

DOCTORAL DISSERTATION

Predicting Sets and Lists: Theory and Practice

CMU-RI-TR-15-18

Debadeepta Dey
debadeep@ri.cmu.edu

The Robotics Institute
Carnegie Mellon University

Submitted in partial fulfillment
of the requirements of the degree of
Doctor of Philosophy in Robotics

2015

Doctoral Committee:

J. Andrew Bagnell, *Carnegie Mellon University* (Chair)
Martial Hebert, *Carnegie Mellon University*
Jeff Schneider, *Carnegie Mellon University*
Rich Caruana, *Microsoft Research, Redmond*
Eric Horvitz, *Microsoft Research, Redmond*

dbagnell@ri.cmu.edu
hebert@ri.cmu.edu
schneide@cs.cmu.edu
rcaruana@microsoft.com
horvitz@microsoft.com

Abstract

Increasingly, real world problems require multiple predictions while traditional supervised learning techniques focus on making a single best prediction. For instance in advertisement placement on the web, a *list* of advertisements is placed on a page with the objective of maximizing click-through rate on that list.

In this work, we build an efficient framework for making sets or lists of predictions where the objective is to optimize any utility function which is (monotone) submodular over a list of predictions. Other examples of tasks where multiple predictions are important include: grasp selection in robotic manipulation where the robot arm must evaluate a list of grasps with the aim of finding a successful grasp, as early on in the list as possible and trajectory selection for mobile ground robots where given the computational time limits, the task is to select a list of trajectories from a much larger set of feasible trajectories for minimizing expected cost of traversal. In computer vision tasks like frame-to-frame target tracking in video, multiple hypotheses about the target location and pose must be considered by the tracking algorithm.

For each of these cases, we optimize for the content and order of the list of predictions.

Crucially— and in contrast with existing work on list prediction — our approach to predicting lists is based on very simple *reductions* of the problem of predicting lists to a series of simple classification/regression tasks.

This provides powerful flexibility to use any existing prediction method while ensuring rigorous guarantees on prediction performance. We analyze these meta-algorithms for list prediction in both the online, no-regret and generalization settings.

Furthermore we extend the methods to make multiple predictions in structured output domains where even a single prediction is a combinatorial object, *e.g.* , challenging vision tasks like semantic scene labeling and monocular pose estimation.

We conclude with case studies that demonstrate the power and flexibility of these reductions in problems from document summarization, prediction of the pose of humans in images, to predicting the best set of robotic grasps and purely vision based autonomous flight in densely cluttered environments.

For Maa, Baba, Dadu and Shannon.

Acknowledgements

*Guru Brahma, Guru Vishnu, Guru devo Maheshwara,
Guru sakshat, param Brahma, tasmai shri guravay namah*

– Guru Mantra from ancient Vedas

This is a hymn in Sanskrit from the ancient Indian Vedas. It roughly translates as: the teacher is the remover (“Ru”) of darkness (“Gu”) and harbinger of awareness and knowledge and I offer my life in service of my teacher. In this role the “Guru” is considered to be the Creator (“Brahma”), the Preserver (“Vishnu”) as well as the Destroyer (“Maheshwara”) since without the “Guru”, one will have no awareness of even God.

I would like to begin by offering my thanks to every teacher I ever had, be it in a formal educational setting or in general. Without them this work would not exist today.

I will begin by thanking my advisor Drew Bagnell for opening the doors of many fields to me. I have admired and drawn inspiration from his fearlessness in delving deep into different fields and drawing novel connections. I am most grateful for his reinforcement of the principle of “deliberate practice” which gives me the confidence to get better over time in every aspect of my research. Thank you for teaching me to fish!

Not many people have the fortune to also have Martial Hebert as a co-advisor. His wisdom, insight and unconditional support have helped this body of work immensely. I have never seen someone care so much about presenting complicated concepts in an intuitive manner, be it at a conference or to novice students in class. I will strive to uphold his lofty standards in everything I do.

I would like to thank Eric Horvitz for many insightful discussions about bounded rationality and situated agents which have influenced my thought process and given this work a much more general flavor from the point of view of Artificial Intelligence; Rich Caruana for sharing tips on making machine learning actually work and emphasis on robust experimentation and analysis; Jeff Schneider for numerous suggestions for improvement and bringing to my attention ideas from Reinforcement Learning and Robotics that have enriched this work. Thank you all for being on my committee and supporting me through graduate school.

When I was a bright-eyed bushy-tailed undergraduate student in 2006, Christopher Geyer took me in and introduced me to the world of research at the Field Robotics Center (FRC), CMU. His love of Mathematics and Computer Vision remains an inspiration. I remember not being able to even write code properly at the time. Thank you for still believing in me in spite of my buggy camera drivers and for teaching me that meat does turn brown on being cooked!

Nicolas Vandapel taught me process. The large number of hours he has spent teaching

me to improve code, make presentations and think critically every step of the way will serve me for life. The fact that I can still hear his voice in my head when working maybe a topic for my therapist but serves me greatly in my work.

My mother was my first teacher who always upheld the highest expectations from me and never ever let me settle for anything less. From her I learnt to do research, to go deep into any topic and respect the sanctity of deep work. My father made sure that every obstacle was removed from my path for achieving my dreams. Dadu loved me unconditionally and bought me every book and toy that I ever wanted.

Shannon for loving me and tolerating all my childish tantrums and princely airs while I finished this work.

I am realizing while writing this that the list of people who touch one's life is too long, but I will like to attempt to thank as many as possible:

- Folks at Microsoft Research: Ashish Kapoor, Andrey Kolobov and Ece Kamar who created an amazing internship experience for me. Lily Mummert and Rahul Sukthankar for being wonderful mentors during my Intel Research internship. Your advice as I started my PhD was invaluable.
- Sanjiv Singh, Stephen Nuske, Ben Grocholsky, Joseph Djughash, Brennan Sellner, Bradley Hamner, Sebastian Scherer and other FRC folks who taught me a lot about Field Robotics. Sadly they could not teach me enough about college basketball to win the NSH 1111 memorial basketball pool even once in all these years. I will persevere.
- Anuj Kapuria was the person who was primarily responsible for introducing me to CMU. Hitech Robotics Systemz was where it all began for me in 2005. My heartfelt thanks to the entire Hitech team for my real introduction to Robotics outside of textbooks.
- Gaurav Taank for being the best friend ever and teaching me to take big risks and work hard. I have never seen a person work harder and with such passion. Thanks for being there always and I am pretty sure you will be a billionaire soon.
- The wonderful current and former members of Lairlab: Boris Sofman, Tommy Liu, Stephane Ross, Jiaji Zhou, Daniel Munoz, Alex Grubb, Paul Vernaza, Kris Kitani, Arun Venkataraman, Allie Del Giorno, Elliot Cuzzillo, Hanzhang Hu, Kevin Waugh, Shervin Javdani, Wen Sun for creating a very supportive atmosphere. I will miss our scrum and lab meetings.
- Kumar Shaurya Shankar, Andreas Wendel, Narek Melik-Barkhudarov, Sam Zeng, Shreyansh Daftry, Harsimrat Sandhawalia and Matthew Powers for helping get that quadcopter to fly!
- Jiuguang Wang, Tomas Simon, Supreeth Achar, Ermine Teves for just being awesome friends and bringing so much color to my life.
- Varun Ramakrishna, Abhinav Shrivastava, Ishan Misra and David Fouhey for sitting through my horrible practice talks. To make it up to you I will send you PowerPoint plots in Comic Sans font everyday.

- Rachel Burcin for being a wonderful friend and advisor, Suzanne Lyons Muth for making sure that I did not get lost, Jessica Butterbaugh and Dana Chaffin for being incredibly patient with my numerous requests.
- The Office of Naval Research (ONR) and Defense Advanced Research Projects Agency (DARPA) for their funding throughout my graduate school.

No work stands in isolation and this work would not have been possible without my co-authors:

- “Contextual Optimization of Lists”: Tommy Liu
- “Data-Efficient Contextual Optimization of Lists”: Stephane Ross, Jiaji Zhou, Yisong Yue
- “Multiple Output Structured Prediction”: Varun Ramakrishna
- “Autonomous UAV Flight through Dense Clutter”: Shreyansh Daftry, Sam Zeng, Narek Melik-Barkhudarov, Kumar Shaurya Shankar

Last but not the least, CMU and Robotics Institute for being an incredibly nurturing place.

Contents

1	Introduction	7
2	Contextual Optimization of Lists	11
2.1	Case Study: Manipulation Planning via Contextual Control Libraries	19
2.2	CONSEQOPT Analysis	23
2.2.1	Submodularity of Lists and the Greedy Algorithm	23
2.2.2	CONSEQOPT: Contextualizing the GREEDY Algorithm	25
3	Data-Efficient Contextual Optimization of Lists	29
3.1	SCP Analysis	35
3.2	Case Studies	38
3.2.1	Case Study: Robotic Manipulation Planning	38
3.2.2	Case Study: Personalized News Recommendation	39
3.2.3	Case Study: Document Summarization	40
4	Multiple Output Structured Prediction	42
4.1	Related Work	44
4.2	Approach	45
4.2.1	An Example	51
4.3	Case Studies	52
4.3.1	Case Study: Human Pose Tracking in Monocular Sequences	53
4.3.2	Case Study: Image Foreground/Background Separation	54
4.3.3	Case Study: Image Segmentation	57
5	Discussion	59
6	Autonomous UAV Flight through Dense Clutter	61
6.1	Hardware and Software Overview	64
6.1.1	Rover	64
6.1.2	UAV	66
6.2	Monocular Depth Prediction	66
6.2.1	Data Collection	66
6.2.2	Depth Prediction by Fast Non-linear Regression	68
6.2.3	Visual Odometry based Depth Prediction	74
6.2.4	Multiple Predictions	76
6.2.5	Pose Estimation	78
6.2.6	Planning and Control	80
6.3	Experiments	84

6.3.1	Perception Evaluation	84
6.3.2	System Performance Evaluation	84
6.4	Conclusion	87
7	Open Issues and Future Work	89
	Appendix	92
A	Proofs of SCP Theoretical Results	92
A.1	Preliminaries	92
A.2	Proofs of Main Results	95
B	Proof of Monotone Submodularity of Quality Function	101
	Bibliography	104

Introduction

Many real world problems require or benefit from *multiple predictions*. Consider advertisement placement on the web: a natural formulation involves the prediction of a *list* of advertisements to show—rather than a single choice—on a web-page with the objective of maximizing click-through rate [Bishop et al. 2006] of that list.

Other examples of domains where multiple predictions are important include: web search where a list of results are presented in response to a query [Carbonell and Goldstein 1998; Manning et al. 2008; Horvitz 2001], extractive document summarization where multiple representative sentences are selected from a document to summarize it [Ross et al. 2013b], grasp selection in robotic manipulation where the robot arm must evaluate a list of grasps with the aim of finding a successful grasp as early on in the list as possible, trajectory selection for mobile ground robots where the task is to select a list of trajectories from a much larger set of feasible trajectories for minimizing expected cost of traversal [Dey et al. 2012], or in frame-to-frame target tracking in video where multiple hypotheses about the target location and pose must be considered by the tracking algorithm [Pirsiavash et al. 2011; Park and Ramanan 2011]. This problem of returning a list of predictions to a query is remarkably common across tasks in fields like computer vision, path planning, manipulation, information retrieval, optimal control, computational biology, and human-robot interaction.

Traditionally, machine learning has developed mature tools with well understood theory and practice for producing a single best prediction. In this work we build simple, provably efficient tools for making multiple predictions whenever the objective is to optimize any objective function which is (monotone) submodular over a list of predictions. In spite of the ubiquity of such list prediction problems, tools for them have lagged behind the state-of-the-art for single best prediction.

Drawing upon recent advances in the optimization literature [Streeter and Golovin 2008; Radlinski et al. 2008; Munagala et al. 2005; Horvitz 2001; Golovin and Krause 2010; Krause

and Guestrin 2008], we develop a set of methods for learning to make multiple predictions with theoretical performance guarantees in a number of broad settings. For each setting we answer the questions 1) what is the right subset of predictions to evaluate given budgeted computation time and or 2) what is the right order of evaluating these predictions when mission success is critical. The settings can be broadly grouped into the following categories:

- **Library of choices:** In settings where the task is to predict a list from an a priori set of large number of choices and the only information available is via executing that list *e.g.* , path planning for autonomous mobile robots where the task is to select a subset of trajectories to evaluate on the terrain around the robot, from a large set of feasible trajectories. The true quality of a trajectory is revealed only after it has been traversed by the robot.
- **Contextual library of choices:** A library of choices where for every available choice additional information is available via features of the environment *e.g.* , advertisement placement, where for a given context on a web page a list of advertisements need to be selected to be placed in the side pane. Another example is making multiple predictions in structured output problems like monocular pose estimation where the task is to annotate the pose of a object from a single image. In this case the context is the features of the image and multiple predictions are necessary per image frame for a subsequent tracking algorithm to find the right pose.

Additionally the above settings can each be further subdivided into two additional categories:

- **Static:** In this setting the order in which queries are received by the system is independent of what list of choices are predicted for each query. The example of monocular pose estimation is a case where the list of predictions for a given query image does not influence in any manner the choice of the next query image.
- **Dynamic:** In contrast to the static setting, in the dynamic setting the choice of list of actions predicted directly influences the next query the system will receive. A good example is path planning for autonomous mobile robots where predicting which sets of trajectories to evaluate in the current step decides the next place in the world the robot will travel to in the next time step. Such dynamic settings also arise often in interactive systems. Other concrete examples include web search, manipulation, and conversational robots.

There are two main properties of a list that one cares about: 1) For a list of budgeted length, is the right answer contained within the list (*i.e.* a list with high recall)? 2) What is the order of evaluating a library of items so that the correct choice is encountered as early on

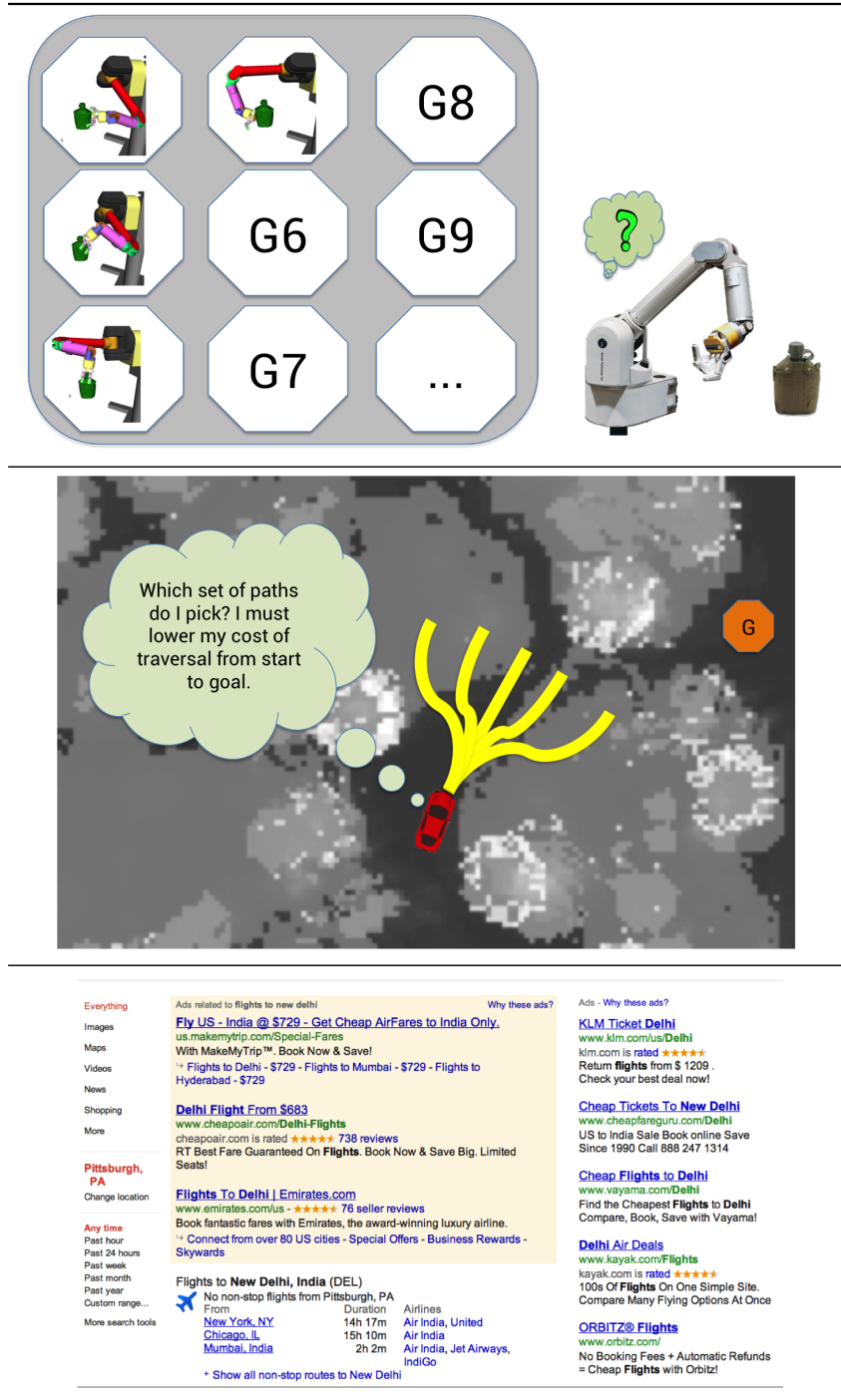


Table 1.1: Examples of multiple prediction applications **Top:** The manipulator must pick a list of grasps to try given a library of pre-computed grasps **Middle:** Autonomous mobile robots must decide which set of trajectories to evaluate given a budget on prediction time **Bottom:** A list of advertisements need to be predicted on a web page for optimizing click-through rate

as possible? In both cases we directly optimize for the objective of interest during training, without incorporating any additional parameters.

We will first discuss algorithms (Chapter 2) and provide a concrete example (Chapter 2.1) and then consider analysis (Chapter 2.2). In Chapter 3 we will detail more efficient versions, their detailed analysis and more case studies. Finally we will be considering the relationship with other approaches in literature for providing multiple predictions (Chapter 5).

Chapter 2 efficiently reduces the greedy algorithm to consider features of the environment by extending it to consider *lists of policies instead of lists of items*. We term this procedure CONSEQOPT short for “CONtextual SEquence OPTimization”. By competing against the best list of policies in a given policy class we produce better lists. Applications to robot motion planning for manipulation are presented.

