and complexity and has been essential to progress since the earliest days of robotics. However, with ever increasing computational and data resources, we can fuse tasks together that were previously separate, eliminating then need for discrete interfaces.

As an example of this rapid progress towards data-driven unitary architectures, consider the Discrete Parts Model (DPM, Felzenszwalb et al. [9]), which was a state of the art object detector only a few years ago. The DPM contained a flexible and abstract framework for modeling how different regions in an object move relative to each other as the object deforms or changes in position.
and orientation. This model-based approach is no longer state of the art, having been replaced in a few short years by many different neural network architectures which are essentially model-free; they have no explicit model for object deformation save for what is contained in the training data.

A similar progression occurred for earlier concepts in vision, such as Marr [29], which used a hierarchy of primitives such as edges, corners, and planar regions, grouped into increasingly complex and meaningful components to compose objects. In general, this approach is not tractable since the multiplicity of object types, view points, clutter, and appearance make this problem far too difficult to model in the real world. Instead, DPM uses HOG (Dalal and Triggs [7]), a higher level local feature descriptor.

We feel that this next step of integrating path planning, vision, and language processing into a single deep learning framework is in keeping with current trends in computer vision research. However, just because the field is moving in this direction does not mean the path forward is also without obstacles, making this a worthwhile research problem.
Chapter 3

Attribute Sensitive Cost Maps

This section under revision
In this chapter, we discuss how to develop a neural network from one that can plan attribute sensitive paths through aerial images, to one that can accept more complex commands. We limit ourselves to a structured language even though some of the techniques we introduce have the capability for interpreting free-form language. We do this for three reasons. Structured language can express complex actions with precision and succinctness that is more than sufficient to test our notional system. For some architectures, this simplifies the command encoding process to something much less opaque, which may lead to more predictable behavior; a desirable characteristic for a planning system. Since the language is structured, it is easier to generate synthetic datasets. Finally, some prior work that will be used as a point of comparison is dependent on structured language. This task will lay the groundwork for more generalized navigation system. First, we will motivate the argument for using state of the art Visual Question Answering (VQA) systems as the foundation for our planning system. Next, we introduce, in detail, several architectures derived from these systems which we intend to test, and finally, we discuss the experiments we wish to perform on these systems. Our objectives in this work are:

- Build a planning system using the loss function developed in Chapter 3 with a command system that accepts complex statements.
- Determine strengths and weaknesses for various notional architectures in the context of the navigation task, by using synthetic data and human-labeled aerial images.
- Test two different mechanisms for command encoding in those architectures that support it.
CHAPTER 4. NAVIGATING WITH NATURAL LANGUAGE COMMANDS AND STATIC AERIAL IMAGERY

Background

Visual Question Answering is the problem of taking an image and a free-form open-ended question and generating an answer to that question, usually in the form a single word. By itself, this problem is extremely broad and poorly defined, and has been likened to the problem of solving artificial intelligence, in general. We review the development of state of the art VQA systems so that we can make an informed decision about which architectures have the potential to effectively reason about the navigational concepts. In Agrawal et al. [1], the authors introduce a benchmark dataset designed to test artificial intelligence algorithms with a broad set of images and human-engineered questions that are designed to challenge the intellect of a hypothetical toddler, alien, or smart robot. That is, the questions were designed to be non-obvious, requiring background knowledge and reasoning skills that current state-of-the art systems may not possess. COCO (Lin et al. [24]) or Visual7W (Zhu et al. [57]) and DAQUAR ([28]) are also commonly used in this field, and like the VQA dataset, they ask generic questions, usually from a human’s perspective of the world. That is, they contain images of human environments and the objects commonly found in them. While it is obviously important for a hypothetical robot to navigate in a human environment, our work is more concerned with establishing the capability of a VQA-derived system to understand concepts related to navigation. Primarily, this means identifying landmarks and their orientations in relation to a hypothetical robot. These concepts are not generally tested by the questions and environments found in the aforementioned datasets. For this, and other reasons discussed later, we instead turn to the CLEVR dataset (Johnson et al. [20]), which is designed to stress spatial and abstract reasoning in VQA systems. Next, we will review some of the fundamentals about how VQA architectures work as motivation for how we intend to modify them to suit our own design objectives. In particular, we will look at architectures that have a mechanism for attention, which is the ability to locate areas of interest in an image as directed by a given query, which we will later use for navigational purposes.

Basic VQA Architectures

Figure 4.1 is an example of a typical early system for solving this problem using a bag-of-words question representation (Zhou et al. [55]). Image features are rarely trained from scratch, and are often derived from high-performing classifiers that were pre-trained on other tasks, such as ResNet (He et al. [17]), GoogLeNet (Szegedy et al. [47]) or VGG (Simonyan and Zisserman [45]). In this purposely naive baseline architecture, the last layer of GoogLeNet before the softmax, is used as the feature. The image feature vector is concatenated with the embedded query vector and a shallow fully connected network generates an answer to the query with a softmax output. The network is trained with cross-entropy loss. However, this example network suffers from many limitations, two of which are critical to our task. First, the bag-of-words representation makes it difficult to encode complex queries. This would be fatal for a structured navigation language, as even simple instructions, such as “Go to the left of the car that is next to the building” would have the same encoding as “Go to the left of the building that is next to the car.” Second, it is unable to perform any kind of reasoning.
CHAPTER 4. NAVIGATING WITH NATURAL LANGUAGE COMMANDS AND STATIC AERIAL IMAGERY

Figure 4.1: Basic VQA architecture adapted from [55]. Image from SBD [11].