Chapter 3, 3.1 proposes a more efficient version of CONSEQOPT which we term as SCP, short for “Sequential Contextual Prediction”. Chapter 3.2 demonstrates additional case studies for both SCP and CONSEQOPT.

Chapter 4 extends the approach of CONSEQOPT to *structured output problems* like semantic scene classification and monocular human pose estimation. The exponentially large label set requires care in implementing or approximating the weighting scheme of Chapter 2. We term the resulting family of algorithms SEQNBEST, a contraction of “SEquence N-Best”.

In Chapter 5, we review, compare and contrast alternate approaches to making multiple predictions. These include methods based on extracting multiple predictions from a single statistical model [Batra et al. \[2012\]](#), heuristic objective functions for constructing diverse predictions [Carbonell and Goldstein \[1998\]](#), and ad-hoc schemes for developing multiple models [Guzman-Rivera et al. \[2014b\]](#).

In Chapter 6 we apply the paradigm of making multiple predictions to enabling pure vision-based autonomous unmanned aerial vehicle (UAV) flight through dense clutter. We demonstrate real world experiments with an off-the-shelf quadrotor and show that by making multiple obstacle depth predictions, up to 71% increase in average flight distance can be achieved over making a single best prediction.

We will conclude with still open problems and future directions in Chapter 7.

Contextual Optimization of Lists

We begin by delving straight into the algorithm for predicting a *list* of items given some context of the example. By the word “context” we mean features of the example under consideration. A *list* is an ordered sequence of items of finite length. In Chapter 2.1 we will give a concrete case using this algorithm for robot motion planning. For now assume that we have a library of items \mathcal{A} and a dataset \mathcal{D} of $|\mathcal{D}|$ examples. We will use i to index position in the list and j to index examples. We use $\{\dots\}$ to indicate a list. A list of N items is indicated by $\{a_i\}_{i=1}^N$. Each example is represented by a feature vector \mathbf{x}_j of length L . By stacking \mathbf{x}_j we create a matrix \mathbf{X} of size $|\mathcal{D}| \times L$. Thus each row of \mathbf{X} represents a feature vector. \mathcal{S} is the space of all possible lists that can be made from items in library \mathcal{A} . Given such a feature matrix \mathbf{X} and library \mathcal{A} our aim is to predict a list of items $S \in \mathcal{S}$ of length $N \leq |\mathcal{D}|$ for each example, so that we can maximize $F = \mathbb{E}_{\mathbf{d} \sim \mathcal{D}}[f(S_j, \mathbf{x}_j)]$ where $f : \mathcal{S} \times \mathbf{d} \rightarrow [0, 1]$ is some task specific utility function we are interested in maximizing. Note that f is a *list function*, *i.e.* it takes in lists of items S_j as input and evaluates them on an example \mathbf{d} represented by feature vector \mathbf{x}_j and returns a non-negative score. The example \mathbf{d} is drawn from a (unknown) distribution ($\mathbf{d} \sim \mathcal{D}$) of examples. Later on we will show how special mathematical properties of f lead to performance guarantees of the algorithms we present in this work. We term the first algorithm we present below as CONSEQOPT BATCH which is short for “CONtextual SEquence OPTimization”. Due to the batch nature of training where it sees all the training data before it predicts a list, we append the word “BATCH”.

The input to CONSEQOPT BATCH (Algorithm 1) is the desired list length N , a multi-class classifier training procedure, the dataset \mathcal{D} of examples and library of items \mathcal{A} . The algorithm proceeds by training a classifier $\pi_i \in \mathcal{H}$ in the loop indexed by i and \mathcal{H} is the hypothesis space. Line 2 evaluates the function computeMarginalLoss and outputs a matrix \mathbf{M}_{L_i} . \mathbf{M}_{L_i} and the feature matrix \mathbf{X} are then used to train a multi-class cost-sensitive classifier π_i in line 3.

We use a toy example to detail every step of the training process. Suppose we have a

Algorithm 1 CONSEQOPT BATCH: Algorithm for training using classifiers

Input: List length N ,

Multi-class cost sensitive classifier training procedure: $\mathbb{R}^{|\mathcal{D}|\times L} \times \mathbb{R}^{|\mathcal{D}|\times |\mathcal{A}|} \rightarrow \mathcal{H}$,

Dataset \mathcal{D} of $|\mathcal{D}|$ number of examples and associated features $\mathbf{X} : \mathbb{R}^{|\mathcal{D}|\times L}$,

Library of items \mathcal{A}

Output: list of classifiers $\{\pi_1, \pi_2, \dots, \pi_N\}$ ($\pi : \mathbb{R}^{|\mathcal{D}|\times L} \rightarrow \mathcal{A}^{|\mathcal{D}|}$)

1: **for** $i = 1$ **to** N **do**

2: $\mathbf{M}_{\mathbf{L}_i} \leftarrow \text{computeMarginalLoss}(\mathbf{X}, \{\pi_1, \pi_2, \dots, \pi_{i-1}\}, \mathcal{A})$

3: $\pi_i \leftarrow \text{train}(\mathbf{X}, \mathbf{M}_{\mathbf{L}_i})$

4: **end for**

Algorithm 2 computeMarginalLoss($\mathbf{X}, \{\pi_1, \pi_2, \dots, \pi_{i-1}, \mathcal{A}\}$)

Input: Features of the dataset $\mathbf{X} : \mathbb{R}^{|\mathcal{D}|\times L}$,

List of multi-class cost sensitive classifiers $\{\pi_1, \pi_2, \dots, \pi_{i-1}\}$ ($\pi : \mathbb{R}^{|\mathcal{D}|\times L} \rightarrow \mathcal{A}^{|\mathcal{D}|}$),

Library of items \mathcal{A}

Output: $\mathbf{M}_{\mathbf{L}_i}$

1: **for** $j = 1$ **to** $|\mathcal{D}|$ **do**

2: **for** $k = 1$ **to** $|\mathcal{A}|$ **do**

3: $\mathbf{M}_{\mathbf{B}_i}[j, k] = f(\pi_1(\mathbf{x}_j) \oplus \pi_2(\mathbf{x}_j) \dots \pi_{i-1}(\mathbf{x}_j) \oplus \mathcal{A}[k], \mathbf{x}_j)$
 $- f(\pi_1(\mathbf{x}_j) \oplus \pi_2(\mathbf{x}_j) \dots \pi_{i-1}(\mathbf{x}_j), \mathbf{x}_j)$

4: **end for**

5: **end for**

6: $\mathbf{M}_{\mathbf{L}_i} = \text{convertGainsToLosses}(\mathbf{M}_{\mathbf{B}_i})$

Algorithm 3 CONSEQOPT BATCH: Algorithm for inference using classifiers

Input: List of multi-class cost-sensitive classifiers $\{\pi_1, \pi_2, \dots, \pi_N\}$,

Test example feature vector $\mathbf{x} : \mathbb{R}^{1\times L}$

Output: List of selected items $S = \{a_1, a_2, \dots, a_N\}$

1: Initialize empty list $S = \{\}$

2: **for** $i = 1$ **to** N **do**

3: $S \leftarrow S \oplus \pi_i(\mathbf{x})$

4: **end for**

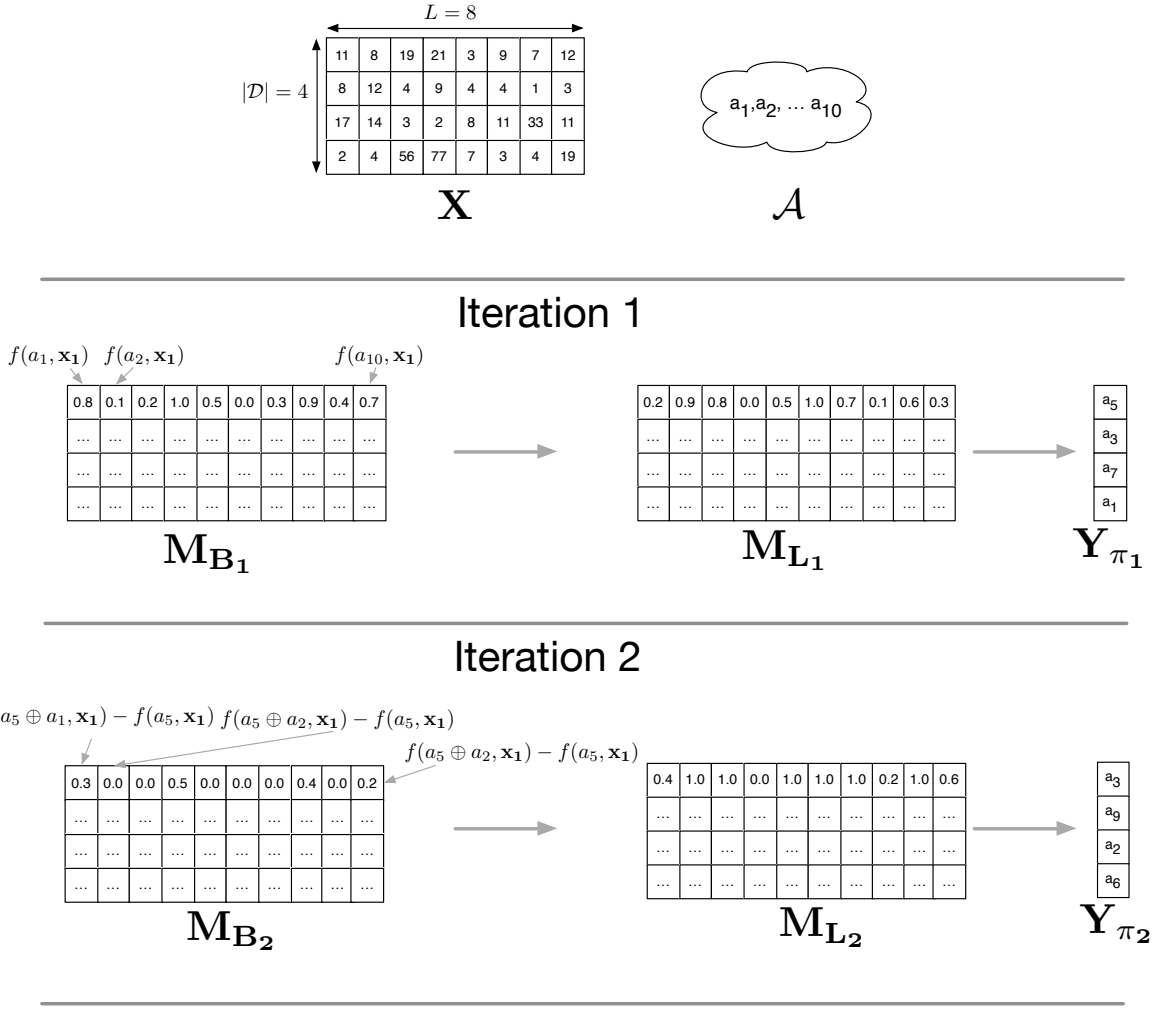


Figure 2.1: Illustration of two iterations of CONSEQOPT BATCH using multi-class classifiers on a toy dataset of 4 examples and library of 10 items.

dataset of $|\mathcal{D}| = 4$ examples, each example is described by a feature vector of dimension $L = 8$. So \mathbf{X} is of shape 4×8 . Also suppose that we have a library of $|\mathcal{A}| = 10$ items. Figure 2.1 shows the toy \mathbf{X} matrix, the library \mathcal{A} and the values of important variables for two iterations of the loop in Algorithm 1. We will now walk through the steps to illustrate how these numbers were obtained:

Iteration 1:

- **computeMarginalLosses:** In the first iteration, line 2 executes `computeMarginalLosses`. It takes in \mathbf{X} , \mathcal{A} and the output of previously trained classifiers on \mathbf{X} as input. Since at this point no previous classifiers have been trained, there are no previous predictions.

We first create the entries of the matrix $\mathbf{M}_{\mathbf{B}_1}$ by evaluating our objective function f on every example, for every item in the library \mathcal{A} . The columns correspond to items. For ease of illustration only the first row has been filled out. Say that when item a_1 is input to f on the first example, the output is $f(a_1, \mathbf{x}_1) = 0.8$. Similarly $f(a_2, \mathbf{x}_1) = 0.1$, $f(a_3, \mathbf{x}_1) = 0.2$, etc, until all columns are filled. This is repeated for every example until the whole matrix is filled. In this step the entries in each row of $\mathbf{M}_{\mathbf{B}_1}$ are subtracted from the maximum entry in that row, and rescaled to $[0, 1]$ to create $\mathbf{M}_{\mathbf{L}_1}$.

- **train:** In this step the train function is executed. This takes in the $\mathbf{M}_{\mathbf{L}_1}$ matrix computed in the last step and the features of the examples matrix \mathbf{X} as input and trains a cost-sensitive multi-class classifier π_1 [Langford and Beygelzimer 2005]. Each item in \mathcal{A} constitutes a class and the matrix $\mathbf{M}_{\mathbf{L}_1}$ constitutes the per example per class cost of predicting an item for that example. For example, for the first example the classifier will see that the best item to pick is a_4 since it is 0.0 cost, while picking a_6 is the worst since it pays a cost of 1.0.

Iteration 2:

- **computeMarginalLosses:** In the second iteration, line 2 executes computeMarginalLosses with \mathbf{X} , \mathcal{A} and the list of classifiers trained up to now, which in this case is just the first classifier π_1 from iteration 1. To create the entries of the $\mathbf{M}_{\mathbf{B}_2}$ matrix we first calculate the item chosen by π_1 on each example. So for every example the item that π_1 thought to be the best is obtained. Due to imperfect learning π_1 may not pick the best item for every example. For example, in the toy example it picks a_5 for the first example even though a_4 is the best one since it has 0.0 cost. In Figure 2.1 we show this step separately by using the variable \mathbf{Y}_{π_1} which stores this chosen item for every example.

Secondly the gain in f obtained by adding an item from \mathcal{A} is calculated. For example, for the first position in the first row we calculate $f(a_5 \oplus a_1, \mathbf{x}_1) - f(a_5, \mathbf{x}_1)$, for the second position $f(a_5 \oplus a_2, \mathbf{x}_1) - f(a_5, \mathbf{x}_1)$ and so on. In this example the numbers are calculated based on the function f being the max function. $f(a_5 \oplus a_1, \mathbf{x}_1)$ is therefore $\max(0.5, 0.8) = 0.8$. Note that $f(a_1, \mathbf{x}_1) = 0.8$ from $\mathbf{M}_{\mathbf{B}_1}$. Since $f(a_5, \mathbf{x}_1) = 0.5$, this leads us to $f(a_5 \oplus a_1, \mathbf{x}_1) - f(a_5, \mathbf{x}_1) = 0.8 - 0.5 = 0.3$. The rest of the entries are filled up this way. Once $\mathbf{M}_{\mathbf{B}_2}$ is calculated, the entries are then subtracted from 1 to turn gains to loss. This is the $\mathbf{M}_{\mathbf{L}_2}$ matrix.

- **train:** As in the first iteration the train function is executed to train a multi-class cost-sensitive classifier π_2 . This uses the rows of $\mathbf{M}_{\mathbf{L}_2}$ for obtaining per example per class costs.

The procedure for `computeMarginalLoss` is formally described in Algorithm 2. Algorithm 3 details the simple inference procedure. On a test example represented by feature vector \mathbf{x} the list of classifiers $\{\pi_1, \pi_2, \dots, \pi_N\}$ are invoked to construct the list of items S .

Algorithm 4 shows the equivalent algorithm using regressors instead of using classifiers (Algorithm 1). This alternate formulation has the advantage of being able to introduce new items to the library \mathcal{A} without retraining the sequence of regressors. Instead of directly identifying a target class, we use a regressor in each position of the list to produce an estimate of the gain from each item at that particular position. $\mathbf{M}_{\mathbf{B}_i}$ calculated in line 2 is a $|\mathcal{D}||\mathcal{A}|$ matrix of the actual marginal benefit computed in a similar fashion as $\mathbf{M}_{\mathbf{L}_i}$ of Algorithm 1, and $\tilde{\mathbf{M}}_{\mathbf{B}_i}$ is the estimate given by the regressor at the i^{th} position. In line 2 we compute the feature matrix \mathbf{X}_i and $\mathbf{M}_{\mathbf{B}_i}$ in the function `computeFeaturesAndBenefit`. This function is detailed in Algorithm 5. In this case, a feature vector is computed *per item* per example in the function `expressItemInExample`. The implementation of this function is application dependent and a concrete example will be provided in Chapter 2.1. By stacking up all such feature vectors \mathbf{x}_j where j indexes into examples, we build up the matrix \mathbf{X}_i . For feature vectors of length L , \mathbf{X}_i has dimensions $|\mathcal{D}||\mathcal{A}| \times L$. The features and gains at the i^{th} slot are used to train regressor \mathcal{R}_i and then invoked on the same training data producing the estimate $\tilde{\mathbf{M}}_{\mathbf{B}_i}$. Note that each data point consists of a row of the matrix \mathbf{X}_i and the target value is the corresponding entry for that item and example in $\mathbf{M}_{\mathbf{B}_i}$. For each example, we pick the item a which produces the maximum entry in the corresponding row in $\tilde{\mathbf{M}}_{\mathbf{B}_i}$ to be our chosen item for that example (Line 5 in Algorithm 4). These items are then stacked up to produce $\mathbf{Y}_{\mathcal{R}_i}$ which is then a $|\mathcal{D}|$ length vector as illustrated in Figure 2.1.

For inference, we take the list of trained regressors obtained from training and a test example \mathbf{d} , express all items in \mathcal{A} in this new example to obtain matrix \mathbf{X} , where each row is the feature vector describing an item from \mathcal{A} in that example. We invoke the regressor for that position of the list, on each feature vector in \mathbf{X} to get the predicted gains of each item at that particular position. We choose the item with the maximum such predicted gain and fill the list at that location with this item. This procedure is repeated for every position of the list.