or attention process that are common in current best-performing architectures because there is no spatial information in the image features and no attention mechanism. Not surprisingly, this naive implementation was designed as a baseline and not as a competitive architecture. A typical solution to the text encoding problem is to use techniques that produce a sequence dependent representation such as a long short term memory (LSTM), convolutional neural network (CNN), or gated recurrent unit (GRU). Figure 4.2 is an example of a simple system from Agrawal et al. [1] that uses LSTMs, which is most frequently used method in VQA literature. In this case, the final hidden state of the LSTM is used to encode the question. We have omitted for clarity that each word is first encoded as a one-hot vector that is then embedded with a matrix $W_e$, followed by a tanh non-linearity. The word embedding matrix $W_e$ can be learned, but in other models the embedding is pre-trained, using GloVe (Pennington et al. [39]) or Word2vec (Mikolov et al. [30]) and others. The choice of encoding

Figure 4.2: VQA architecture with LSTM text encoding adapted from [1]. Word embedding left out for clarity.
can affect results, as Word2vec, for example, is designed to place semantically similar words, like “red” and “blue” in close proximity in the vector space, which may make discrimination between the two colors more difficult since they have a similar embedding. In this case, \( W_c \in \mathbb{R}^{300 \times N} \) where \( N \) is the number of words in the input vocabulary. Recall, the basic formulation of the LSTM, with \( \circ \) as the Hadamard product, \( \sigma \) is the sigmoid function, and \( \phi \) as the \text{tanh} function, is:

\[
\begin{align*}
    f_t &= \sigma(W_fx_t + U_fh_{t-1} + b_f) \\
    i_t &= \sigma(W_ix_t + U_ih_{t-1} + b_i) \\
    o_t &= \sigma(W_ox_t + U_oh_{t-1} + b_o) \\
    g_t &= \phi(W_cx_t + U_ch_{t-1} + b_c) \\
    c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
    h_t &= o_t \circ \phi(c_t) \\
    h_0 &= c_0 = 0
\end{align*}
\]

With the above choice for word embedding and an LSTM question embedding of 1024 dimensions, we see that the dimensions of the parameters in the LSTM are completely defined. More recent work in machine translation demonstrates that deeper LSTMs can perform better for, especially for sequence prediction (Graves [13]). This is more common in systems with have iterative processes that can benefit from sequence to sequence translation, as in Johnson et al. [21] and Hu et al. [19]. Additionally, the hidden state may be used as input to there processes as it evolves over time. The GRU, used in Xiong et al. [51], can be though of as a simplified LSTM, though its use in VQA seems less frequent. An example of the CNN used for encoding can be found in Yang et al. [53], although it since it is not recurrent, this network is limited to the three successive words of the CNN’s receptive field, limiting its ability to encode complex structure in the query. As for the example above, Agrawal et al. [1] try a 2 layer LSTM and observe some benefit, especially when combined with \( \ell_2 \) normalization of the image features. Like the prior example, the network is trained with cross-entropy loss.

**Attention Models**

Still missing from this architecture is any notion of spatial information. In the prior example, a query such as “Is there a dog in the image?”, may work, but a more complex question, such as, “Is there a dog resting next to the chair?”, will not simply because the combination of “dog”, “chair”, and “next to” are not possible to efficiently encode with a single flat representation. Attention mechanisms are one way to solve this. Because grounding is essential to navigation, the attention mechanism is of particular importance to this work. We will present three different types of attention mechanism; multiplicative, additive, outer product, and procedural. We will discuss the procedural models in Section 4.3 since these models have dynamically constructed networks that are very different from the other three. All of the methods presented are a form of soft-attention, where no one region is
singled out, but rather, where evidence is weighted relative to the region’s importance.

**Visual7W**

An example of a basic network that integrates spatial information with an LSTM is Visual7W from Zhu et al. [57] with their multiplicative attention model for grounded VQA (Figure 4.3). In this example, the authors use an LSTM in a process for encoding the question and image at the same time, and then use a decoding step to generate a solution. The only difference between Equations (4.1) to (4.4) and Equations (4.8) to (4.11) is the inclusion of an additional input to the forget, input, output, and the LSTM input functions.

\[
\begin{align*}
    f_t &= \sigma(W_f x_t + U_f h_{t-1} + W_{ff} r_{t-1} + b_f) \\
    i_t &= \sigma(W_i x_t + U_i h_{t-1} + W_{ir} r_{t-1} + b_i) \\
    o_t &= \sigma(W_o x_t + U_o h_{t-1} + W_{or} r_{t-1} + b_o) \\
    g_t &= \phi(W_c x_t + U_c h_{t-1} + W_{rc} r_{t-1} + b_c) \\
    c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
    h_t &= o_t \circ \phi(c_t) \\
    h_0 &= c_0 = 0
\end{align*}
\]

Figure 4.3: Simplified diagram of the Visual7W VQA network adapted from Zhu et al. [57]. The attention mechanism is integrated with the LSTM query encoding process. Not all attention map updates illustrated for clarity.
CHAPTER 4. NAVIGATING WITH NATURAL LANGUAGE COMMANDS AND STATIC AERIAL IMAGERY

This additional input is the attention mechanism, which is a function of the hidden state and the convolutional feature map $C(I) \in \mathbb{R}^{512 \times 14 \times 14}$ taken from the fourth convolutional layer of VGG-16. Equations (4.15) to (4.17) shows that the attention term is the weighted sum of the vectors in the feature map. Given the hidden state of the LSTM $h_{t-1}$ which is dependent on the prior words in the questions and the image, the attention mechanism adjust the weights of the image features to amplify those which are pertinent the question, as encoded so far.

$$ e_t = \omega_T \phi(W_{hc}h_{t-1} + W_{cc}C(I)) + b_a $$ (4.15)

$$ a_t = \text{softmax}(e_t) $$ (4.16)

$$ r_t = a_T C(I) $$ (4.17)

Finally, to complete the architecture, the initial inputs and final output to the process are defined as:

$$ x_0 = W_i F(I) + b_i $$ (4.18)

$$ x_1 = W_o OH(t_i) $$ (4.19)

$$ Z = \text{softmax}(F(I) \cdot h_n) $$ (4.20)

Here, $F(I)$ is the output of the last fully connected layer of VGG-16, and $OH$ is the one-hot representation of the word $t_i$. $W_o \in \mathbb{R}^{512 \times N}$ and $W_i \in \mathbb{R}^{512 \times 4096}$ are embedding matrices, where $N$ are the number of words in the vocabulary.

**Stacked Attention Network (SAN)**

Instead of integrating attention with question encoding, one can also make it an independent process, as in that Stacked Attention Network (SAN) from Yang et al. [53] (Figure 4.4). The additive attention mechanism in this model receives both the $512 \times 14 \times 14$ feature map from the last pooling layer of VGG-16 and the encoded question, which can either come from an LSTM or a CNN. For the sake of brevity, we will not describe the CNN question encoding, except to note that it is not common in the VQA literature\(^1\). The LSTM is the same as in Equations (4.1) to (4.4).

$$ h_A = \phi(W_{I,A}v_i \oplus (W_{Q,A}v_Q + b_A)) $$ (4.21)

$$ p_I = \text{softmax}(W_P h_A + b_P) $$ (4.22)

Here, $v_I \in \mathbb{R}^{d \times m}$ where $m$ are the number of image regions, 196 in this case (14 x 14) , and $d$ is the dimension of the feature vector, 512. This means the feature map from VGG-16 is converted from a 3D matrix to a 2D array of feature vectors. In Equation (4.21), the input image representation $v_I$ is embedded and combined with the question representation using the $\oplus$ operator. This simply adds

\(^1\)The CNN encoding is competitive and in many cases slightly better than the LSTM model for the datasets tested in Yang et al. [53]. Since this is not a NLP thesis, we will however follow the convention in the VQA field, and not go into detail about this method.
CHAPTER 4. NAVIGATING WITH NATURAL LANGUAGE COMMANDS AND STATIC AERIAL IMAGERY

Figure 4.4: Simplified diagram of a Stacked Attention Network adapted from Yang et al. [53]. The attention mechanism is a separate process after input encoding. In this case, there are two attention layers.

The result from $W_{Q,A}v_Q + b_A$ to each column vector in the matrix $W_{I,A}v_i$. ($W_{Q,A}, W_{I,A} \in \mathbb{R}^{k \times d}$)

The attention map is the result of a softmax operation Equation (4.22), as in Equation (4.16). In this case, $W_P \in \mathbb{R}^{1 \times k}$, so the final result is a probability for each of $m$ regions in the image. Each image feature is weighted by this attention map and summed, to compute a new, overall image representation vector. (Equations (4.23) to (4.24))

\[
\tilde{v}_I = \sum_i p_i v_i \tag{4.23}
\]

\[
u = \tilde{v}_I + v_Q \tag{4.24}
\]

In order to handle queries that require iterative refinement, such as, “What is on the desk next to the chair next to the sofa?”, this new representation $u$ can be processed again, as in Equations (4.25) to (4.26), to compute a new attention map based on what was discovered in the prior map. The new representation, $u^k$ is computed in a similar fashion as in Equations (4.23) to (4.24).

\[
h^k_A = \phi(W_{I,A}^k u_i \oplus (W_{Q,A}^k u_{k-1} + b^k_A)) \tag{4.25}
\]

\[
p^k_I = \text{softmax}(W_P^k h^k_A + b^k_P) \tag{4.26}
\]

The final answer is computed as:

\[
p_{\text{ans}} = \text{softmax}(W_u u^k + b_u) \tag{4.27}
\]

So far, we have seen attention via multiplication and addition, but there is also an attention mechanism that uses an outer product, embodied by MultiModal Compact Bilinear Pooling of Fukui et al. [10]. For the sake of brevity, we will not describe this method in detail at this point.
Adapting a VQA System for Navigation

As demonstrated in Chapter 3, it is possible to train a network to generate a cost map given static images and some discrete attributes. We intend to adapt a VQA system for this purpose by using the attention map as a cost map. However, before proceeding further, one must ask if its possible for the VQA networks to understand navigational concepts such as position and orientation for if a VQA system is unable to understand queries that utilize on these concepts, then there can be no hope of the attention map approximating a cost map. This question, of understanding whether a VQA system actually understands anything is not unique to our task. Dataset bias has long been a problem in deep learning. In Goyal et al. [12], the authors point out that “tennis” is the correct answer for 41% of questions starting with “What sport is,” and “yes” is the correct answer 87% of the time for questions starting with “Do you see a”. By producing alternate questions for the same image, they were able to determine if the VQA architecture is truly learning how to reason about the image, or are just exploiting biases in the question. The unfortunate conclusion of Goyal et al. [12] is that bias is a significant component of the apparent performance of many state of the art VQA architectures. Instead of supplementing existing, real-world data, Johnson et al. [20] choose to generate synthetic data. The goal of CLEVR is directed specifically at testing abstract reasoning processes. Therefore, real-world object identification is not a priority and synthetic images generated with a professional rendering tool are sufficient. Since the world is synthetic, the authors can guarantee the questions and images are unbiased. Of particular interest to this proposed work, the questions in CLEVR include spatial relationships, which are essential to robot navigation. Therefore, we feel that performance, as measured against this dataset, is the best indicator currently available of how similar architectures will behave when used for the robot navigation task. Figure 4.5 is an example of a typical CLEVR scene. Each object has one of 3 shapes, 8 colors, 2 materials and 2
CHAPTER 4. NAVIGATING WITH NATURAL LANGUAGE COMMANDS AND STATIC AERIAL IMAGERY