Online Variants: Algorithms 7 and 8 detail the analogous online variants of CONSEQOPT BATCH using classifiers and regressors respectively. The online versions train N copies (one copy per position) of online variants of multi-class cost-sensitive classification and regression like online Support Vector Machines (SVM) [Hazan et al. 2007; Ratliff et al. 2007b], process one example at a time by sampling from \mathcal{D} , and updating themselves using the loss incurred on that example by using the “update” function to update the internal distribution over

Algorithm 4 CONSEQOPT BATCH: Algorithm for training using regressors

Input: List length N ,

Regressor training procedure: $\mathbb{R}^{|\mathcal{D}||\mathcal{A}| \times L} \times \mathbb{R}^{|\mathcal{D}||\mathcal{A}|} \rightarrow \mathcal{H}$,

Dataset \mathcal{D} of $|\mathcal{D}|$ number of examples,

Library of items \mathcal{A}

Output: list of regressors $\{\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_N\}$ ($\mathcal{R} : \mathbb{R}^{|\mathcal{D}||\mathcal{A}| \times L} \rightarrow \mathbb{R}^{|\mathcal{D}||\mathcal{A}|}$)

```
1: for  $i = 1$  to  $N$  do
2:    $\mathbf{X}_i, \mathbf{M}_{\mathbf{B}_i} \leftarrow \text{computeFeaturesAndBenefit}(\mathcal{D}, \{\mathbf{Y}_{\mathcal{R}_1}, \mathbf{Y}_{\mathcal{R}_2}, \dots, \mathbf{Y}_{\mathcal{R}_{i-1}}\}, \mathcal{A})$ 
3:    $\mathcal{R}_i \leftarrow \text{train}(\mathbf{X}_i, \mathbf{M}_{\mathbf{B}_i})$ 
4:    $\tilde{\mathbf{M}}_{\mathbf{B}_i} \leftarrow \mathcal{R}_i(\mathbf{X}_i)$ 
5:    $\mathbf{Y}_{\mathcal{R}_i} \leftarrow \text{argmax}(\tilde{\mathbf{M}}_{\mathbf{B}_i})$ 
6: end for
```

Algorithm 5 computeFeaturesAndBenefit($\mathcal{D}, \{\mathbf{Y}_{\mathcal{R}_1}, \mathbf{Y}_{\mathcal{R}_2}, \dots, \mathbf{Y}_{\mathcal{R}_{i-1}}\}, \mathcal{A}$)

Input: Dataset \mathcal{D} ,

Current regressor list output on dataset $\{\mathbf{Y}_{\mathcal{R}_1}, \mathbf{Y}_{\mathcal{R}_2}, \dots, \mathbf{Y}_{\mathcal{R}_{i-1}}\}$, ($\mathbf{Y}_{\mathcal{R}} : \mathcal{A}^{|\mathcal{D}|}$)

Library of items \mathcal{A}

Output: $\mathbf{M}_{\mathbf{B}_i} : \mathbb{R}^{|\mathcal{D}||\mathcal{A}|}$, $\mathbf{X}_i : \mathbb{R}^{|\mathcal{D}||\mathcal{A}| \times L}$

```
1:  $\mathbf{X}_i = []$ 
2: for  $j = 1$  to  $|\mathcal{D}|$  do
3:   for  $k = 1$  to  $|\mathcal{A}|$  do
4:      $\mathbf{M}_{\mathbf{B}_i}[j, k] = f(\mathbf{Y}_{\mathcal{R}_1}[j] \oplus \mathbf{Y}_{\mathcal{R}_2}[j] \dots \mathbf{Y}_{\mathcal{R}_{i-1}}[j] \oplus \mathcal{A}[k], \mathbf{x}_j) - f(\mathbf{Y}_{\mathcal{R}_1}[j] \oplus$   

        $\mathbf{Y}_{\mathcal{R}_2}[j] \dots \mathbf{Y}_{\mathcal{R}_{i-1}}[j], \mathbf{x}_j)$ 
5:      $\mathbf{x}_j \leftarrow \text{expressItemInExample}(\mathbf{d}_j, \mathcal{A}_k)$ 
6:      $\mathbf{X}_i \leftarrow \mathbf{X}_i \oplus \mathbf{x}_j$ 
7:   end for
8: end for
```

Algorithm 6 CONSEQOPT BATCH: Algorithm for inference using regressors

Input: Trained list of regressors $\{\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_N\}$,

Test example \mathbf{d}

Output: List of selected items $S = \{a_1, a_2, \dots, a_N\}$

```
1:  $S = \{\}$ 
2:  $\mathbf{X} = []$ 
3: for  $k = 1$  to  $|\mathcal{A}|$  do
4:    $\mathbf{x}_k \leftarrow \text{expressItemInExample}(\mathbf{d}, \mathcal{A}[k])$ 
5:    $\mathbf{X} \leftarrow \mathbf{X} \oplus \mathbf{x}_k$ 
6: end for
7: for  $i = 1$  to  $N$  do
8:    $a \leftarrow \text{argmax } \mathcal{R}_i(\mathbf{X})$ 
9:    $S \leftarrow S \oplus a$ 
10: end for
```

Algorithm 7 CONSEQOPT ONLINE: Algorithm for training using online classifiers

Input: List length N ,
Multi-class cost-sensitive online classifier training procedure: $\mathbb{R}^L \times \mathbb{R}^{|\mathcal{A}|} \rightarrow \mathcal{H}$,
Distribution of examples \mathcal{D} ,
Library of items \mathcal{A}

Output: list of online classifiers $\{\Phi_1, \Phi_2, \dots, \Phi_N\}$ ($\Phi : \mathbb{R}^L \rightarrow \mathcal{A}$)

```
1: for  $t = 1$  to  $T$  do
2:    $\mathbf{d}_t \leftarrow \text{Sample}(\mathcal{D})$ 
3:    $\mathbf{x}_t \leftarrow \text{computeFeatures}(\mathbf{d}_t)$ 
4:   for  $i = 1$  to  $N$  do
5:      $\mathbf{M}_{\mathbf{L}_{ti}} \leftarrow \text{computeMarginalLoss}(\mathbf{x}_t, \{\mathbf{Y}_{\Phi_1}, \dots, \mathbf{Y}_{\Phi_{i-1}}\}, \mathcal{A})$ 
6:      $\Phi_i \leftarrow \text{update}(\mathbf{x}_t, \mathbf{M}_{\mathbf{L}_{ti}})$ 
7:      $\mathbf{Y}_{\Phi_i} \leftarrow \Phi_i(\mathbf{x}_t)$ 
8:   end for
9: end for
```

Algorithm 8 CONSEQOPT ONLINE: Algorithm for training using online regressors

Input: List length N ,
Online regression training procedure: $\mathbb{R}^{|\mathcal{A}| \times L} \times \mathbb{R}^{|\mathcal{A}|} \rightarrow \mathcal{H}$,
Distribution of examples \mathcal{D} ,
Library of items \mathcal{A}

Output: list of online regressors $\{\Upsilon_1, \Upsilon_2, \dots, \Upsilon_N\}$ ($\Upsilon : \mathbb{R}^{|\mathcal{A}| \times L} \rightarrow \mathbb{R}^{|\mathcal{A}|}$)

```
1:  $\mathbf{d}_t \leftarrow \text{Sample}(\mathcal{D})$ 
2: for  $t = 1$  to  $T$  do
3:   for  $i = 1$  to  $N$  do
4:      $\mathbf{X}_{\mathbf{t}i}, \mathbf{M}_{\mathbf{B}_{ti}} \leftarrow \text{computeFeatureAndBenefit}(\mathbf{d}_{ti}, \{\mathbf{Y}_{\Upsilon_1}, \dots, \mathbf{Y}_{\Upsilon_{i-1}}\}, \mathcal{A})$ 
5:      $\Upsilon_i \leftarrow \text{update}(\mathbf{x}_{ti}, \mathbf{M}_{\mathbf{B}_{ti}})$ 
6:      $\tilde{\mathbf{M}}_{\mathbf{B}_{ti}} \leftarrow \Upsilon_i(\mathbf{x}_{ti})$ 
7:      $\mathbf{Y}_{\Upsilon_i} \leftarrow \text{argmax}(\tilde{\mathbf{M}}_{\mathbf{B}_{ti}})$ 
8:   end for
9: end for
```

learners.¹

Algorithm 7 runs for T rounds, sampling an example \mathbf{d}_t each round, computing its feature representation \mathbf{x}_t and using the list of online cost-sensitive classifiers Φ_i to compute the marginal loss of each action at every position and then updating itself using the marginal loss matrix $\mathbf{M}_{\mathbf{L}_{ti}}$. Note that $\mathbf{M}_{\mathbf{L}_{ti}}$ now contains only one row since there is only a single example. Then the updated online classifier is used to predict the correct item on the example to produce \mathbf{Y}_{Φ_i} .

Similarly, the online version using regressors in Algorithm 8 also samples an example each round and updates a list of online regressors Υ_i .

The inference procedures are the same as before in Algorithm 3 and 6 for classification and regression respectively.

¹In case of a deterministic algorithm, the update function updates the internal policy for choosing an item

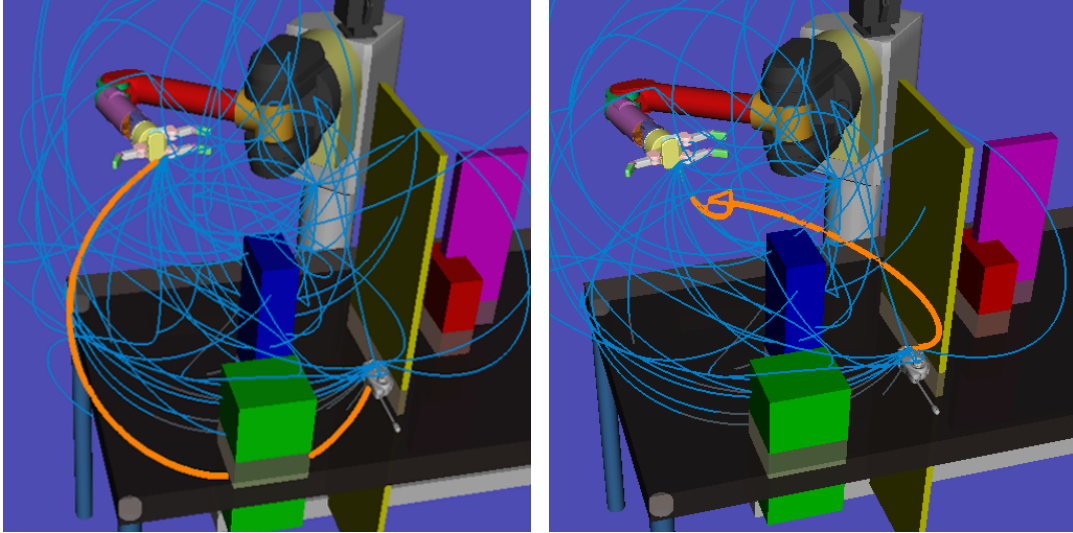
2.1 Case Study: Manipulation Planning via Contextual Control Libraries

In manipulation a common reoccurring task is to find a collision-free path from start to goal location. This is a challenging problem as this usually involves finding feasible paths in the configuration (joint) space of a manipulator which are typically high dimensional. Probabilistically complete planners like RRT [Kuffner and LaValle 2000] and PRM [Kavraki et al. 1996] are guaranteed to find a feasible path provided there exists at least one such path from start to goal. Unfortunately they can be expensive to run in such high dimensional spaces.

Recent work [Zucker et al. 2013; Jetchev and Toussaint 2009], has shown an alternate approach which proceed by relaxing the hard constraint of avoiding obstacles into a soft penalty term on collision and use simple local optimization techniques that quickly lead to smooth, collision-free trajectories suitable for robot execution. Often the default initialization trajectory seed is a simple straight-line initialization in joint space [Zucker et al. 2013]. This heuristic is surprisingly effective in many examples, but suffers from local convergence and may fail to find a trajectory even when one exists. In practice, this may be tackled by providing cleverer initialization seeds using classification [Jetchev and Toussaint 2009; Zucker 2009]. While these methods reduce the chance of falling into local minima, they do not have any alternative plans should the chosen initialization seed fail. A contextual ranking of a library of initialization seeds using CONSEQOPT can provide feasible alternative seeds should earlier choices fail. We take an approach similar to Dragan et al. [Dragan et al. 2011] where novel trajectories can be evaluated with respect to the environment the manipulator is operating in currently. A library of proposed initialization trajectory seeds can be developed in many ways including human demonstration [Ratliff et al. 2007a] or use of a slow but complete planner [Kuffner and LaValle 2000].

We conducted experiments where we attempt to plan a trajectory to a pre-grasp pose (goal) over the target object in a cluttered example using the local optimization planner CHOMP [Zucker et al. 2013] and minimize the total planning and execution time of the trajectory. A training dataset of $|\mathcal{D}| = 310$ examples and test dataset of 212 examples are generated. Each example contains a table surface with five obstacles and the target object is randomly placed on the table. The starting pose of the manipulator is randomly assigned, and the robot must find a collision-free trajectory to the end pose above the target object. To populate the control library, we consider initialization trajectories that move first to an "exploration point" and then to the goal. The exploration points are generated by randomly perturbing the midpoint of the original straight line initialization in joint space. The resulting

initial trajectories are then piecewise straight lines in joint space from the start point to the exploration point, and from the exploration point to the goal.² 30 trajectories generated with the above method form our control library \mathcal{A} . Figure 2.2a shows an example set for a particular example. Notice that in this case the straight-line initialization of CHOMP goes through the obstacle and therefore CHOMP has a difficult time finding a valid trajectory using this initial seed.



(a) The default straight-line initialization of CHOMP is in bold. Notice this initial seed goes straight through the obstacle and causes CHOMP to fail to find a collision-free trajectory. (b) The initialization seed for CHOMP found using CONSEQOPT is in bold. Using this initial seed CHOMP is able to find a collision free path that also has a relatively short execution time.

Figure 2.2: CHOMP initialization trajectories generated as control actions for CONSEQOPT. The end effector path of each trajectory in the library is traced out. The trajectory in bold in each image traces the initialization seed generated by the default straight-line approach and by CONSEQOPT, respectively.

In our results we use a small list (3 positions) to ensure the overhead of ordering and evaluating the library is small. When CHOMP fails to find a collision-free trajectory for multiple initializations seeds, one can always fall back on slow but complete planners. Thus the contextual control sequence’s role is to quickly evaluate a few good options and choose the initialization trajectory that will result in the minimum execution time. We note that in our experiments, the overhead of ordering and evaluating the library is negligible as we rely on a fast predictor and features computed as part of the trajectory optimization, and by choosing a small list length we can effectively compute a motion plan with expected planning time

²Half of the seed trajectories are prepended with a short path to start from an elbow-left configuration, and half are in an elbow-right configuration. This is because the local planner has a difficult time switching between configurations, while environmental context can provide much information about which configuration to use.

under 0.5 seconds. We can solve most manipulation problems that arise in our manipulation research very quickly, falling back to initializing the trajectory optimization with a complete motion planner only in the most difficult of circumstances.

For each initialization trajectory, we calculate 17 simple feature values which populate a row of the feature matrix \mathbf{X}_i .³ During training time, we evaluate each initialization seed in our library on all examples in the training set, and use their performance and features to train each regressor \mathcal{R}_i in CONSEQOPT BATCH (Algorithm 4). At inference time, we simply run Algorithm 6 to produce $\mathbf{Y}_{\mathcal{R}_1}, \dots, \mathbf{Y}_{\mathcal{R}_N}$ as the sequence of initialization seeds to be evaluated. Note that while the first regressor uses only the 17 basic features, the subsequent regressors also include the difference in feature values between the remaining actions and the actions chosen by the previous regressors. These difference features improve the algorithm’s ability to consider trajectory diversity in the chosen actions.

We compare CONSEQOPT BATCH with two methods of ranking the initialization library: a random ordering of the actions, and an ordering by sorting the output of the first regressor. Sorting by the first regressor is functionally the same as maximizing the absolute benefit rather than the marginal benefit at each slot. We compare the number of CHOMP failures as well as the average execution time of the final trajectory. For execution time, we assume the robot can be actuated at 1 rad/second for each joint and use the shortest trajectory generated using the N seeds ranked by CONSEQOPT BATCH as the performance. If we fail to find a collision free trajectory and need to fall back to a complete planner (RRT [Kuffner and LaValle 2000] plus trajectory optimization), we apply a maximum execution time penalty of 40 seconds due to the longer computation time and resulting trajectory.

The results over 212 test examples are summarized in Figure 2.3. With only simple straight line initialization, CHOMP is unable to find a collision free trajectory in 162/212 examples, with a resulting average execution time of 33.4 seconds. While a single regressor ($N = 1$) can reduce the number of CHOMP failures from 162 to 79 and the average execution time from 33.4 to 18.2 seconds, when we extend the sequence length, CONSEQOPT is able to reduce both metrics faster than a ranking by sorting the output of the first regressor. This is because for $N > 1$, CONSEQOPT BATCH chooses a primitive that provides the maximum marginal benefit, which results in trajectory seeds that have very different features from the previous slots’ choices. Ranking by the absolute benefit tends to pick trajectory seeds that are similar to each other, and thus are more likely to fail when the previous seeds fail. At

³Length of trajectory in joint space; length of trajectory in task space, the position (x,y,z) values of the end effector position at the exploration point (3 values), the distance field values used by CHOMP at the quarter points of the trajectory (3 values), joint values of the first 4 joints at both the exploration point (4 values) and the target pose (4 values), and whether the initialization seed is in the same left/right kinematic arm configuration.

a sequence length of 3, CONSEQOPT BATCH has only 16 failures and an average execution time of 8 seconds. A 90% improvement in success rate and a 75% reduction in execution time. Note that planning times are generally negligible compared to execution times for manipulation hence this improvement is significant. Figure 2.2b shows the initialization seed found by CONSEQOPT BATCH for the same example as in Figure 2.2a. Note that this seed avoids collision with the obstacle between the manipulator and the target object enabling CHOMP to produce a successful trajectory.