sizes. Between 3 and 10 objects are randomly generated and placed in the scene, ensuring that no object significantly obstructs another from the point of view of the camera. The data generation process produces scene graphs, which contain all information used to create the image. It is then a simple task to generate questions with known answers for training and testing. We investigate the top 5 VQA systems, as ranked by their self-reported accuracy on the CLEVR dataset as candidates for transformation into a robot navigation system. We categorize the different algorithms by their architectures. A monolithic architecture is one where there are no modules used to solve particular sub-problems, which is in contrast to a modular network which uses specialized sub-networks. A dynamic network will choose the appropriate module for each iteration of the problem solving process, while an iterative network maintains the same architecture, but applies it repeatedly, as in a traditional recurrent network. Co-attention is an iterative attention process, whereby the attention map is updated via repeated operations on both the image and query data. Finally, a static network will generate an answer in a single timestep, as in the traditional network in Figure 4.1. We see

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Accuracy %</th>
<th>Reasoning Model</th>
<th>Language Model</th>
<th>Attention Model</th>
<th>Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inferring &amp; Executing Programs (IEP) [21]</td>
<td>2017</td>
<td>88.6-96.9</td>
<td>Dynamic, Modular</td>
<td>2-layer LSTM</td>
<td>Procedural</td>
<td>Torch</td>
</tr>
<tr>
<td>End-to-End Module Networks (N2NMMN) [19]</td>
<td>2017</td>
<td>72.1</td>
<td>Dynamic, Modular</td>
<td>2-layer LSTM</td>
<td>Procedural</td>
<td>TensorFlow</td>
</tr>
<tr>
<td>Stacked Attention Networks (SAN) [53]</td>
<td>2016</td>
<td>68.5</td>
<td>Iterative, Monolithic</td>
<td>LSTM/CNN</td>
<td>Multiplicative co-attention</td>
<td>Theano²</td>
</tr>
<tr>
<td>Multimodal Compact Bilinear Pooling (MCB) [10]</td>
<td>2016</td>
<td>51.4</td>
<td>Static, Monolithic</td>
<td>2-layer LSTM</td>
<td>Bilinear</td>
<td>CAFFE</td>
</tr>
<tr>
<td>Visual7W [57]</td>
<td>2016</td>
<td>-</td>
<td>Static, Monolithic</td>
<td>LSTM</td>
<td>Additive co-attention</td>
<td>Torch</td>
</tr>
</tbody>
</table>

that there has been a significant increase in performance in the last year, as techniques moved to iterative networks, and then further towards specialized problem solving strategies, dynamically selected based on the query. Finally, these high performing architectures have also published source code that reproduces their reported performance, making modification possible without having to re-engineer each network.

As we have not yet introduced the procedural network model, we will first show how we intend to modify the Stacked Attention Network (SAN) (Yang et al. [53]) and the Visual7W network (Zhu et al. [57]). In order to modify these systems, we need to add a mechanism for generating a cost map and a mechanism for picking a starting location. We will see that there are architectural choices that we can make. The goal of this proposed work is to try these various options to see what architectures are most effective for this problem.

²Third party Torch implementation available.
CHAPTER 4. NAVIGATING WITH NATURAL LANGUAGE COMMANDS AND STATIC AERIAL IMAGERY

Modifying Visual7W

We will refer to this modified architectures as Nav-Visual7W (Figures 4.6 to 4.7) for which we have four variants. There are two variants for building the cost map, and two variants for encoding the initial position of the robot. To adapt this network for our problem, we first remove the decoding network and replace it with a cost map encoder.

Cost Map Decoding 1

In this variant, we adapt the existing attention mechanism to produce a cost map using the final hidden state of the LSTM at time \( N \). We modify Equations (4.15) to (4.17) to produce a cost map, instead of an attention map using separate weights for this step with respect to the weights used to compute the attention map in the LSTM. This network can be scene in Figure 4.6.

\[
e_N = w^T_M \phi(W_{hm}h_N + W_{cm}C(I)) + b_m \tag{4.28}
\]

\[
a_N = \text{softmax}(e_N) \tag{4.29}
\]

Given the cost map \( a_N \), we can then use a differentiable path planner, as introduced in Chapter 3,
Cost Map Decoding 2

A second possible mechanism for developing the cost map is to have a separate process for computing the cost map (Figure 4.7). This means adding an additional term to the LSTM network to accommodate an additional input, \( m_t \), in Equations (4.30) to (4.33).

\[
\begin{align*}
    f_t &= \sigma(W_f x_t + U_f h_{t-1} + W_{rf} r_t + W_{mf} m_t + b_f) \\
    i_t &= \sigma(W_i x_t + U_i h_{t-1} + W_{ri} r_t + W_{mi} m_t + b_i) \\
    o_t &= \sigma(W_o x_t + U_o h_{t-1} + W_{ro} r_t + W_{mo} m_t + b_o) \\
    g_t &= \phi(W_c x_t + U_c h_{t-1} + W_{rc} r_t + W_{mc} m_t + b_c) \\
    c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
    h_t &= o_t \circ \phi(c_t) \\
    h_0 &= c_0 = 0
\end{align*}
\]

The update can then follow Equations (4.15) to (4.16) for the update and Equations (4.28) to (4.29) for the final output. This concept is also a logical way to encode the problem since we construct a representation of the cost map over time, as it makes choices about attention, just as a human would.

Note the softmax may not be necessary. It imposes non-negativity constraints on the cost map, but also scales the cost map in a way that may not be desirable since the attention map is essentially a single probability distribution. A change to one element of the cost map necessitates a change to every other cost. Our experience in Chapter 3 has shown that non-negativity constraints are
useful, but they are not always necessary to get a successful solution. They are, however, helpful in avoiding local minima in the initial training of the network. It is possible to use projection methods to get a close by cost map that meets constraints, as in Ratliff et al. [41].

Path Initialization 1

We also need to encode the starting location of the path. Following the lead of the original architecture which initializes the LSTM with the final layer of VGG-16, we can insert an additional starting state between the first and second LSTM in Figure 4.6, which accepts an embedded one-hot vector $X$ that represents the starting location in matrix form. (This will be clearer once I have the figure rendered properly)

$$x'_0 = W_x X$$  \hspace{1cm} (4.37)

This will affect the hidden state that enters the LSTM which parses the instruction, which is sensible since how one interprets the command should depends on the starting location. It is possible to swap the $x_0$ input with the $x'_0$ input.

Path Initialization 2

A second mechanism for initializing the path is to force the attention mechanism to a particular value. This is demonstrated in Figure 4.7. Note the attention map is fixed, unlike the normal update. Note that all weights are non-zero so that spatial features are not completely lost, even if they do not correspond to the starting location.

Modifying SAN

We will refer to this modified architectures as Nav-SAN (Figure 4.8). Unlike Visual7W, the attention map in SAN is computed after the input encoding. We can simply take the last attention map as our cost map, removing the costmap and replacing it with the differentiable planner from Chapter 3.

We still, however, have the problem of encoding the starting location. One solution is to adopt the solution from Section 4.2.1, and have an initial input to the LSTM that encodes the starting location. This however, may not be a good idea. The LSTM has no notion about the features in the scene, unlike with Visual7W, and this added information about the starting location has not context. Therefore, an alternative solution is to modify the initial state of the output encoding. Again, assume $X$ is the one-hot encoding of the initial location, then:

Improving Map Resolution

In both the architectures shown in Section 4.2.1 and Section 4.2.2, the map resolution is limited to a small fraction of the input image size, making the attention map fairly useless. There are two potential solutions to this problem. Note that neither of the precursor networks are dependent
CHAPTER 4. NAVIGATING WITH NATURAL LANGUAGE COMMANDS AND STATIC AERIAL IMAGERY

Figure 4.8: Simplified diagram of a Stacked Attention Network modified for navigation.

on the specific CNN used to generated the feature maps that are used as input to the attention process. We can either choose a feature map from a lower level in the VGG network, or chose a different CNN, such as the fully convolutional neural network (FCN) from Shelhamer et al. [43]. This network fuses feature maps with low spatial resolution but high dimensional features, with high spatial resolution feature maps with lower dimensional features from lower in the network. By doing so, Shelhamer et al. [43] are able to construct a classified image at the same resolution as the input image.

Increasing the feature map resolution is not without cost. We see that weight matrices in Equation (4.15) and Equation (4.22) will have to increase greatly in size to the resolution of the feature map. However, because the attention mechanism in Visual7W operates once per LSTM iteration, one would expect a larger computational cost for training this model. How this translates to actual performance, regardless of computational cost is unknown, but part of what we intend to study.

Training

Both of the networks in Section 4.2.1 and Section 4.2.2 are fully differentiable, as is the loss functions we are using to build the cost map (See Chapter 3), we need no exotic techniques to train these networks. Visual7W was trained with ADAM and SAN with SGD. In both these models, the deep network used to generate features was fixed. This has worked well for our prior experiments, but it may be beneficial to allow some of the upper layers the latitude to adapt to the new task.

Modifications to the CLEVR Dataset

In order to create a suitable development environment, we propose to modify the CLEVR dataset generation tools to generate images from a top-down perspective, and to use the accompanying scene graph to generate navigation commands using a modification of the language introduced in
Figure 2.3. We have already accomplished part of this task, having generated a new dataset with a restricted set of navigational commands.

**Procedural Attention Models**

This section needs to be written. As these models are quite different from the older ones, I have separated them into a different section. In particular, we are interested in Johnson et al. [21] and Hu et al. [19]. These models learn a policy to dynamically construct a network out of specialized shallow network modules, based on the content of the question and the image. One way to look at this is that a the query is compiled into a program that operates on the image to generate an answer. This gives us the choice to compile the program using an LSTM, or to translate the program directly from the structured query language. We term this indirect and direct encoding, respectively, and we plan to investigate this further, as module networks are currently the best performing architectures on the CLEVR dataset.

**Experiments**

The purpose of this task is to understand the limitations of these algorithms, when suitably modified for the navigation task. We plan a series of experiments to this end. The first set of experiments will use a synthetic CLEVR dataset. The second set of experiments will take at least one of the best performing algorithms identified in the prior experiment, and apply it to a real aerial images.

There are three objectives to the first experiment.

First, we want to measure the efficacy of modified VQA architectures in learning cost maps over image induced by command. With synthetic data, we will be able to determine the limitations of what kinds of commands the network can understand with large amounts of unbiased data and without the complication of an overly complex perception environment.

Second, we would like to determine the difference between direct and indirect language encodings in context of precision navigation. This is only relevant to networks with procedural attention models which can support both models of language encoding. The former will allow more free-form and error-tolerant command encoding, while the later will give a more transparent view as to what the system is doing, which is important for a field robot.

Finally, we need to understand the affects of changing various other hyper-parameters. For example, how does feature map size affect training converge and final testing error? Concepts such “in front of” are not precise at the pixel level, and there is therefore some tolerance in the cost map resolution. Having lower resolution feature and cost maps may be beneficial to training testing time and converge since there are fewer parameters to learn.