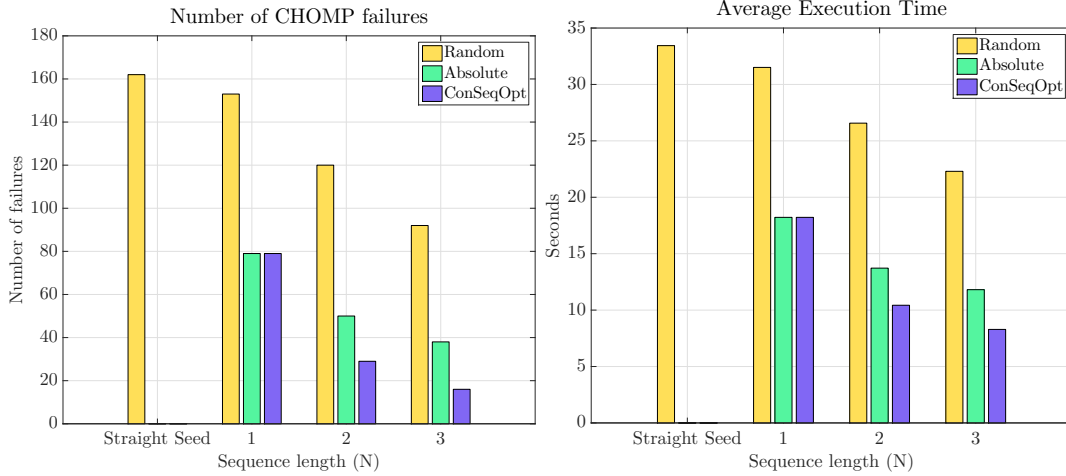


Figure 2.3: Results of CONSEQOPT BATCH for manipulation planning in 212 test examples. The top image shows the number of CHOMP failures for three different methods after each slot in the sequence. CONSEQOPT BATCH not only significantly reduces the number of CHOMP failures in the first slot, but also further reduces the failure rate faster than both the other methods when the sequence length is increased. The same trend is observed in the bottom image, which shows the average time to execute the chosen trajectory. The ‘Straight Seed’ column refers to the straight-line heuristic used by the original CHOMP implementation

We confirm from this experiment our initial intuition that diversity in high ranks is important to avoid selecting redundant items which are highly likely to fail together. One could conceivably enforce diversity by first finding an appropriate distance measure between items and then selecting items sequentially in a greedy-like manner where an item is only added to the list if it is maximum distance away from all previously selected items currently in the list. In practice such a procedure is difficult to implement because of two reasons: 1) the problem of finding the right distance measure between high dimensional objects like grasps, trajectories in configuration space etc is non-trivial and 2) this brings up the question of how much diversity is to be enforced and usually leads to free parameters which need to be then cross-validated [Batra et al. 2012]. By observing the mathematical properties of the task objective (monotone submodular) we elegantly sidestep both of these issues and are also able to provide performance guarantees of the algorithm.

2.2 ConSeqOpt Analysis

In this section we analyze CONSEQOPT BATCH in detail and provide details on when such a procedure will work, the explicit assumptions, the applicability of such assumptions to real world list prediction problems and performance guarantees of our methods.

2.2.1 Submodularity of Lists and the Greedy Algorithm

A function $f : \mathcal{S} \times \mathcal{D} \rightarrow [0, 1]$ is monotone submodular for any list $S \in \mathcal{S}$ and example $\mathbf{d} \sim \mathcal{D}$, where \mathcal{S} is the set of all lists if it satisfies the following two properties:

- (Monotonicity) for any list $S_1, S_2 \in \mathcal{S}$, $f(S_1) \leq f(S_1 \oplus S_2)$ and $f(S_2) \leq f(S_1 \oplus S_2)$
- (Submodularity) for any list $S_1, S_2 \in \mathcal{S}$, $f(S_1)$ and any item $a \in \mathcal{A}$, $f(S_1 \oplus S_2 \oplus a) - f(S_1 \oplus S_2) \leq f(S_1 \oplus a) - f(S_1)$

where \oplus means order dependent concatenation of lists, \mathcal{A} is the library of all available items. In this work we are concerned with *contextually* maximizing such monotone submodular objective functions. We will show through numerous case studies (Chapter 3.2) the wide real world applicability of CONSEQOPT and its more efficient variants.

So the first question that naturally arises is when a list length N is specified, what are the items from \mathcal{A} one should put in the list to maximize the objective of interest. Another closely related question is what is the order in which items in \mathcal{A} should be evaluated such that the probability of encountering an item which succeeds at the task at hand is maximized. This is exactly the question being asked in the robot manipulation path planning problem in Chapter 2.1. Unfortunately, the answer to both of the above questions was proven to be NP-hard by Nemhauser et al. [Nemhauser et al. 1978]. This implies that the only way to find the best list of specified length or the best ordering of all items in the library is to enumerate all possible lists of items and score each list using the monotone submodular objective and pick the highest scoring one. Even for relatively small sized libraries the set of all possible lists grows exponentially prohibitively large. For example for a library of 30 items, the set of all lists of length 30 has $30! = 2.6525286E+32$ lists in it (without replacement) and 30^{30} (with replacement).

But it turns out that there exist simple algorithms with good approximation guarantees for maximizing monotone submodular functions.

Consider the GREEDY algorithm (Algorithm 9) which proceeds by first initializing an empty list S and selecting each item such that the addition of the item to the existing list

Algorithm 9 GREEDY

Input: List length N ,

Library of items \mathcal{A}

Output: List of items $S = a_1, a_2, \dots, a_N$

```
1:  $S = \{\}$ 
2: for  $i = 1$  to  $N$  do
3:   for  $j = 1$  to  $|\mathcal{A}|$  do
4:      $g_j = \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(S \oplus a_j, \mathbf{d}) - f(S, \mathbf{d})$ 
5:   end for
6:    $a = \operatorname{argmax}_{j \in 1, \dots, |\mathcal{A}|} g_j$ 
7:    $S \leftarrow S \oplus a$ 
8: end for
```

up to that point increases f the most in expectation over a distribution of examples $\mathbf{d} \sim \mathcal{D}$. The loop stops once N items have been picked.

Feige proved [Feige 1998] that the list returned by the greedy algorithm is guaranteed to achieve $(1 - \frac{1}{e})$ of the optimal list of length N *i.e.* :

$$\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(S_{\text{greedy}}, \mathbf{d}) \geq \left(1 - \frac{1}{e}\right) \max_{|S| \leq N} \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(S, \mathbf{d}). \quad (2.1)$$

This problem of finding the optimal list of budgeted size N has been studied in literature as the BUDGETED MAXIMUM SUBMODULAR COVERAGE problem [Khuller et al. 1999; Streeter and Golovin 2008].

Additionally Feige et al., [Feige et al. 2004] proved that for the problem of finding the optimal ordering of *all* items in \mathcal{A} so that the successful item can be encountered as early on as possible, the ordering returned by GREEDY is guaranteed to find the successful item within 4 times the depth of the optimal ordering:

$$\text{cost}(S_{\text{greedy}}) \leq 4 \text{cost}\left(\operatorname{argmax}_{|S| \leq |\mathcal{A}|} \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(S, \mathbf{d})\right), \quad (2.2)$$

where $\text{cost}(S)$ is the number of elements of the list we had to go through to find the one that succeeded at the task at hand. Such problems have been studied in literature as MIN-SUM SUBMODULAR SET COVER [Feige et al. 2004; Streeter and Golovin 2008; Munagala et al. 2005].

In the manipulation planning example in Chapter 2.1, the task is to find an ordering of the initial trajectories in the library of 30 initial trajectories so that we can encounter the trajectory which succeeds in finding a collision free path as early on as possible. This

can be equivalently phrased as maximizing the probability of encountering a successful initial trajectory in a given list S of trajectories over a distribution of examples $\mathbf{d} \sim \mathcal{D}$, *i.e.* $\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(S, \mathbf{d}) = \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} P(S, \mathbf{d})$. Streeter and Golovin [Streeter and Golovin 2008] proved that such objectives are monotone, submodular. Therefore given a dataset of training examples one can use the GREEDY algorithm detailed above (Algorithm 9) for finding the best ordering of the initial trajectories in the library $|\mathcal{A}|$. The GREEDY solution will have the performance guarantees outlined above for both BUDGETED MAXIMUM SUBMODULAR COVERAGE and MIN-SUM SUBMODULAR SET COVER.

Intuitively, the GREEDY method is picking items in the list which provide the maximum marginal benefit at that position of the list. This way additions to the list which do not result in much gain to the objective are avoided. Another way of looking at this for the manipulation problem is to imagine picking an initial trajectory in the first position of the list hoping that it will be successful over most of the examples it encounters, but due to imperfection in both data and evaluation it fails. What this immediately shows is that initial trajectories similar to the first one are likely to fail as well and should not be selected for the second position. Initial trajectories which are different from the first one, but are still likely to succeed should be selected (diverse but relevant). The GREEDY algorithm naturally captures this notion of diversity and relevance when the objective is monotone submodular and comes up with an optimal list up to approximation guarantees.

But the GREEDY algorithm has a crucial limitation: it ignores the context of the example while constructing the list of initial trajectories. Ideally we would like a method that takes into account the rich features of the example available through the various sensors (*e.g.* cameras, depth cameras, lidars) mounted on the manipulator to aid in predicting a list of initial trajectories to maximize the probability of finding a successful one as early as possible. In the following section we analyze how CONSEQOPT overcomes this limitation while maintaining the performance guarantees of the GREEDY algorithm.

2.2.2 ConSeqOpt: Contextualizing the Greedy Algorithm

We will first examine how the GREEDY algorithm is contextualized by a reduction to learning a list of classifiers to result in the CONSEQOPT BATCH algorithm detailed in Algorithms 1 and 4.

In machine learning, reduction [Alina Beygelzimer and Zadrozny 2009] is the process of breaking down a challenging problem for which no easy solution exists to smaller problems for which well-understood theoretical and practical solutions exist. By relating the performance of the solutions to the smaller problems, statements can be made about the quality of the

solution to the more challenging problem.

We reduce the problem of predicting lists to invoking a list of multi-class classifiers, each of which in turn predicts an item in the resulting list. We establish a formal regret reduction [Beygelzimer et al. 2005] between cost sensitive multi-class classification error and the resulting error on the learned list of classifiers. Specifically, we demonstrate that if we consider the items in \mathcal{A} to be the classes and train a series of classifiers—one for each position of the list—on the features of a distribution of examples, we can then produce a near-optimal list of classifiers, which in turn can be invoked to produce the near-optimal list of items for a test example.

Theorem 1. *If each of the classifiers $\{\pi_1, \pi_2, \dots, \pi_1, \dots, \pi_N\}$ trained in Algorithm 1 achieves multi-class cost-sensitive regret of r_i , then the resulting list of classifiers is within at least $(1 - \frac{1}{e}) \max_{S \in \mathcal{S}} \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(S, \mathbf{d}) - \sum_{i=1}^N r_i$ of the optimal **such list of classifiers** S from the same hypothesis space.*⁴

Proof. (Sketch) Define the loss of a multi-class, cost-sensitive classifier π over a distribution of examples \mathcal{D} as $l(\pi, \mathcal{D})$. Each example can be represented as $(x_n, m_n^1, m_n^2, m_n^3, \dots, m_n^{|\mathcal{A}|})$ where x_n is the set of features representing the n^{th} example and $m_n^1, m_n^2, m_n^3, \dots, m_n^{|\mathcal{A}|}$ are the per class costs of misclassifying x_n . $m_n^1, m_n^2, m_n^3, \dots, m_n^{|\mathcal{A}|}$ are simply the n^{th} row of $\mathbf{M}_{\mathbf{L}_i}$ (which corresponds to the n^{th} example in the dataset \mathcal{D}). The best class has a 0 misclassification cost and while others are greater than equal to 0 (There might be multiple actions which will yield equal marginal benefit). Classifiers generally minimize the expected loss $l(\pi, \mathcal{D}) = \mathbb{E}_{(x_n, m_n^1, m_n^2, m_n^3, \dots, m_n^{|\mathcal{A}|}) \sim \mathcal{D}} [C_{\pi(x_n)}]$ where $C_{\pi(x_n)} = m_n^{\pi(x_n)}$ denotes the example-dependent multi-class misclassification cost. The best classifier in the hypothesis space \mathcal{H} minimizes $l(\pi, \mathcal{D})$

$$\pi^* = \operatorname{argmin}_{\pi \in \Pi} \mathbb{E}_{(x_n, m_n^1, m_n^2, m_n^3, \dots, m_n^{|\mathcal{A}|}) \sim \mathcal{D}} [C_{\pi(x_n)}]. \quad (2.3)$$

The regret of π is defined as $r = l(\pi, \mathcal{D}) - l(\pi^*, \mathcal{D})$. Each classifier associated with i^{th} slot of the list has a regret r_i .

Streeter et al. [Streeter and Golovin 2008] consider the case where the i^{th} decision made by the greedy algorithm is performed with additive error ϵ_i . Denote by $\hat{S} = \langle \hat{s}_1, \hat{s}_2, \dots, \hat{s}_N \rangle$ a variant of the list S in which the i^{th} argmax is evaluated with additive error ϵ_i . This can

⁴When the objective is to minimize the time (depth in list) to find a satisficing element then the resulting list of classifiers $\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(\hat{S}_{\langle N \rangle}, \mathbf{d}) \leq 4 \int_0^\infty 1 - \max_{S \in \mathcal{S}} \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(S_{\langle n \rangle}, \mathbf{d}) dn + \sum_{i=1}^N r_i$.

be formalized as

$$\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} \left[f(\hat{S}_i \oplus \hat{s}_i, \mathbf{d}) - f(\hat{S}_i, \mathbf{d}) \right] \geq \max_{s_i \in \mathcal{A}} \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} \left[f(\hat{S}_i \oplus s_i, \mathbf{d}) - f(\hat{S}_i, \mathbf{d}) \right] - \epsilon_i, \quad (2.4)$$

where $\hat{S}_0 = \langle \rangle$, $\hat{S}_i = \langle \hat{s}_1, \hat{s}_2, \hat{s}_3, \dots, \hat{s}_{i-1} \rangle$ for $i \geq 1$ and s_i is the predicted item by classifier π_i . They demonstrate that, for a budget or list length of N

$$\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(\hat{S}_{\langle N \rangle}, \mathbf{d}) \geq (1 - \frac{1}{e}) \max_{S \in \mathcal{S}} \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(S, \mathbf{d}) - \sum_{i=1}^N \epsilon_i, \quad (2.5)$$

assuming each item takes equal time to execute.

Thus the i^{th} argmax in (2.4) is chosen with some error $\epsilon_i = r_i$. An ϵ_i error made by classifier π_i corresponds to the classifier picking an item whose gain is ϵ_i less than the maximum possible. Hence the performance bound on additive error greedy list construction stated in (2.5) can be restated as

$$\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(\hat{S}_{\langle N \rangle}, \mathbf{d}) \geq (1 - \frac{1}{e}) \max_{S \in \mathcal{S}} \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(S, \mathbf{d}) - \sum_{i=1}^N r_i. \quad (2.6)$$

□

Theorem 2. *The list of squared-loss regressors $\{\mathcal{R}_1, \dots, \mathcal{R}_i, \dots, \mathcal{R}_N\}$ trained in Algorithm 4 is within at least $(1 - \frac{1}{e}) \max_{S \in \mathcal{S}} f(S) - \sum_{i=1}^N \sqrt{2(|\mathcal{A}| - 1)r_{\text{reg}_i}}$ of the optimal list of classifiers S from the hypothesis space of multi-class cost-sensitive classifiers.*

Proof. (Sketch) Langford et al. [Langford and Beygelzimer 2005] show that the regret reduction from multi-class classification to squared-loss regression has a regret of $\sqrt{2(|k| - 1)r_{\text{reg}}}$ where k is the number of classes and r_{reg} is the squared-loss regret on the underlying regression problem. In Algorithm 4 we use squared-loss regression to perform multi-class classification thereby incurring for each slot of the list a reduction regret of $\sqrt{2(|\mathcal{A}| - 1)r_{\text{reg}_i}}$ where $|\mathcal{A}|$ is the number of items in the library and r_{reg_i} is the regret of the regressor for the i^{th} slot. Theorem 1 states that the list of classifiers achieve $\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(\hat{S}_{\langle N \rangle}, \mathbf{d}) \geq (1 - \frac{1}{e}) \max_{S \in \mathcal{S}} \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(S, \mathbf{d}) - \sum_{i=1}^N r_i$ of the optimal list of classifiers. Plugging in the regret reduction from [Langford and Beygelzimer 2005] we get the result that the resulting list of regressors in Algorithm 4 is within at least $(1 - \frac{1}{e}) \max_{S \in \mathcal{S}} \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} f(S, \mathbf{d}) - \sum_{i=1}^N \sqrt{2(|\mathcal{A}| - 1)r_{\text{reg}_i}}$ of the optimal list of multi-class cost-sensitive classifiers. □

Theorem 1 proves that CONSEQOPT BATCH using classifiers efficiently finds the approximately greedy *list of classifiers* from the hypothesis space of all such classifiers. Similarly

Theorem 2 proves that CONSEQOPT BATCH using regressors also find the approximately greedy *list of regressors* from the hypothesis space of all such regressors.

Data-Efficient Contextual Optimization of Lists

In this chapter we develop a more efficient version of CONSEQOPT. CONSEQOPT sequentially trains N classifiers or regressors, one for each position of the list. Since predictors responsible for earlier positions of the list will generally start choosing optimal or near-optimal items for most examples in the dataset, the predictors responsible for later positions often do not observe enough examples to learn effectively to predict items which will bring the maximum gain to the objective at those positions. In other words the later predictors usually starve for data unless a large amount of data is available to begin with. We propose a closely related approach which trains a *single* (no-regret) online learner (policy) to produce a list of predictions. We term this approach as “Submodular Contextual Policy” algorithm, or in short SCP.

By leveraging recent work in imitation learning [Ross et al. 2011], SCP preserves similar performance guarantees as CONSEQOPT while being more data-efficient since all the data is used for training the learner. We will first describe the algorithm for the context-free case, where no features of the example are available. In this case as with the GREEDY algorithm, the performance of a list $S \in \mathcal{S}$ is evaluated by its expected value over an unknown distribution of examples $d \sim \mathcal{D}$: $\mathbb{E}_{d \sim \mathcal{D}} f(S, \mathbf{d})$ where f is monotone submodular. The GREEDY algorithm with perfect knowledge of \mathcal{D} can find a list S of length N such that it has the performance guarantees listed in 2.1 and 2.2 for the MIN-SUM SUBMODULAR SET COVER and BUDGETED MAXIMUM SUBMODULAR COVERAGE problems respectively. Although \mathcal{D} is unknown, we assume (as in the case of CONSEQOPT) that we observe samples $\mathbf{d} \sim \mathcal{D}$ and can evaluate any list $S \in \mathcal{S}$ using the objective function $f(S, \mathbf{d})$ during training. Our goal is to develop a computationally and statistically more efficient algorithm, which has similar performance guarantees as CONSEQOPT.

Algorithm 10 describes SCP in the context-free setting for the online BUDGETED MAX-

Algorithm 10 SCP: Algorithm for training in context-free setting

Input: List length N ,

 List length of best list to compete against K ,

 No-regret online learner routine “update”,

 Library of items \mathcal{A}

Output: Learnt internal distribution over items $p : \mathcal{A} \rightarrow [0, 1]$

```
1: for  $t = 1$  to  $T$  do
2:    $S_t = \{\}$ 
3:   for  $i = 1$  to  $N$  do
4:      $a \leftarrow \text{sample}(p_t)$ 
5:      $S_t \leftarrow S_t \oplus a$ 
6:   end for
7:    $\mathbf{d} \leftarrow \text{sample}(\mathcal{D})$ 
8:   for all  $a \in \mathcal{A}$  do
9:      $r_t(a) \leftarrow \sum_{i=1}^N (1 - \frac{1}{K})^{N-j} b(a|S_{t,i-1}, \mathbf{d})$ 
10:  end for
11:  for all  $a \in \mathcal{A}$  do
12:     $l_t(a) \leftarrow \max_{a' \in \mathcal{A}} r_t(a') - r_t(a)$ 
13:  end for
14:  for all  $a \in \mathcal{A}$  do
15:     $p_{t+1} \leftarrow \text{update}(l_t(a))$ 
16:  end for
17: end for
```

Algorithm 11 SCP: Algorithm for inference in context-free setting

Input: List length N ,

 Learnt internal distribution over items $p : \mathcal{A} \rightarrow [0, 1]$,

 Library of items \mathcal{A}

Output: $S \in \mathcal{S}$ of length N

```
1:  $S = \{\}$ 
2: for  $i = 1$  to  $N$  do
3:    $a \leftarrow \text{sample}(p)$ 
4:    $S \leftarrow S \oplus a$ 
5: end for
```

IMUM SUBMODULAR COVERAGE setting. SCP requires an online learning algorithm subroutine (denoted by the function “update”) that is no-regret with respect to a bounded loss function $l : \mathcal{A} \rightarrow [0, 1]$, maintains an internal distribution over items in \mathcal{A} for prediction and can be queried for multiple predictions (*i.e.* multiple samples). Algorithms that meet these requirements include Randomized Weighted Majority [Littlestone and Warmuth 1994], Follow-the-Leader [Kalai and Vempala 2005], EXP3 [Auer et al. 2003] and many others. In contrast to prior work [Streeter and Golovin 2008; Dey et al. 2012], SCP employs only a *single* no-regret online learning routine in the inner loop. The sample function samples the online learner’s internal distribution over items in \mathcal{A} to output an item a . The update function takes in a loss l_t and updates the internal distribution over items.

SCP proceeds by training over a sequence of examples $\mathbf{d} \sim \mathcal{D}$. At each iteration SCP queries the online learner to generate a list of N items (via $\text{sample}(p_t)$) which samples from its internal distribution over items p_t , evaluates a weighted cumulative gain of each item on the sampled list to define a loss related to each item and then uses the online learner (via update) to update its internal distribution.

During training we allow the algorithm to construct lists of length N rather than K . In its simplest form, one may simply choose $N = K$. However it may be beneficial to choose N differently than K , as is shown later in the theoretical analysis (Chapter 3.1).

Perhaps the most unusual aspect of Algorithm 10 is how the loss is defined using the weighted cumulative gains of each item:

$$r_t(a) \leftarrow \sum_{i=1}^N \left(1 - \frac{1}{K}\right)^{N-i} b(a|S_{t,i-1}, \mathbf{d}), \quad (3.1)$$

where $S_{t,i-1}$ denotes the first $i-1$ items in S_t and $b(a|S_{t,i-1}) = f(S_{t,i-1} \oplus a, \mathbf{d}) - f(S_{t,i-1}, \mathbf{d})$. Intuitively 3.1 represents the weighted sum of gains of item a in example \mathbf{d} had we added it at any intermediate position in S_t . The gains in different positions are weighed differently, where position i is adjusted by a factor $(1 - \frac{1}{K})^{N-i}$. These weights are derived via our theoretical analysis and indicate that benefits in early positions should be more discounted than benefits in later positions. Intuitively, this weighting has the effect of re-balancing the benefits so that each position contributes more equally to the overall loss.

In principle SCP and CONSEQOPT can be applied in partial feedback settings (*e.g.* advertisement placement) where the value of f is only observed for some items by using bandit algorithms instead (*e.g.* EXP3, [Auer et al. 2003]). As this is an orthogonal issue, we will focus here on the full information feedback case.

We now consider the contextual setting where features v of each example \mathbf{d} are observed

before choosing the list. Consider a hypothesis space Π which has uncountably many hypotheses. Conceivably the GREEDY algorithm can consider each such hypothesis and come up with the best ordering of these hypotheses (up to known approximation bounds). Since this is not feasible to do in practice due to the uncountably many hypotheses contained in Π , in Chapter 2.2 we leveraged optimization based techniques to modify GREEDY so that it could be “lifted” to the space of classifiers or regressors (hypotheses). The resulting algorithm CONSEQOPT could thus successfully compete against the approximately best list of hypotheses that the GREEDY algorithm could find if it could consider the uncountably many hypotheses in Π . Similarly our goal here is to compete against the best list of hypotheses $(\psi_1, \psi_2, \dots, \psi_N)$ from a hypothesis class Π . Each of these hypotheses are assumed to choose an item solely based on features of the example $\mathbf{d} \sim \mathcal{D}$.

We embed Π within a larger class $\Pi \subset \tilde{\Pi}$ where hypotheses in $\tilde{\Pi}$ are functions of both example and a partially constructed list. ($\tilde{\Pi} : \mathbb{R}^L \times \mathcal{S} \rightarrow \mathcal{A}$ where L is the size of the feature vector v representing example \mathbf{d} and \mathcal{S} is the space of all possible lists of items). Then any $\psi \in \tilde{\Pi}$, $(\psi(v, S))$, selects an item to append to S , given features of \mathbf{d} and features of list S . We will learn a hypothesis (or distribution of hypotheses) from $\tilde{\Pi}$ that attempts to generalize list construction across multiple positions of the list.¹

Algorithm 12 details an extension of SCP to the contextual setting. At each iteration, SCP constructs a list S_t for the example \mathbf{d} , using its current hypothesis or by sampling from its current distribution over hypotheses. Analogous to the context-free setting, we define a loss function over the learner subroutine “update”. We represent the loss using weighted cost-sensitive classification examples $\{(v_{ti}, c_{ti}, w_{ti})\}_{i=1}^N$, where v_{ti} denotes features of the example \mathbf{d} and list $S_{t,i-1}$, $w_{ti} = (1 - \frac{1}{K})^{N-i}$ is the weight associated to this example, and c_{ti} is the cost vector specifying the cost of each item $a \in \mathcal{A}$:

$$c_{ti}(a) = \max_{a' \in \mathcal{A}} b(a'|S_{t,i-1}, \mathbf{d}) - b(a|S_{t,i-1}, \mathbf{d}). \quad (3.2)$$

The loss incurred by any hypothesis ψ is defined by its loss on this set of cost-sensitive classification examples *i.e.*

$$l_t(\psi) = \sum_{i=1}^N w_{ti} c_{ti}(\psi(v_{ti})). \quad (3.3)$$

These new examples are then used to update the hypothesis (or distribution over hypotheses) using a no-regret online algorithm “update”.

This reduction effectively transforms the task of learning a hypothesis for this submodular

¹Competing against the best list of hypotheses in $\tilde{\Pi}$ is difficult in general as it violates submodularity: hypotheses can perform better when added later in the list (due to list features). Nevertheless, we can still learn from $\tilde{\Pi}$ and compete against the best list of hypotheses in Π .

Algorithm 12 SCP: Algorithm for training in contextual setting

Input: List length N ,

 List length of best list to compete against K ,

 Contextual no-regret online learning routine,

 Hypothesis class $\tilde{\Pi} : \mathbb{R}^L \times \mathcal{S} \rightarrow \mathcal{A}$,

 Library of items \mathcal{A}

Output: Hypothesis (or distribution over hypotheses) $\psi : \mathbb{R}^L \times \mathcal{S} \rightarrow \mathcal{A}$

```
1: for  $t = 1$  to  $T$  do
2:    $S_t = \{\}$ 
3:    $\mathbf{d} \leftarrow \text{sample}(\mathcal{D})$ 
4:   for  $i = 1$  to  $N$  do
5:      $v_{ti} \leftarrow \text{computeFeatures}(S_{t,i-1}, \mathbf{d})$ 
6:      $S_t \leftarrow S_t \oplus \psi_t(v_{ti})$ 
7:      $c_{ti} \leftarrow []$ 
8:     for  $j = 1$  to  $|\mathcal{A}|$  do
9:        $c_{tij} \leftarrow \max_{a' \in \mathcal{A}} b(a'|S_{t,i-1}, \mathbf{d}) - b(a_j|S_{t,i-1}, \mathbf{d})$ 
10:     $c_{ti} \leftarrow c_{ti} \oplus c_{tij}$ 
11:   end for
12:    $w_{ti} \leftarrow (1 - \frac{1}{K})^{N-i}$ 
13: end for
14:  $\psi_{t+1} \leftarrow \text{update}(\psi_t, \{(v_{ti}, c_{ti}, w_{ti})\}_{i=1}^N)$ 
15: end for
```

Algorithm 13 SCP: Algorithm for inference in contextual setting

Input: List length N ,

 Library of items \mathcal{A} ,

 Hypothesis (or distribution over hypotheses) $\psi : \mathbb{R}^L \times \mathcal{S} \rightarrow \mathcal{A}$

Output: List of selected items S

```
1:  $S = \{\}$ 
2: for  $i = 1$  to  $N$  do
3:    $v \leftarrow \text{computeFeatures}(S, \mathbf{d})$ 
4:    $a \leftarrow \psi(v)$ 
5:    $S \leftarrow S \oplus a$ 
6: end for
```

list optimization problem into a standard cost-sensitive classification problem². Analogous to the context-free setting, we can also extend to partial feedback setting where f is only partially measurable by using contextual bandit algorithms like EXP4 [Auer et al. 2003] as the online learner. Having transformed our problem into online cost-sensitive classification, we now present approaches that can be used to achieve no-regret on such tasks.

For finite policy classes $\tilde{\Pi}$, one can leverage any no-regret online algorithm such as Weighted Majority [Littlestone and Warmuth 1994]. Weighted Majority maintains a distribution over hypotheses in $\tilde{\Pi}$ based on the loss $l_t(\psi)$ for each ψ and achieves regret at a rate of

$$R = \sqrt{K' \frac{\log |\tilde{\Pi}|}{T}}, \quad (3.4)$$

for $K' = \min(N, K)$. In fact the context-free setting can be seen as a special case where $\Pi = \tilde{\Pi} = \{\psi | a \in \mathcal{A}\}$ and $\psi(v) = a$ for any v . However, achieving no-regret for uncountably many hypotheses classes is in general not tractable. A more practical approach is to employ existing reductions of cost-sensitive classification problems to convex optimization problems, for which we can efficiently run no-regret convex optimization (*e.g.* gradient descent). These reductions effectively upper bound the cost-sensitive loss by a convex loss, and thus bound the original loss of the list prediction problem. We briefly describe two such reductions from [Beygelzimer et al. 2005].

Reduction to Regression We transform cost-sensitive classification into a regression problem of predicting the cost of each item $a \in \mathcal{A}$. Afterwards, we choose the item with the lowest predicted cost. Analogous to CONSEQOPT using regressors in Algorithm 4, we convert each weighted cost-sensitive example (v_{ti}, c_{ti}, w_{ti}) into $|S|$ weighted regression examples. For example, if we use least-squares linear regression, the weighted squared-loss for a particular example (v_{ti}, c_{ti}, w_{ti}) and regressor \mathcal{R} would be

$$l(\mathcal{R}) = w \sum_{a \in \mathcal{A}} (\mathcal{R}^T v_{ti}(a) - c(a))^2. \quad (3.5)$$

Reduction to Ranking Another useful reduction transforms the problem into a “ranking” problem that penalizes ranking an item a above a better item a' . In our experiments, we employ a weighted hinge loss. The penalty is therefore proportional to the difference in cost of the mis-ranked pair. For each cost sensitive example (v_{ti}, c_{ti}, w_{ti}) we generate $|\mathcal{A}|(|\mathcal{A}| - 1)/2$ ranking examples for every distinct pair of items (a, a') where we must predict the best item between (a, a') (potentially by a margin) with a weight $w_{ti}|c_{ti}(a) - c_{ti}(a')|$. For example if

²This is similar to DAGGER [Ross et al. 2011] developed for sequential prediction problems like imitation learning can be seen as a specialization of DAGGER for submodular list optimization and ensures that we learn learners that pick good items under the lists they construct.

we train a linear SVM [Joachims 2005], we obtain a weighted hinge loss of the form:

$$w_{ti}|c_{ti}(a) - c_{ti}(a')| \max(0, 1 - h^T(v_{ti}(a) - v_{ti}(a'))) \operatorname{sign}|c_{ti}(a) - c_{ti}(a')|, \quad (3.6)$$

where h is the linear hypothesis. At prediction time, we simply predict the item $a^* = \operatorname{argmax}_{a \in \mathcal{A}} h^T v_{ti}(a)$. This reduction proves advantageous whenever it is easier to predict pairwise rankings rather than the actual cost.

3.1 SCP Analysis

We now show that Algorithm 10 is no-regret with respect to the GREEDY algorithm's expected performance over the training instances. Our main theoretical result provides a reduction to an online learning problem and directly relates the performance of our algorithm on the submodular list optimization problem to the standard online learning regret incurred by the routine. Although Algorithm 10 uses only a *single* instance of an online learner routine it achieves the same performance guarantee as prior work [Streeter and Golovin 2008] and CONSEQOPT that employ N separate instances of an online learner. This leads to a surprising fact: it is possible to sample from a stationary distribution over items to construct a list that achieves the same guarantee as the GREEDY algorithm!

For a sequence of training examples $\{d_t\}_{t=1}^T$, let the sequence of loss functions $\{l_t\}_{t=1}^T$ defined in Algorithm 10 correspond to the sequence of losses incurred in the reduction to the online learning problem. The expected regret of the online learning algorithm is:

$$\mathbb{E}[R] = \sum_{t=1}^T \mathbb{E}_{a' \sim p_t}[l_t(a')] - \min_{a \in \mathcal{A}} \sum_{t=1}^T l_t(a), \quad (3.7)$$

where p_t is the internal distribution of the online learner used to construct list S_t . Note that an online learner is called no-regret if R is sublinear in T .

Let $F(p, N) = \mathbb{E}_{S_N \sim p}[\mathbb{E}_{\mathbf{d} \sim \mathcal{D}}[f(S_N, \mathbf{d})]]$ denote the expected value of constructing lists by sampling (with replacement) N elements from distribution p , and let $\hat{p} = \operatorname{argmax}_{t \in \{1, 2, \dots, T\}} F(p_t, N)$ denote the best distribution found by the algorithm. We define a mixture distribution \bar{p} over lists that constructs a list as follows: sample an index t uniformly in $\{1, 2, \dots, T\}$, then sample N elements (with replacement) from p_t . Note that $F(\bar{p}, N) = \frac{1}{T} \sum_{t=1}^T F(p_t, N)$ and $F(\hat{p}, N) \geq F(\bar{p}, N)$. Thus it suffices to show that $F(\bar{p}, N)$ has good guarantees. We show that in expectation \bar{p} (and thus \hat{p}) constructs lists with performance guarantees close to GREEDY.

Theorem 3. Let $\alpha = \exp(-\frac{N}{K})$ and $K' = \min(N, K)$. For any $\delta \in (0, 1)$, with probability $\geq 1 - \delta$

$$F(\bar{p}, N) \geq (1 - \alpha)F(S_K^*) - \frac{\mathbb{E}[R]}{T} - 3\sqrt{\frac{2K'\ln(2/\delta)}{T}}, \quad (3.8)$$

where S_K^* is the optimal list of length K .

Proof. See Appendix A and A.2. \square

Corollary 1. If a no-regret algorithm is used on the sequence of losses l_t , then as $T \rightarrow \infty$, $\frac{\mathbb{E}[R]}{T} \rightarrow 0$ and

$$\lim_{T \rightarrow \infty} F(\bar{p}, N) \geq (1 - \alpha)F(S_K^*). \quad (3.9)$$

Proof. See Appendix A and A.2. \square

Theorem 3 provides a general approximation ratio to the best list of size K , when constructing a list of a different size N . For $N = K$, we obtain the typical $(1 - \frac{1}{e})$ approximation ratio [Feige 1998]. As K increases, this provides approximation ratios that converge exponentially to 1. Naively one might expect regret $\mathbb{E}[R]/T$ to scale linearly in K' as it involves loss in $[0, K']$. However we show that regret actually scales as $O\sqrt{K'}$ (e.g. using Weighted Majority [Littlestone and Warmuth 1994]). Our result matches the best known results for this setting [Streeter and Golovin 2008], while using a *single* online learner, and is especially beneficial in the contextual setting due to improved generalization.

Corollary 2. Using Weighted Majority [Littlestone and Warmuth 1994] with the optimal learning rate guarantees with probability $\geq 1 - \delta$

$$F(\bar{p}, N) \geq (1 - \alpha)F(S_K^*) - O\left(\sqrt{\frac{K' \log(1/\delta)}{T}}\right) + \sqrt{\frac{K' \log |\mathcal{A}|}{T}}. \quad (3.10)$$

Proof. See Appendix A and A.2. \square

We now present performance guarantees for SCP in the contextual setting, that relate performance on the submodular list optimization task to the regret of the corresponding online cost-sensitive classification task. Let $l_t : \tilde{\Pi} \rightarrow \mathbb{R}$ compute the loss of each hypothesis ψ on the cost-sensitive classification examples $\{(v_{ti}, c_{ti}, w_{ti})\}_{i=1}^N$ collected in Algorithm 12 for example d. We use $\{l_t\}_{t=1}^T$ as the sequence of losses for the online learning problem.

³Additionally, if the distributions p_t converge, then the last distribution p_{T+1} must have performance arbitrarily close to \bar{p} as $T \rightarrow \infty$. In particular, we can expect this to occur when the examples are randomly drawn from a fixed distribution that does not change over time.

For a deterministic online algorithm that picks the sequence of hypotheses $\{\psi_t\}_{t=1}^T$, the regret is:

$$R = \sum_{t=1}^T l_t(\psi_t) - \min_{\psi \in \Pi} \sum_{t=1}^T l_t(\psi). \quad (3.11)$$

For a randomized online learner, let ψ_t be the distribution over hypotheses at iteration t , with expected regret:

$$\mathbb{E}[R] = \sum_{t=1}^T \mathbb{E}_{\psi'_t \sim \psi_t} [l_t(\psi'_t)] - \min_{\psi \in \Pi} \sum_{t=1}^T l_t(\psi). \quad (3.12)$$

Let:

$$F(\psi, N) = \mathbb{E}_{S_{\psi, N} \sim \psi} [\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, N}, \mathbf{d})]], \quad (3.13)$$

denote the expected value of constructing lists $(S_{\psi, N})$ by sampling (with replacement) N hypotheses from hypotheses distribution ψ (if ψ is a deterministic hypothesis, then this means we use the same hypothesis at each position of the list). Let $\hat{\psi} = \operatorname{argmax}_{t=\{1, 2, \dots, T\}} F(\psi_t, N)$ denote the best distribution found by the algorithm in hindsight.