We propose to use two baselines. First, we will use the planner in Boularias et al. [4] for which we have access to source code. Second, we can use the scene graph generated by CLEVR to construct a cost map programatically and use A* to compute an ideal path. Error for both methods will
be measured as the modified Hausdorff distance between the computed and predicted paths, as is customary in navigation literature.

The objective of the second experiment is to demonstrate that these techniques are applicable to real images. A large amount of data are required to train most deep networks in order for them to generalize well, and we don’t expect these networks to be any different. However, collecting real world images that have been properly labeled and also without some bias is challenging and time consuming. Therefore, we propose to use Amazon Mechanical Turk to collect a limited dataset. The labeler’s task, given an aerial image and a starting point and a command, will be to follow the command and draw a trajectory to the specified destination. We plan to modify the tool we developed for Task 1.

For training, we plan to mix this new data with the synthetic CLEVR data to make a hybrid dataset. We have seen this technique used in | (Need to find reference to paper where image captioning with cartoon images mixed with real data) and look to see if it will transfer to a new domain.

Conclusion

We have presented a path towards a system that can generate trajectories through a aerial image given a text command a starting point. To our knowledge, this is a novel way of solving this problem, but there are an increasing number of papers that use deep networks to generate cost maps directly from perception data. For example, recent work by Gupta et al. [15], shows that its possible to learn cost maps for each image that a robot sees as a travels through an office environment. Other work, such as Anderson et al. [2], use reinforcement learning to follow a path through a home environment that has a topological map. None of the literature we have seen, to this point, also integrate language and the paths are usually constrained by nature of the map (topological) or because the plan is rather short term.

At the conclusion of this task, we will be able to chose the best performing architecture in this simplified test environment of synthetic and aerial images, so that we can confidently apply it to more complex and practical problems in Chapter 5.

Timeline
CHAPTER 4. NAVIGATING WITH NATURAL LANGUAGE COMMANDS AND STATIC AERIAL IMAGERY

Figure 4.9: Timeline for task 2
Chapter 5

Navigating in Uncertain and Partially Observable Environments

Many problems that must be addressed to make this work more practical for a robot in the field. A fully observable, static world was selected as the development environment, but few real-world applications have such luxury, even if we can show our research at work on real aerial images. Through our literature survey, we have identified a set of qualities that a minimally practical navigation system must have. At the conclusion of this task, we expect to have a system which could be fielded on a robot similar to the one in Figure 2.1, while at the same time carefully sidestepping the complications that come with field testing on real hardware by using simulated environments where possible.

While we are interested in practical applications, based our own field experience, we are also aware of how much time and effort is required to make that possible while not directly contributing our research goals. For example, with great effort from a team of people over a period of several days, only 46 individual runs were executed in Oh et al. [35], testifying to the difficulty of merely testing a robot in the field, independent of the systems integration and development time. While we do not wish to discount the importance of designing algorithms that work on real robots, we also want to make a maximal contribution in the time allowed for this work.

Based on our literature review, we believe that a practical system for a ground robot must have at least the following qualities:

1. Produce stable paths over time.
2. Maintain a map-store that is updated over time, accumulating information about the world that has been observed.
3. Operate from a ground-vehicle perspective, implying not only a change in perspective between the robot and the planner but also that the world is not, in general, fully observable.
4. Efficiently explore the world when there is no definitive plan.
CHAPTER 5. NAVIGATING IN UNCERTAIN AND PARTIALLY OBSERVABLE ENVIRONMENTS

5. Reason about topological command constraints, as well as metric ones.

6. Declare success or detect a failure and ambiguous situations.

7. Integrate additional sources of information, such as maps, and demonstrate inter-operability with existing algorithms.

The existing algorithm from chapter 4 has no model of time since up to now, the world has been assumed to be static and fully observable. At time $t = 0$, the robot has complete information about the state of the world. This is generally not the case in reality, and as time progresses and the robot moves, new information will change the data available for path planning. A simple solution to this problem is to periodically re-plan as the robot gathers a better map of the environment, for example, as the robot’s pose changes significantly. (Algorithm 1) It is desirable that the algorithm generates paths which are stable over time unless contradictory evidence requires a change in plans.

Algorithm 2 A simplified planning loop with discrete memory. We use VGG-16 as a placeholder for spatial features from a generic deep network.

```
1: procedure SimplePlanner($\Lambda, I_0, M_0, X_0$)
2:     $\Lambda$ \hspace{1cm} \triangleright \text{Navigation Command}
3:     $I_0$ \hspace{1cm} \triangleright \text{Perception}
4:     $M_0$ \hspace{1cm} \triangleright \text{Map}
5:     $X_0$ \hspace{1cm} \triangleright \text{Position of robot}
6:     success $\leftarrow$ False
7:     failure $\leftarrow$ False
8:     $t \leftarrow 0$
9:     while success and failure do
10:        $M_t \leftarrow \text{UpdateMap}(M_{t-1}, X_t, \text{VGG16}(I_t))$ \hspace{1cm} \triangleright \text{Store deep features in map}
11:        $p_t \leftarrow \text{Plan}(M_t, X_t, \Lambda)$ \hspace{1cm} \triangleright \text{Update plan}
12:        $X_{t+1} \leftarrow \text{Execute}(p_t)$ \hspace{1cm} \triangleright \text{Execute plan one step and update location}
13:        $t \leftarrow t + 1$
14:        success, failure $\leftarrow \text{CommandComplete}(\Lambda, I_t, X_t, M_t)$
15:     end while
16:     return success, failure
17: end procedure
```

The navigation systems we have surveyed (Oh et al. [35], Hemachandra et al. [18], and others) have no explicit notion of time and are similarly invoked in a loop, without information about previous plans that can affect future decisions. That is not to say the representation of knowledge is static. Each contains a map, whether metric, topological, or both, that is updated periodically. However, the grounding and planning operations are independent of prior plans (Line 11).