We use a mixture distribution $\bar{\psi}$ over hypotheses to construct a list as follows: sample an index t uniformly in $\{1, 2, \dots, T\}$, then sample N learners from ψ_t to construct the list. As before, we note that $F(\bar{\psi}, N) = \frac{1}{T} \sum_{t=1}^T F(\psi_t, N)$ and $F(\hat{\psi}, N) \geq F(\bar{\psi}, N)$. We again focus on providing good guarantees for $F(\bar{\psi}, N)$ as shown by the following theorem:

Theorem 4. *Let $\alpha = \exp(-\frac{N}{K})$ and $K' = \min(N, K)$. For any $\delta \in (0, 1)$. After T iterations, for deterministic online algorithms, with probability $\geq 1 - \delta$:*

$$F(\bar{\psi}, N) \geq (1 - \alpha)F(S_{\psi, K}^*) - \frac{R}{T} - 2\sqrt{\frac{2 \ln(1/\delta)}{T}}, \quad (3.14)$$

where $S_{\psi, K}^*$ is the list of length K that can be constructed by the best distribution ψ over hypotheses. Similarly, for randomized online algorithms, with probability at least $1 - \delta$:

$$F(\bar{\psi}, N) \geq (1 - \alpha)F(S_{\psi, K}^*) - \frac{\mathbb{E}[R]}{T} - 3\sqrt{\frac{2K' \ln(2/\delta)}{T}}, \quad (3.15)$$

Proof. See Appendix A and A.2. □

Thus as in the context-free case, a no-regret online algorithm must achieve $F(\bar{\psi}, N) \geq (1 - \alpha)F(S_{\psi, K}^*)$ with high probability as $T \rightarrow \infty$. This matches similar guarantees provided by CONSEQOPT. Despite having similar guarantees, we intuitively expect SCP to outperform CONSEQOPT in practice because SCP can use all data to train a single predictor, instead

of being split to train K separate ones. We empirically verify this intuition in Chapter 3.2. When using surrogate convex loss functions (such as regression or ranking loss), we provide a general result that applies if the online learner uses any convex upper bound of the cost-sensitive loss. An extra penalty term is introduced that relates the gap between the convex upper bound and the original cost-sensitive loss:

Theorem 5. *Let $\alpha = \exp(-\frac{N}{K})$ and $K' = \min(N, K)$. If we run an online algorithm on the sequence of convex losses C_t instead of l_t , then after T iterations, for any $\delta \in (0, 1)$, we have that with probability at least $1 - \delta$:*

$$F(\bar{\psi}, N) \geq (1 - \alpha)F(S_{\psi, K}^*) - \frac{\tilde{R}}{T} - 2\sqrt{\frac{2\ln(1/\delta)}{T}} - \mathcal{G}. \quad (3.16)$$

Proof. See Appendix A and A.2. □

This result implies that using a good surrogate convex loss for no-regret convex optimization will lead to a learner (or distribution of learners) that has good performance relative to the optimal list of learners. Note that the gap \mathcal{G} , often may be small or non-existent. For instance, in the case of the reduction to regression or ranking, $\mathcal{G} = 0$ in realizable settings where there exists a “perfect” hypothesis in the class. Similarly where the problem is near-realizable we would expect \mathcal{G} to be small.⁴

3.2 Case Studies

3.2.1 Case Study: Robotic Manipulation Planning

We applied SCP to the robot manipulation planning task used to showcase CONSEQOPT in Chapter 2.1. The goal is to predict a set of initial trajectories so as to maximize the chance that one of them leads to a collision-free trajectory. We use local trajectory optimization techniques such as CHOMP [Zucker et al. 2013], which have proven effective in quickly finding collision-free trajectories using local perturbations of an initial trajectory. Note that selecting a diverse set of initial trajectories is important since local techniques such as CHOMP often get stuck in local optima.⁵

We use the same dataset as used for CONSEQOPT. It consists of 310 training and 212 test environments of random obstacle configurations around a target object, and 30 initial seed trajectories. In each environment, each seed trajectory has 17 features describing the

⁴We conjecture that this gap term \mathcal{G} is not specific to our particular scenario, but rather is (implicitly) always present whenever one attempts to optimize classification accuracy via surrogate convex optimization.

⁵*i.e.* similar or redundant initial trajectories will lead to the same local optima.

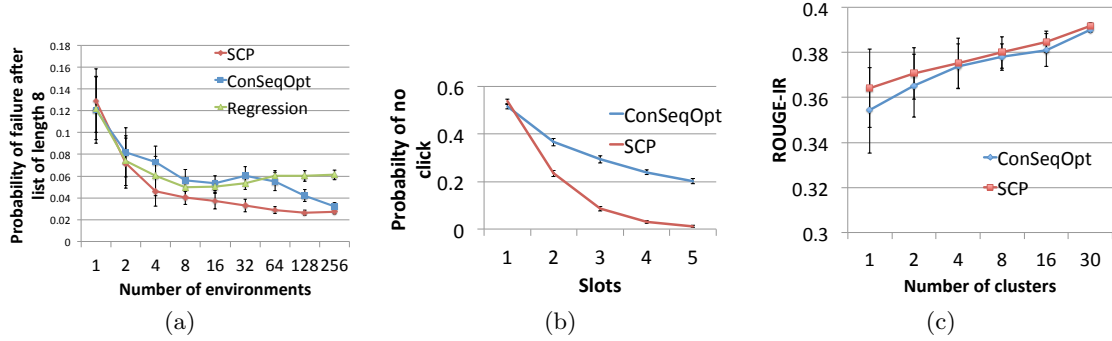


Figure 3.1: (a) SCP performs better at even low data availability while CONSEQOPT suffers from data starvation issues (b) With increase in slots SCP predicts news articles which have lower probability of the user not clicking on any of them compared to CONSEQOPT (c) ROUGE-1R scores with respect to the size of the training data

spatial properties of the trajectory relative to obstacles. In addition to the base features, we add features of the current list with respect to each initial trajectory. We use the per feature minimum absolute distance and average absolute value of the distance to the features of initial trajectories in the list. We also use a bias feature always set to 1, and an indicator feature which is 1 when selecting the element in the first position, 0 otherwise. This enables a distinction between the case where the minimum and average features are 0 because there are no trajectories in the list yet, versus when they are 0 because we are actually considering a trajectory which is already in the list.

We compare SCP to CONSEQOPT (which learns N separate predictors), and Regression (regress success rate from features to sort initial trajectories; this accounts for relevance but not diversity).

Figure 3.1 (left) shows the failure probability over the test environments versus the number of training environments. CONSEQOPT employs a reduction to N classifiers. As a consequence, CONSEQOPT faces data starvation issues for small training sizes, as there is little data available for training predictors lower in the list.⁶ In contrast, SCP has no data starvation issue and outperforms both CONSEQOPT and Regression.

3.2.2 Case Study: Personalized News Recommendation

In the news recommendation setting the task is to present a sequence of news articles to a user so as to maximize the probability of the user clicking on at least 1 recommended article, which is similar to the initial trajectory selection problem in manipulation.

⁶When a successful initial trajectory is found, benefits at later positions are 0. This effectively discards training environments for training classifiers lower in the list in CONSEQOPT.

We built a stochastic user simulation based on 75 user preferences derived from a user study in [Yue and Guestrin 2011]. Using this simulation as a training oracle, our goal is to learn to recommend articles to any user (depending on their contextual features) to minimize the failure case where the user does not like any of the recommendations.⁷

Articles are represented by features, and user preferences by linear weights. We derived user contexts by soft-clustering users into groups, and using corrupted group memberships as contexts.

We perform five-fold cross validation. In each fold, we train SCP and CONSEQOPT on 40 users’ preferences, use 20 users for validation, and then test on the held-out 15 users. Training, validation and testing are all performed via simulation. Figure 3.1 (middle) shows the results, where we see the recommendations made by SCP achieves significantly lower failure rate as the number of recommendations is increased from 1 to 5.

3.2.3 Case Study: Document Summarization

Method	ROUGE-1F	ROUGE-1P	ROUGE-1R
SubMod	37.39	36.86	37.99
DPP	38.27	37.87	38.71
ConSeqOpt	39.02 \pm 0.07	39.08 \pm 0.07	39.00 \pm 0.12
SCP	39.15\pm0.15	39.16\pm0.15	39.17\pm0.15
Greedy (Oracle)	44.92	45.14	45.24

Table 3.1: ROUGE unigram score on the DUC 2004 test set

In the extractive multi-document summarization task, the goal is to extract sentences (with character budget N) to maximize coverage of human-annotated summaries.

Following the experimental setup from [Lin and Bilmes 2010] and [Kulesza and Taskar 2011], we use data from the Document Understanding Conference (DUC) 2003 and 2004 (Task 2) [Dang 2005]. Each training or test instance corresponds to a cluster of documents, and contains approximately 10 documents belonging to the same topic and four human reference summaries. We train on the 2003 data (30 clusters) and test on the 2004 data (50 clusters). The budget is $N = 665$ bytes, including spaces.

We use the ROUGE [Lin 2004] unigram statistics (ROUGE-1R, ROUGE-1P, ROUGE-1F) for performance evaluation. Our method directly attempts to optimize the ROUGE-1R

⁷Also known as abandonment [Radlinski et al. 2008].

objective with respect to the reference summaries, which can be easily shown to be monotone submodular [Lin and Bilmes 2011].

We aim to predict sentences that are both short and informative. Therefore we maximize the normalized marginal benefit,

$$b'(a|S_{t,i-1}) = b(a|S_{t,i-1})/l(a), \quad (3.17)$$

where $l(a)$ is the length of the sentence a .⁸ We use a reduction to ranking as described in Chapter 3 using (3.17). While not performance-optimized, our approach takes less than 15 minutes to train.

Following [Kulesza and Taskar 2011], we consider features \mathbf{x}_i for each sentence consisting of *quality features* q_i and *similarity features* ϕ_i ($\mathbf{x}_i = [q_i^T, \phi_i^T]^T$). The quality features, attempt to capture the representativeness for a single sentence. Similarity features ϕ_i for sentence a_i as we construct the list S_t measure a notion of distance of a proposed sentence to sentences already included in the set.⁹

Table 3.1 shows the performance (Rouge unigram statistics) comparing SCP with existing algorithms. We observe that SCP outperforms existing state-of-the-art approaches, which we denote SubMod [Lin and Bilmes 2010] and DPP [Kulesza and Taskar 2011]. “Greedy (Oracle)” corresponds to the clairvoyant oracle which uses the GREEDY algorithm 9 to directly optimize the test Rouge score and thus serves as an upper bound on this class of techniques. Figure 3.1 (right) plots Rouge-1R performance as a function of the size of training data, suggesting SCP’s superior data-efficiency compared to CONSEQOPT.

⁸This results in a knapsack constrained optimization problem. We refer the reader to [Zhou et al. 2013] for a detailed analysis.

⁹A variety of similarity features were considered, with the simplest being average squared distance of tf-idf vectors. Performance was very stable across different features. The experiments presented use three types: 1) following the idea in [Kulesza and Taskar 2011] of similarity as a volume metric, we compute the squared volume of the parallelepiped spanned by the TF-IDF vectors of sentences in the set $S_{t,k} \cup a_i$; 2) the product between $\det(G_{S_{t,k} \cup a_i})$ and the quality features; 3) the minimum absolute distance of quality features between a_i and each element in $S_{t,k}$.

Multiple Output Structured Prediction

In previous chapters (Chapter 2 and 3) we have detailed a method for directly predicting sets and lists that maximize monotone submodular objectives in general settings. In this chapter we propose extensions to the structured output setting. Such tasks are ubiquitous in computer vision tasks such as object recognition [Girshick et al. 2014], semantic segmentation [Carreira and Sminchisescu 2010], tracking [Kalal et al. 2012], monocular human pose estimation [Yang and Ramanan 2011] and point cloud classification [Munoz et al. 2010]. These tasks are often addressed by a pipeline architecture where each module of the pipeline produces several hypotheses as input to the next module. Considering multiple options at each stage is good practice as it avoids premature commitment to a single answer which, if wrong, can jeopardize the quality of decisions made downstream [Felzenszwalb and McAllester 2007; Viola and Jones 2001]. As an example consider Figure 4.1 where multiple predictions are generated for a foreground/background segmentation task. We see that the prediction with the highest confidence (denoted by prediction 1) can be far from the groundtruth. The principal requirement of a list is that at least one hypothesis in the list is close to the groundtruth labeling (high list recall). A characteristic of lists which achieve high recall in a small number of hypotheses is diversity [Radlinski et al. 2008] which increases the odds of at least one accurate prediction.

Our central insight is that diversity in a list of structured predictions need not be enforced, but that it is an emergent property of optimizing the correct submodular recall objective. Similar to CONSEQOPT our procedure trains a sequence of predictors, each of which produces a hypothesis. Consider the semantic scene labeling problem where the task is to label every pixel with a semantic label like “grass”, “sky”, etc. It is not beneficial to predict a labeling in the second position of the list which differs from the first labeling in the list only by a few pixels. Note that the desired property is to achieve high recall, while diversity is merely a characteristic of lists that achieve this. Therefore our objective optimizes for recall and does not explicitly enforce diversity, instead the maximization of the submodular objective

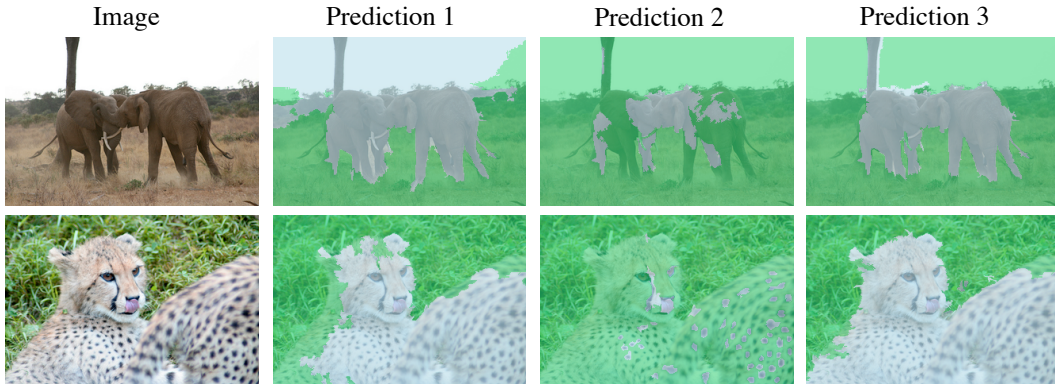


Figure 4.1: For a given image our method trains a small number of structured predictors in sequence. For a test image, the list of predictors are invoked to produce multiple hypotheses. Our approach produces high recall lists within a small number of hypotheses and can use any structured predictor available.

naturally produces diverse hypotheses. Conveniently, as summarized in Chapter 2, submodular monotone functions can be maximized efficiently by greedily maximizing the marginal benefit which ensures performance within $1 - \frac{1}{e}$ ($\sim 63\%$) of the optimal list of items of a fixed length [Nemhauser et al. 1978].

Making a single best prediction in structured problems is difficult due to the combinatorially large state space that has to be considered. While a number of approaches, both probabilistic [Lafferty et al. 2001; Kohli et al. 2013] and margin-based [Tsochantaridis et al. 2005; Taskar et al. 2003], for learning and inference in structured problems are well known, methods for making multiple predictions in structured problem domains are relatively few [Guzman-Rivera et al. 2012; Batra et al. 2012; Park and Ramanan 2011; Kulesza and Taskar 2010]. We develop a learning-based approach to produce a small list of structured predictions that ensures high recall in a variety of computer vision tasks.

In contrast to recent developments which train a single model during the learning phase and modify the inference procedure to produce multiple hypotheses at test time [Batra et al. 2012; Park and Ramanan 2011; Kulesza and Taskar 2010], our approach trains separate predictors during the learning phase to produce each of the hypotheses in the list. This alternate approach has several advantages—the learning procedure is optimized for the task of producing a list with high recall; diversity does not need to be enforced in an ad hoc fashion but is an emergent property of lists that maximize our objective; it is agnostic to the inference method used and can be utilized for any class of structured predictor. We empirically demonstrate our approach on common vision tasks such as estimating human pose from a single image, semantic scene segmentation and foreground/background segmentation.

The primary contributions of our approach are:

- Our approach is a *model agnostic* framework applicable to extending any structured prediction algorithm to make multiple predictions. In any task domain for which learning algorithms exist to generate a single best prediction, our approach can be employed for making multiple predictions by training multiple instances.
- Our approach is *parameter free*. In contrast, current state-of-the-art approaches enforce diversity by explicitly introducing a diversity modeling term in the objective function. Such parameters are tuned on validation data. It is not clear that artificially enforcing diversity in such a way is the right thing for the task at hand to achieve the best performance [Caruana et al. 2004; Misra et al. 2014].
- We study the empirical performance of our approach and demonstrate state-of-the-art results on multiple predictions for monocular pose estimation and foreground/background segmentation on benchmark datasets.

4.1 Related Work

Kulesza et al. [Kulesza and Taskar 2011] have adapted determinantal point processes (DPP), a model used in particle physics for optimizing for diverse but low error predictions. DPPs are especially attractive because they allow for efficient, exact inference procedures and are similar to monotone, submodular optimization methods.

The related work in multiple structured prediction can be grouped into two categories: 1) The first are methods which are *model-dependent*. These methods are tied to the specific learning and inference procedure being used (e.g. S-SVM, CRF) and cannot easily be adapted to different structured prediction methods. 2) The second category of models are *model-agnostic*, which are not tied to the specifics of the chosen structured prediction method.

Model-dependent methods: Batra et al. [Batra et al. 2012] deal with the problem of inferring low error and diverse solutions from a Markov Random Field (MRF). They approach this problem by introducing a constraint in the MRF objective function which says that a new solution must be at least some distance away from each of the previous solutions. The constraint is moved to the objective by a Lagrangian multiplier λ and then solved using a supergradient algorithm. λ is treated as a free parameter and is optimized over a validation set. They term this approach as DIVMBEST. Note that a single model is initially learnt (the MRF) and only during inference time diverse solutions are obtained by imposing constraints on the inference procedure. Nilsson et al. [Nilsson 1998] and Weiss et al. [Yanover and Weiss 2004] propose methods for using loopy belief propagation for finding the most

probably solutions at inference time in graphical models. But their methods don't try to incorporate diversity to improve performance. Park and Ramanan [Park and Ramanan 2011] use a modified version of standard max-product inference which aims to enforce diversity by incorporating part-overlap constraints, which they term as NBEST.

In the approach proposed by Guzman-Rivera et al. [Guzman-Rivera et al. 2012], a structured SVM [Tschantz et al. 2005] (S-SVM) is trained for each position of the list. During inference time, each S-SVM is invoked to predict a structured output. They minimize an upper bound of the non-convex, structured hinge loss via a kmeans-based initialization step and an expectation-maximization (EM) style coordinate-descent minimization algorithm. They term their approach as Multiple Choice Learning (MCL). In more recent work, Guzman-Rivera et al. [Guzman-Rivera et al. 2014a] explicitly add diversity to the MCL objective and optimize a surrogate via an EM style block coordinate-descent minimization routine similar to MCL. An extra parameter which trades off between diversity and accuracy is then tuned via cross-validation. They term this approach as Diverse Multiple Choice Learning (DivMCL).

In comparison to such model-dependent methods, our proposed method is model-agnostic and can use *any* structured prediction approach.

Model-agnostic methods: To the best of our knowledge, the only such method is the ad hoc boosting-like weighting scheme used in [Guzman-Rivera et al. 2012, 2014b] which we denote henceforth as GR14. The weighting scheme of GR14 [Guzman-Rivera et al. 2014b] has been used for specific task of camera re-localization. This method has a free parameter which must be tuned on validation data. In comparison our approach is parameter free and achieves comparable or better results on standard vision tasks.

4.2 Approach

Structured prediction problems in machine learning and computer vision are characterized by a multidimensional *structured* output space \mathcal{Y} , where the notion of *structure* varies according to the problem. For example, in semantic scene understanding, the structure in the output $\mathbf{y} \in \mathcal{Y}$ refers to the fact that nearby regions in the image tend to have correlated semantic labels. In human pose estimation from images, the location of a limb in the image is correlated with the locations of other limbs.

One possible approach to structured predictions could be to use the well understood approach of multi-class classification by treating each possible structured output as a label. If this were possible, multiple low error and diverse interpretations could be directly generated using a scheme such as CONSEQOPT [Dey et al. 2013], described in Chapter 2. However,

the challenge in such structured prediction tasks is that the space of possible output variable combinations is exponential in the number of labels for each variable. For example for an image with 10^4 pixels and 21 possible labels for each pixel there are 21^{10^4} possible labelings. This is also why structured prediction tasks cannot be addressed by multi-class classification as the number of classes is exponentially large. As a result, directly applying a procedure such as CONSEQOPT to generate multiple interpretations is infeasible.

Our approach is inspired by the ideas set forth in CONSEQOPT and SCP. We define a monotone submodular function over a list of structured predictors and show that a simple greedy algorithm can be used to train a list of such predictors to produce a set of structured predictions with high recall. More formally our problem can be stated as follows.

Problem Statement: The goal of our approach is, given an input image $I \in \mathcal{I}$, to produce a list of N structured outputs $Y(I) = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\} \in \mathcal{Y}$ with low error and high recall. We formulate this as the problem of learning a list of structured predictors $S = \{h_1, h_2, \dots, h_N\}$ where each predictor $h_i : \mathcal{I} \rightarrow \mathcal{Y}, h_i \in \mathcal{H}$, in the list produces the corresponding structured output \mathbf{y}_i , where \mathcal{H} is a hypothesis class of structured predictors and \mathcal{Y} is the space of structured predictions.

We begin by describing a submodular objective that captures the notion of low error and high recall. Let us denote j^{th} training sample as a tuple $\{(I^j, \mathbf{y}_{gt}^j)\}_{j \in 1 \dots |\mathcal{D}|}$, where for each image I^j , the ground truth structured label is denoted by \mathbf{y}_{gt}^j . We denote by $l : \mathcal{Y} \times \mathcal{Y} \rightarrow [0, 1]$, a *loss* function that measures the disagreement between the predicted structured output \mathbf{y} and the ground truth structured label \mathbf{y}_{gt} . The corresponding function measuring *gain* is thus given by $g(\mathbf{y}, \mathbf{y}_{gt}) = 1 - l(\mathbf{y}, \mathbf{y}_{gt})$. We define a list of structured outputs as:

$$Y_S(I) = \{h_i(I)\}_{i \in 1 \dots N}. \quad (4.1)$$

We then define the quality function,

$$f(Y_S(I), \mathbf{y}_{gt}) = \max_{i \in 1, \dots, N} \{g(h_i(I), \mathbf{y}_{gt})\}, \quad (4.2)$$

$$= 1 - \min_{i \in 1, \dots, N} \{l(h_i(I), \mathbf{y}_{gt})\} \quad (4.3)$$

that scores the list of structured predictions $Y_S(I)$ by the score of the best prediction produced by the list of predictors $S = \{h_1, h_2, \dots, h_N\}$. We note that to maximize this scoring function with respect to the list of predictions at least one of the predictions \mathbf{y}_i in the list needs to be close to the ground truth. In order to learn a list of predictors that works well across a distribution of the data, the objective function we would like to optimize is the expected

value of the above function over the distribution of the data \mathcal{D} :

$$F(S, \mathcal{D}) = \mathbb{E}_{(I, \mathbf{y}_{gt}) \sim \mathcal{D}} [f(Y_S(I), \mathbf{y}_{gt})]. \quad (4.4)$$

The resulting optimization problem is therefore to find the list of predictors S that maximizes the objective function F in Equation 4.4 and can be written as follows:

$$\max_S \mathbb{E}_{(I, \mathbf{y}_{gt}) \sim \mathcal{D}} [f(Y_S(I), \mathbf{y}_{gt})]. \quad (4.5)$$

The function F of the form in Equation 4.4 can be shown to be a monotone submodular function over lists of input items as shown in Appendix B, [Dey et al. 2012]. The natural approach for submodular optimization problems of the form in 4.5 is to use a greedy algorithm [Nemhauser et al. 1978]. In each greedy step i , we add the structured predictor \tilde{h}_i^* , that maximizes the marginal benefit. For our objective, maximizing the marginal benefit is written as:

$$\tilde{h}_i^* = \arg \max_{h \in \mathcal{H}} F(S_{i-1} \oplus \{h\}, \mathcal{D}) - F(S_{i-1}, \mathcal{D}). \quad (4.6)$$

Maximizing the marginal benefit, as written above, over the space of structured predictors by enumeration is difficult, because there can be uncountably many such predictors. *Instead, we take the approach of directly training a structured predictor to maximize the marginal benefit.* As we do not have access to the true distribution of the data, we maximize the marginal benefit using the empirical distribution $\tilde{\mathcal{D}}$. We denote the loss $l_i^j = l(\mathbf{y}_i^j, \mathbf{y}_{gt}^j)$ as shorthand for the loss of the i^{th} predictor on the j^{th} training sample. Rewriting the objective with respect to the empirical data distribution and in terms of the loss per example we have,

$$\begin{aligned} & F(S_{i-1} \oplus \tilde{h}, \tilde{\mathcal{D}}) - F(S_{i-1}, \tilde{\mathcal{D}}) \\ &= \sum_{j \in \tilde{\mathcal{D}}} \left(\min \{l_1^j, \dots, l_{i-1}^j\} - \min \{l_1^j, \dots, l_i^j\} \right), \end{aligned} \quad (4.7)$$

$$= \sum_{j \in \tilde{\mathcal{D}}} \max \left(\min \{l_1^j, \dots, l_{i-1}^j\} - l_i^j, 0 \right), \quad (4.8)$$

$$= \sum_{j \in \tilde{\mathcal{D}}} \max \left(\xi_{i-1}^j - l_i^j, 0 \right). \quad (4.9)$$

where ξ_{i-1}^j in 4.9 is the minimum loss obtained by the list of $i-1$ predictors on j^{th} sample till now. Since training procedures for structured predictors are usually implemented to minimize loss we rewrite 4.9 and 4.6 as:

Algorithm 14 SEQNBEST: Algorithm for training

Input: List length N ,
Structured prediction routine $\tilde{h} \in \mathcal{H}$,
Dataset \mathcal{D} of $|\mathcal{D}|$ examples,

- 1: $S = \{\}, \{\mathbf{w}_0^j = 1\}_{j \in 1 \dots |\tilde{\mathcal{D}}|}$
- 2: **for** $i = 1$ **to** N **do**
- 3: $\tilde{h}_i = \text{trainStructuredPredictor}(\tilde{\mathcal{D}}, \mathbf{w}_{i-1})$
- 4: $S \leftarrow S \oplus \tilde{h}_i$
- 5: $\mathbf{w}_i = \text{computeMarginalWeights}(S, \tilde{\mathcal{D}})$
- 6: **end for**
- 7: **Return:** $S = \{\tilde{h}_1, \tilde{h}_2, \dots, \tilde{h}_N\}$

Algorithm 15 computeMarginalWeights ($S, \tilde{\mathcal{D}}$)

Input: List of trained structured predictors S ,
Dataset $\tilde{\mathcal{D}}$

- 1: **for** $j = 1$ **to** $|\tilde{\mathcal{D}}|$ **do**
- 2: $l = \{\}$
- 3: **for** $i = 1$ **to** $|S|$ **do**
- 4: $l_i = l(\tilde{h}_i(I^j), \mathbf{y}_{gt}^j)$
- 5: $l \leftarrow l \oplus l_i$
- 6: **end for**
- 7: $\xi \leftarrow \min(l)$
- 8: **if** SEQNBEST1 **then**
- 9: $w^j \leftarrow \xi$
- 10: **else if** SEQNBEST2 **then**
- 11: $w^j \leftarrow \xi^3 / (3\xi^2 - 3\xi + 1)$
- 12: **end if**
- 13: **end for**
- 14: **Return:** $\mathbf{w} = \{w^1, w^2, \dots, w^{|\tilde{\mathcal{D}}|}\}$

Algorithm 16 SEQNBEST: Algorithm for inference

Input: Trained list of classifiers $\tilde{h}_1, \tilde{h}_2, \dots, \tilde{h}_N$,
Test example I

Output: List of structured predictions S

- 1: $S = \{\}$
- 2: **for** $i = 1$ **to** N **do**
- 3: $\mathbf{y} \leftarrow \tilde{h}_i(I)$
- 4: $S \leftarrow S \oplus \mathbf{y}$
- 5: **end for**

$$h_i^* = \arg \max_{h \in \mathcal{H}} \sum_{j \in \tilde{\mathcal{D}}} \max \left(\xi_{i-1}^j - l_i^j, 0 \right), \quad (4.10)$$

$$= \arg \min_{h \in \mathcal{H}} \sum_{j \in \tilde{\mathcal{D}}} \min \left(l_i^j - \xi_{i-1}^j, 0 \right) \quad (4.11)$$

Let us denote the per-example desired loss $l_{\text{Actual}} = \min \left(l_i^j - \xi_{i-1}^j, 0 \right)$ which is the summand in Equation 4.10. Consider the relationship of the loss l_{Actual} as a function of the loss of the current predictor l_i^j and the best loss seen before the current predictor (ξ_{i-1}^j). This is drawn in Figure 4.2a and denoted by the line l_{Actual} . We observe that if a predictor obtains a loss greater than the previous best, ξ_{i-1}^j on an example it does not contribute towards lowering of the loss defined in Equation 4.10. Whereas if it achieves loss less than ξ_{i-1}^j , it lowers the objective by the same amount that it is less than ξ_{i-1}^j . Optimizing such a loss directly tends to be difficult as it can require modifications that are specific to the structured predictor’s training procedure. Instead, we take the approach of optimizing a tight linear upper bound of the loss ($l_{\text{SeqNBest1}}$ in Figure 4.2a) which results in a procedure that only requires re-weighting the training data and is model-agnostic. Consider a linear upper bound on l_{Actual} defined by the parameter w_i^j ,

$$w_i^j l_i^j \geq l_{\text{Actual}}. \quad (4.12)$$

Training a predictor which optimizes the surrogate loss on the left hand side of 4.12 is equivalent to training a structured predictor which weights each data sample with the weight w_i^j :

$$\sum_{j \in \tilde{\mathcal{D}}} w_i^j l_i^j \geq \sum_{j \in \tilde{\mathcal{D}}} \min \left(l_i^j - \xi_{i-1}^j, 0 \right). \quad (4.13)$$

Note that by setting the weight of each sample to be proportional to the marginal benefit left ($w_i^j \propto \xi_{i-1}^j$) we are *minimizing a tight linear upper bound* of the actual loss function we wish to minimize (l_{Actual}). This relationship is indicated by the line $l_{\text{SeqNBest1}}$ in Figure 4.2b. Our training procedure might be reminiscent of boosting [Freund et al. 1999] where several predictors are combined to produce a single output. In contrast, our procedure is trained to produce a list of predictors each of which makes a separate prediction in a list of predictions. Additionally we are also optimizing a completely unrelated loss function.

An alternative tight linear upper bound can be calculated by minimizing the L_2 norm

between l_{Actual} and a linear loss function given by $w_i^j l_i^j$. Consider a family of linear upper bounds of the quality function l_{Actual} which has the form

$$l_{\text{SeqNBest2}} = w_i^j l_i^j + b. \quad (4.14)$$

Note that to achieve a tight upper bound, candidate lines must pass through the point $(\xi_{i-1}^j, 0)$. Substituting $(\xi_{i-1}^j, 0)$ in Equation 15 we get

$$0 = a\xi_{i-1}^j + b, \quad (4.15)$$

$$b = -a\xi_{i-1}^j. \quad (4.16)$$

To obtain the tightest upper bound (in a L_2 sense) we minimize the L_2 distance between l_{Actual} and $l_{\text{SeqNBest2}}$ to obtain a . The L_2 distance between l_{Actual} and $l_{\text{SeqNBest2}}$ is

$$A = \int \|l_{\text{SeqNBest2}} - l_{\text{Actual}}\|_2^2 dl, \quad (4.17)$$

$$= \int_0^{\xi_{i-1}^j} [(l_i^j - \xi_{i-1}^j) - (al_i^j + b)]^2 dl \quad (4.18)$$

$$+ \int_{\xi_{i-1}^j}^1 (al_i^j + b)^2 dl. \quad (4.19)$$

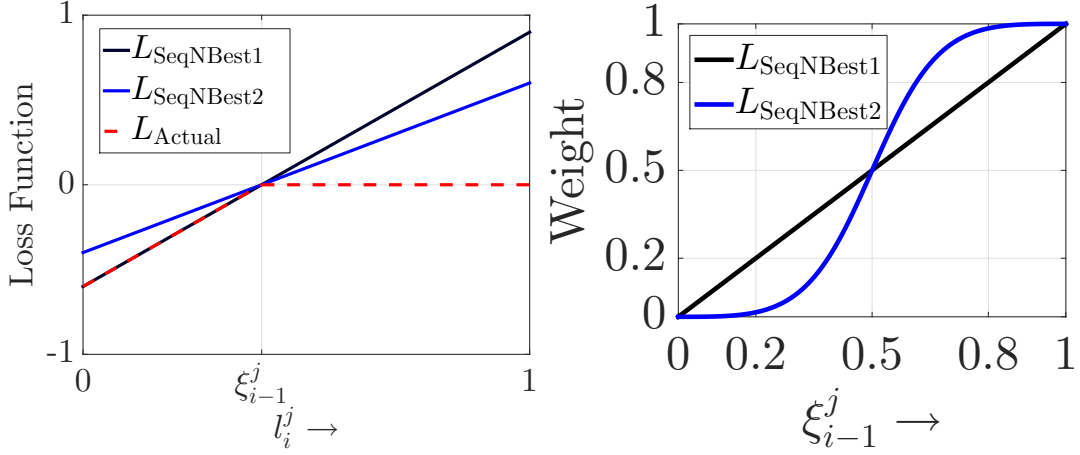
Differentiating the expression for A with respect to a (the slope of the line), setting it to 0, and using the constraint that the line must pass through $(\xi_{i-1}^j, 0)$, we get

$$a = \frac{(\xi_{i-1}^j)^3}{3(\xi_{i-1}^j)^2 - 3\xi_{i-1}^j + 1}. \quad (4.20)$$

This gives the optimal slope of the line $l_{\text{SeqNBest2}}$ which minimizes the gap between it and l_{Actual} .

The graphical relationship between ξ_{i-1}^j and the optimal weight (slope of the line) in $l_{\text{SeqNBest2}}$ is shown in Figure 4.2a. We see that $l_{\text{SeqNBest1}}$ weights the examples directly proportional to the previous best loss ξ_{i-1}^j , while $l_{\text{SeqNBest2}}$ tends to aggressively upweight hard samples which have high best previous loss ($\xi_{i-1}^j > 0.5$) and aggressively downweights easier examples which have low best previous loss ($\xi_{i-1}^j < 0.5$).

We summarize our algorithm in Algorithm 14. We begin by assigning a weight of 1 to



(a) Upper bound relationship between the proposed surrogate loss and the desired actual loss function (b) Comparison between the proposed weighting schemes as a function of the best loss seen so far.

each training sample in the dataset. In each iteration i , we train the structured predictor \hat{h}_i with the dataset \mathcal{D} and associated weights for each training sample \mathbf{w} and append it to the list of predictors S . We recompute the weights \mathbf{w} using the scheme described in Algorithm 15. We iterate for the specified N iterations and return the list $S = \{\hat{h}_1, \dots, \hat{h}_N\}$ of structured predictors. We term this simple but powerful approach as “Sequential N-Best” or SEQNBEST.

4.2.1 An Example

As an example, on the task of image segmentation the required inputs are the number of predictions N , we want to make per example, training dataset \mathcal{D} and the structured prediction procedure for learning and inference \hat{h} . We illustrate the algorithm via a toy dataset of 3 images (See Figure 4.2), where the task is to perform foreground/background segmentation by marking each pixel with either the foreground or background label. Assume that we have trained 2 predictors already, and are calculating the importance (weight) of each image for the 3rd predictor. The second and third rows show the performance of the two predictors on these images. Note that none of the predictors do well on the image of the elephant, however one of the predictors does really well on the helicopter. This tells us intuitively, that training the third predictor should concentrate more on the image of the elephant, but not as much on the other two since at least one of the previous predictors has done relatively well on it. The last row in the figure shows the weights for each image which is the minimum of the errors obtained by all previous predictors. This weighting rule achieves the desired behavior of working harder on examples which none of the previous predictors have performed well on.