This vastly simplifies the navigation algorithm since each call to the planner is independent, and the existing loss functions we have developed are sufficient. Indeed, given a static and fully observable world, the above loop should yield the same result at each iteration, with each call the to planner returning a path segment that is a subset of the initial plan as the robot moves. However, in a partially observable world, new information accumulated in the map may make another plan more desirable. In this case, we assume that any new plan will be sensible and the robot will not
constantly thrash between plans. (During our experiments, we have observed behavior in Oh et al. [35] that may be explained in this way.)

Another drawback to this architecture is that the memory update (Line 10 is independent of the planner. This has three implications. First, the map can contain only features that the designer deems relevant to the problem. In this case, we are storing only image features. The planner is unable to save other data that may be helpful in making future decisions. Second, we desire to utilize images from the robot’s perspective. The spatial component of the image features do not align with a grid map, and it is not clear how to make that transformation. Finally, given two sets of observations for the same map cell, we have no strategy for how to combine prior information with new information.

**Integrating Perspective Images, Memory and Planning**

Map building using a deep network is currently being explored by Zhang et al. [54] using concepts from Graves et al. [14] for a location addressable memory that is also a differentiable recurrent network. Zhang et al. [54] point out this is a feature not found in Gupta et al. [15] and other navigation systems that integrate neural networks, and this prevents the agent from making long-term plans. As the memory becomes part of the network, the features stored in it are also determined by training, as is the memory update rule.

Indirectly, this may also solve the problem of transforming deep features from a perspective image to features stored in the map. The transformation of image information from the ground robot’s perspective to a top-down view is a solved problem in computer vision. While the mathematics is clear in principle, there is no theory on how features transform with perspective. In addition, the observer has less information on objects that are far away since there are likely to be fewer pixels covering that object. A human would naturally adjust their confidence in what they see based on the type of object and its distance and maybe even orientation.

Rather than using an explicit model for feature transformation to convey a confidence measure based on distance an object type, this model can be learned as part of the deep network as demonstrated by Gupta et al. [15]. A robot’s perspective view is transformed into a metric map that ultimately conveys the potential reward for moving into an adjacent map cell when it comes to achieving the robot’s objective. This transformation is learned, not programmed, and results directly from having a loss function that operates in the aerial frame while receiving images from the ground view. As part of this process, Gupta et al. [15] can produce a confidence map that captures confidence models which have been challenging to design and are often ignored in integrated robot systems.

This more complicated system will require more complex training data and a more complex loss function. The planner is no longer independent at each iteration of the planning loop, and therefore it will need to train over a sequence of path predictions, each generated as the robot moves to a new location while executing the current best plan as part of some greater strategy (Section 5.0.1). The planning loss is the loss function we have studied already and reflects the quality of the current best
CHAPTER 5. NAVIGATING IN UNCERTAIN AND PARTIALLY OBSERVABLE ENVIRONMENTS

path given observed evidence. The strategy loss penalizes the system for prior plans that have not reached the target. This could be simply a measure of time or distance traveled for each partially completed plan or something more elaborate. Similarly, what that strategy should be or how to generate ground truth for it is something that will also be a topic of research in this task.

\[ L = \sum_{Planning} L_P(p_t) + \sum_{Strategy} L_s(p_i) \]

Other Goals

If the world is not fully observable, then it is likely that some referent in a command is not observable as well (Item 5). To complete a plan without seeing the objective, existing robot navigation systems use topological maps that operate in parallel to a metric map. The topological map conveys information about the relations between objects without requiring specific knowledge about where they exist in space. This might happen, for example, if a command refers to an object that is not in view, but that is known to be behind another object that is in view. In Duvallet [8], phantom goal points are generated in metric space to provide hypotheses over which the graphical model could reason using observable data as supporting evidence.

While it is not clear how to integrate topological and metric mapping into our current framework, it is worth noting that deep networks have been demonstrated to forecast using prior knowledge, the likely location of relevant objects. For example, in the semantic task from Gupta et al. [15], the network is trained to find chairs. At test time, it generates maps that guide the robot towards likely places where a chair could be found, even if unseen, based on the current image. While grounding is not a part of this model and the hypothetical target is not explicitly modeled, it does demonstrate the capacity to learn how to associate the visible world with likely locations of hidden objects.

A closely related issue to both items 3 to 4 and item 5, states that the robot should develop paths that sensibly explore the world if there is no concrete plan to the goal. This could mean that some or all of the referents are not initially in view and the robot must develop a search strategy to clarify the situation. For a robot with a camera that has a limited field of view, this strategy could be as simple as turning around, or as complex as exploring partially grounded paths one by one until enough evidence is gathered to form a concrete plan or declare failure. In navigation literature, this can be accomplished as an independent searching behavior or induced by seeding the unseen parts of the world with hypothetical destinations so that the navigation system explores the world in an orderly manner. Therefore, we propose that the navigation system we are designing should also provide a search behavior that utilizes knowledge of the world it sees currently and information about scenes it has been trained on to generate plans in ambiguous situations. Key to this point is that the exploration must be efficient in some measure. Simply performing a grid search is not a good strategy when partial information is available.

Finally, item 7 addresses practical problems that come about when integrating a new technology into an established platform. In particular, robots have many other real and virtual sensors, including
imaging LADAR and prior maps. Additionally, we need to understand if the information the system generates is sufficient to make the system usable, and if not, what additional outputs are required. For example, item 6 could have been easily overlooked, but its essential for any higher-level intelligence process to know when the planner believes that the command objective has been achieved and its time to move on to other tasks. It is also important to know how confident the navigation system is in this assertion. A low confidence of task completion would prompt the higher-level processes to enter a recovery mode, perhaps back-tracking or prompting the operator for clarification.

Proposed Simulation Experiments

We propose two sets of experiments to explore how we can evolve the navigation system into something more practical, being mindful that real-world experiments are often time-consuming. See Figure 5.2 for a timeline for the proposed tasks.

As a control on all experiments, we propose to use a system derived from Oh et al. [34] and continue to use a structured command language. We will modify one of the best performing architectures, as determined by our experiments in chapter 4 as the core of our new algorithm, while at the same time judging its suitability for this task in light of our specifications in items 1 to 7. Some architectures may more efficiently support our design objectives over others.

Task 3A: Navigation with Mapping and Exploration

This task will address item 1 and items 4 to 6 and is intended to explore problems with map building and partial observability in as simple an environment as possible. Because this task develops the basic theory for an exploring navigation system, we feel that the International Joint Conference on Artificial Intelligence (IJCAI) is an appropriate venue for this work.

Simple Loop Navigation and Strategic Exploration

The first sub-task is to build a simple navigation system similar to Algorithm 1. It is designed to be a baseline for more elaborate systems with integrated memory, but it also serves to develop basic concepts, such as what ground truth looks like for a system that plans and re-plans paths as the world is explored, and concurrently, what is an appropriate loss function. The concept of strategic exploration, the efficient exploration of an environment given a command and an incomplete grounding, will also be developed as part of this task and is intimately tied to the loss metric. Further, this simplified system will be a laboratory for developing hybrid topological/metric grounding, and for testing ideas on how to determine success or failure and how to measure ambiguous situations that require clarification.

Integrated Map Memory

The second sub-task is to integrate memory as part of the navigation system. The work by Zhang et al. [54] is somewhat new and not thoroughly developed, so its likely that other techniques will
be considered here. The goal is to produce a system where the contents of the memory and the
updates to the cells are learned, rather than explicitly modeled. This system will then be compared
against our baseline to determine the benefits of this more complex model.

Experiments with CLEVR-based Simulator

The prior tasks will be tested in as simple an environment as possible. We propose to use a modified
CLEVR-world by simulating a perception system that is position and time-dependent and which has
a limited field of view. (Figure 5.1) We can also use Amazon Mechanical Turk to label aerial images,
as in Chapter 3, but with the same constraints we use in the trivial CLEVR-based simulation. The
human will have to handle an ambiguous task over a sequence of images. We expect this to be a
more laborious task than our prior labeling tasks, and therefore entirely expect a smaller dataset
with limited quality.

With careful network design, it may be possible to use weights learned in a static world model as
initial weights for a system with a dynamic world model, thereby simplifying training. For example,
one could reduce the field of view of the simulated robot in the CLEVR experiment, from total
omniscience to near-sighted, with the network adapting to the perception system limitations slowly
CHAPTER 5. NAVIGATING IN UNCERTAIN AND PARTIALLY OBSERVABLE ENVIRONMENTS

and in the process, developing strategies to mitigate them.

Navigation with Perspective Images

Once the basic design has been tested, we can address item 3. Again, we feel that simulation is the best technique for getting results without undue complication. AirSim, (Shah et al. [42]) is a state of the art simulator-in-the-loop environment that is open source and well-suited for this task. It is currently being used to train deep reinforcement learning networks to steer vehicles through realistic environments. Naturally, development in a 3D simulated environment will be much more difficult than our prior work with 2D data but is still preferable to development and testing on actual hardware which requires large setup time.

The primary technological challenge is to integrate perspective images with the navigation system. Unlike the prior experiment, the robot does not have a 360-degree field of view, so we will have to modify our algorithm to accept both position and orientation information. It is not clear how the initial position and orientation of the robot will be encoded to the network. However, once this information is available, it also becomes possible to specify commands that reference objects relative to the robot's position and orientation. In addition, as part of its search behavior, the robot will also need to plan where to search and in which direction. However, we will assume the robot is still holonomic to keep path planning simple.

We expect that coding, training, and testing will dominate the work in this task since we must integrate the navigation system with a sophisticated simulation environment and train over a great many sequences over time. Additionally, we will need to find a test environment and develop suitable navigation tasks to make a fair assessment of the system.

At the conclusion, we believe that International Conference on Intelligent Robots (IROS) is an appropriate conference for this work, as the contribution is a demonstration of a system that is almost practical for real-world applications.

Conclusion

We propose several essential elements that a realistic navigation system for ground vehicles must have. The primary challenge is that not all referents in the command will be observable at the same time. This trivial change to the problem definition brings with it a great many challenges which we hope to address by adding capabilities for mapping, reasoning about unseen regions, and for exploration. A secondary challenge is to integrate into the aerial map, features observed from a different perspective. Training a deep network often requires a large amount of data that is difficult to label. This is made even more difficult by the fact that by adding time to the system, we would also need to train over a sequence of path predictions over time.

By using simulators, we hope to demonstrate that in principle, this technique can work on a real robot without having to spend inordinate effort labeling data. At its conclusion, we expect that our contribution will be a new and practical method for robot navigation of ground vehicles that can be
developed for a real-life application as future work.

Timeline

![Timeline for task 3]

Figure 5.2: Timeline for task 3
Bibliography


[10] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. EMNLP, 2016. 4.1.2, 4.1


in VQA matter: Elevating the role of image understanding in Visual Question Answering. In Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 4.2


[27] Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the talk: Connecting


2.1, 2.2