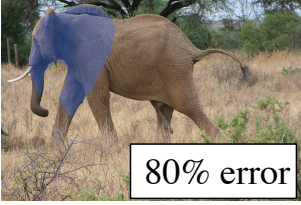




Image			
Output of Predictor 1			
Output of Predictor 2			
Weight for Predictor 3	$= \min(0.9, 0.3)$ $= 0.3$	$= \min(0.8, 0.6)$ $= 0.6$	$= \min(0.85, 0.05)$ $= 0.05$

Figure 4.2: Illustration of SEQNBEST training procedure. Consider a toy training dataset of 3 images (chosen from the iCoseg dataset [Batra et al. 2010], where the task is to do foreground/background separation. The first predictor gets 30% pixel error on the bear image, while the second predictor gets 90% pixel error. Intuitively, since the first predictor did well already on this image, we should not try as hard on this image compared to the elephant image where none of the 2 predictors did very well. The rule for weighting data points for training the next predictor is minimum of the error by the previous predictors and the last column shows this being applied to this contrived example. Note that the elephant image has the highest weight since none of the previous predictors did well on it, while the helicopter one has the lowest weight, since the first predictor did really well on it.

4.3 Case Studies

We evaluate our methods against both *model-dependent* [Park and Ramanan 2011; Guzman-Rivera et al. 2012; Batra et al. 2012] and model-independent methods [Guzman-Rivera et al. 2014b] (See Chapter 4.1). Note that the weighting scheme of GR14 [Guzman-Rivera et al. 2014b] has been used for the specific task of camera re-localization and published results on standardized datasets do not exist. We make a best effort comparison by reimplementing their method for standardized tasks.

We demonstrate that using our simple yet powerful weighting scheme results in better performance than model-dependent methods and comparable or better performance for model-agnostic methods with much less computation due to lack of parameter tuning step.

4.3.1 Case Study: Human Pose Tracking in Monocular Sequences

In monocular pose estimation the task is to estimate the 2D locations of anatomical landmarks from an image. The task is challenging due to the large variation in appearance and configuration of humans in images. Additional challenges are posed by partial occlusions, self-occlusions, and foreshortening. A related task is to track the pose of a human subject through a sequence of frames of video. In the tracking by detection paradigm of human pose tracking, multiple hypothesis poses are generated per frame of video and then stitched together using a data association algorithm. This avoids making hard commitment to a single best pose at a frame. As long as the correct pose is present amongst the multiple hypothesized poses for each frame, the algorithm can have a chance at picking the correct one using additional temporal information.

Datasets: We evaluate our method on producing multiple predictions for each image in the PARSE dataset used introduced by Yang and Ramanan [Yang and Ramanan 2011] and on the tracking datasets introduced in Park and Ramanan [Park and Ramanan 2011] named “lola”, “lola”, “walkstraight” and “baseball”. We use the same model, code and training set as Yang and Ramanan [Yang and Ramanan 2011] and use our two weighting methods to train N models as detailed in Algorithm 14 to produce 4 models. We use the same test set used by Yang and Ramanan to compare the average percentage of correct parts (PCP) of the *best* pose as the number of pose hypotheses is increased from 1 to 4.

Analysis Figure 4.3 shows that as the number of hypotheses is increased SEQNBEST1 and SEQNBEST2 find accurate poses earlier in the list than NBEST. The figure plots the average across the test set, of the best pose predicted as the number of pose hypotheses is increased. Batra et al. [Batra et al. 2012] refer to this as the “oracle” accuracy of a list of predictions. We show results with NBEST with and without non-maximum suppression post processing. Note that even with non-maximum suppression, NBEST is unable to outperform SEQNBEST, which requires no post-processing step. We also compare against the boosting-like weighting scheme of GR14 [Guzman-Rivera et al. 2014b]. GR14 performs marginally better than SEQNBEST, achieving 81.95% oracle accuracy compared to SEQNBEST1’s 80.83% by position 4. Note that this boosting-like weighting scheme has a free parameter which is tuned by cross-validation, while we are parameter free. We used the exact same set of values of this free parameter as used in [Guzman-Rivera et al. 2014b] to tune it for all our following experiments.

In Figure 4.1 we compare the performance of DIVMBEST with respect to SEQNBEST1 and SEQNBEST2. Three models were trained using the two SEQNBEST schemes on the PARSE training set and then compared to the “oracle” PCPs reported by NBEST and DIVMBEST. In each video sequence SEQNBEST1 or SEQNBEST2 achieves higher recall. In the

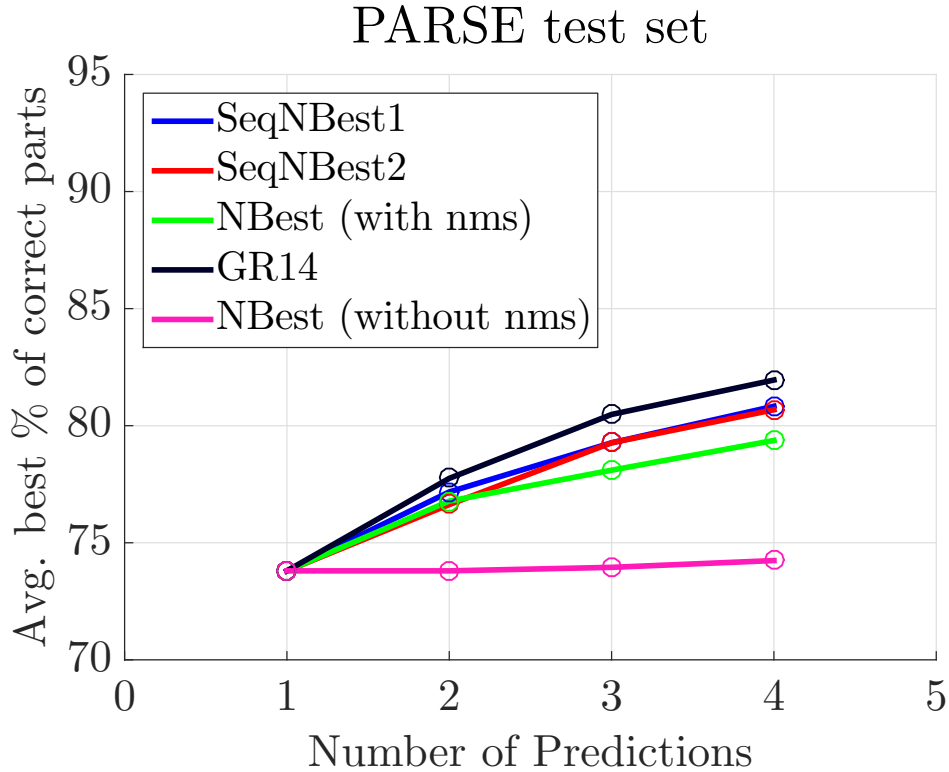


Figure 4.3: As the number of pose hypotheses allowed is increased from one to four, SEQNBEST predicts more accurate poses compared to NBEST with non-maximum suppression. Both start out at 73.8% percentage of correct parts since the first position’s model is identical to both but by the 4th position SEQNBEST has achieved 81.61% average best accuracy while NBEST achieves 79.37%.

“walkstraight” dataset SEQNBEST1 achieves 98.5% PCP in 3 positions where DIVMBEST needs 100 predictions to reach the same accuracy. Similarly for “lola1”, 20 predictions, “lola2”, 7 predictions and for “baseball” 7 predictions are needed by DIVMBEST to reach the same “oracle” PCP as SEQNBEST2. Note that GR14 after much tuning on validation data is still behind SEQNBEST on all four videos.

4.3.2 Case Study: Image Foreground/Background Separation

We apply our method to the task of foreground/background segmentation where the task is to assign each pixel in an image with either the foreground or background label.

Dataset: We use the set of 166 images of the iCoseg dataset [Batra et al. 2010], spanning 9 different events, as used by MCL [Guzman-Rivera et al. 2012]. The dataset is roughly, equally split into training, validation and test sets. The exact splits were provided to us by the authors of MCL.

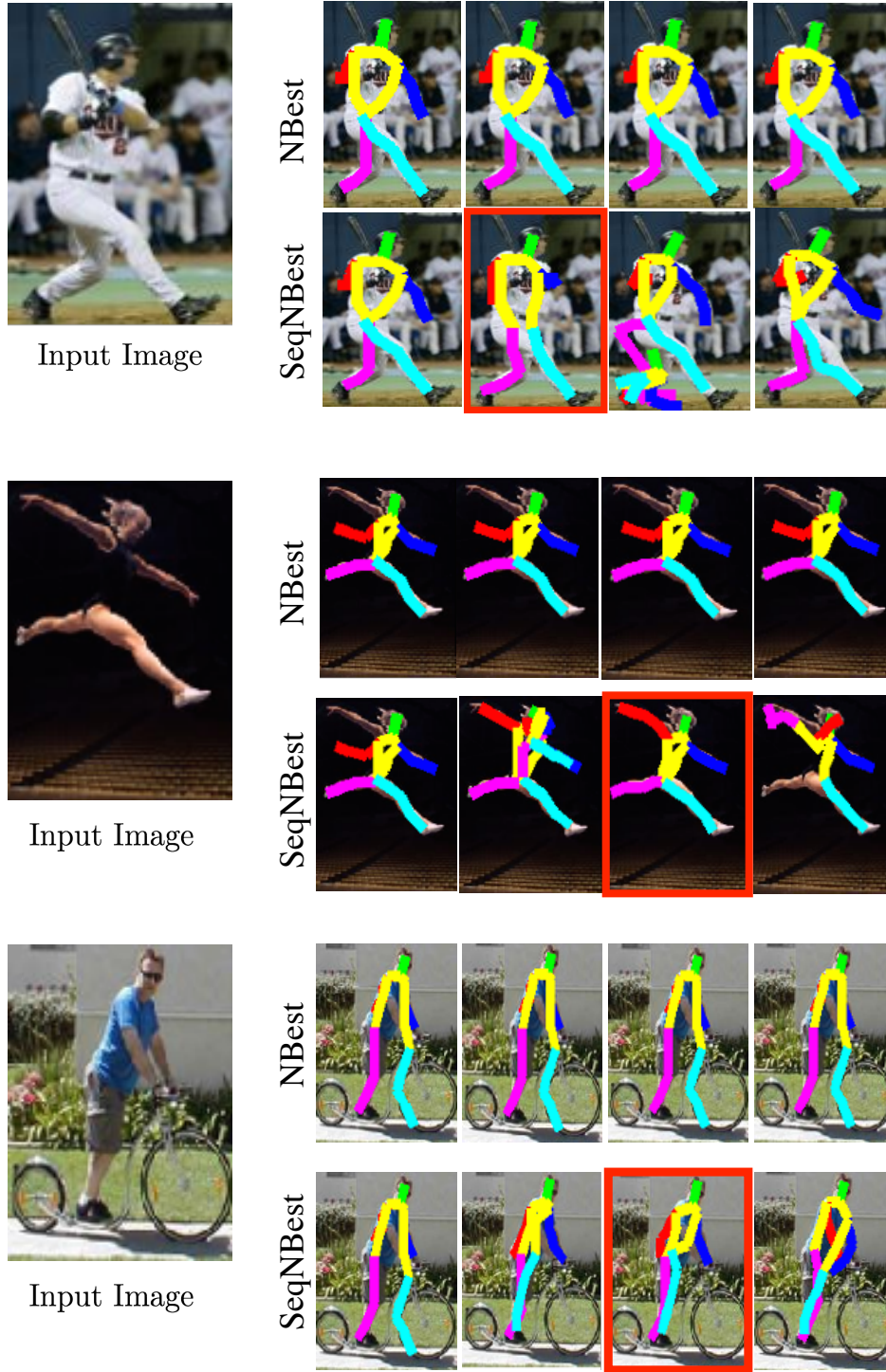


Figure 4.4: For each of the three images the top row is the list of 4 pose hypothesis by NBEST while the bottom 4 are by SEQNBEST. For baseball player SEQNBEST predicts the correct pose in the 2nd guess, for the gymnast in the 3rd guess and the 4th guess for the cyclist. Note that in each case SEQNBEST1 produces poses which are diverse from each other while trying to be relevant to the scene. In each case NBEST produces poses which are almost identical to each other and none of which are close to the ground truth pose.

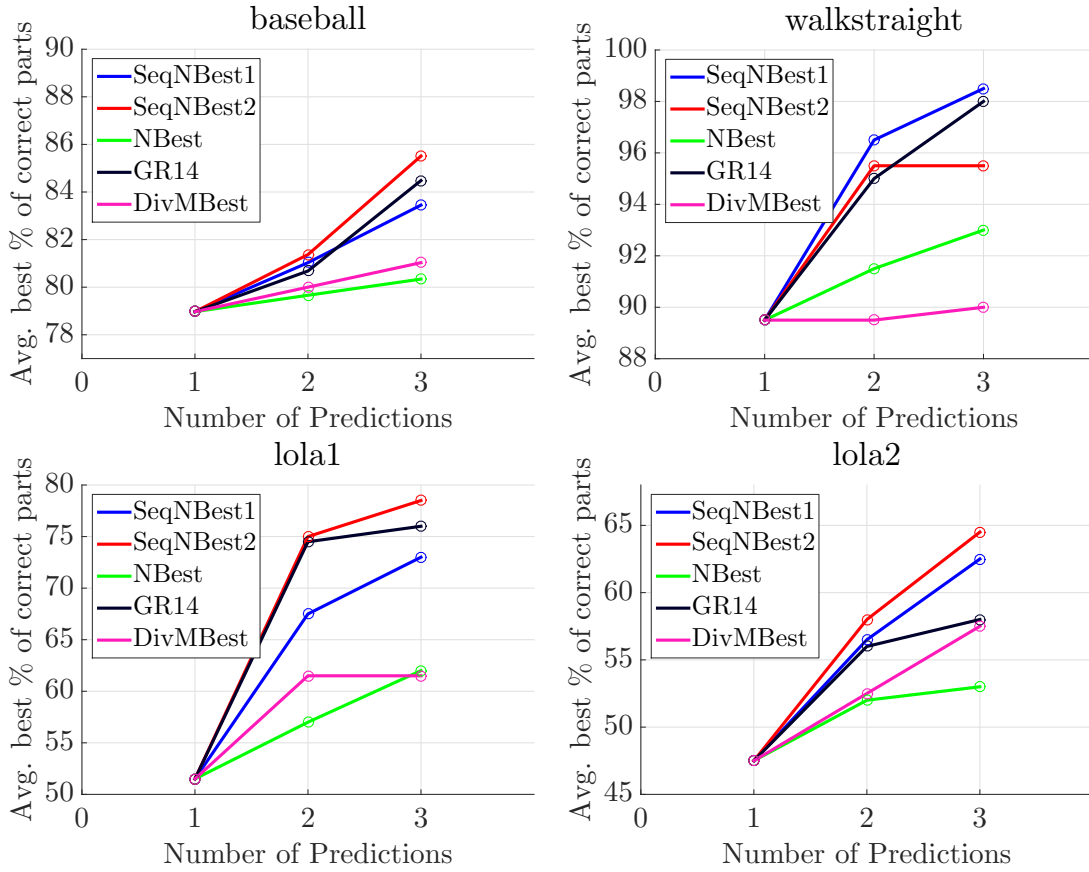


Table 4.1: Comparison of SEQNBEST1 and SEQNBEST2 to NBEST and DIVMBEST. The average best PCP plotted as the budget for generating hypotheses is increased. In each case SEQNBEST1 and/or SEQNBEST2 predicts more accurate poses for the number of hypotheses allowed.

Analysis: We compare the performance of SEQNBEST to MCL in two ways: 1) We use the exact implementation of S-SVM provided to us by the authors of MCL as the structured predictor routine in SEQNBEST to train 6 predictors 2) Secondly, *to showcase the flexibility* of SEQNBEST to use *any* structured predictor available, we use the Hierarchical Inference Machine (HIM) algorithm by Munoz et al. [Munoz et al. 2010] to train SEQNBEST. We use texture and C-SIFT [Gould et al. 2010] as features. Figure 4.5 (left) shows the “oracle” accuracy of a list of predictions. Additionally we compared against GR14 [Guzman-Rivera et al. 2014b]. We find that using the *same predictor and features* as in MCL, SEQNBEST1 and MCL have comparable performance in Figure 4.5 (left). When HIM is used as the structured predictor (Figure 4.5 (right)), it performs much better from the first position and obtains 6% average best error in 6 predictions. The reduction of error stops after the first 3 positions because the HIM model starts approaching the theoretical limits of its performance on the test set, which is 2% (this was obtained by training and testing HIM on the test set itself).

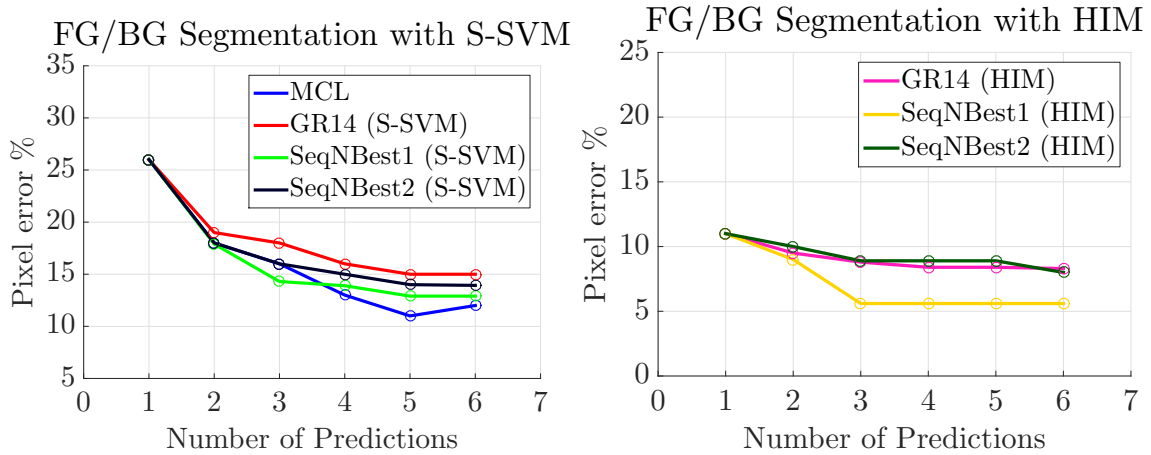


Figure 4.5: Average best pixel error in the image background, foreground segmentation task as number of predictions are increased. SEQNBEST (with S-SVM) uses the same S-SVM structured predictor routine as MCL.

In summary, variants of SEQNBEST performed on par with model-dependent methods like MCL, which have the advantage of leveraging the specifics of the chosen structured predictor (in this case S-SVM). SEQNBEST, however, is *model-agnostic* and can be readily applied to any structured predictor. We find that SEQNBEST used in conjunction with HIM outperforms the other model-agnostic method, GR14, which is also trained with HIM as the base predictor (Figure 4.5 (right)). This also serves as an example of SEQNBEST’s flexibility in being able to plug-in any powerful predictor.

4.3.3 Case Study: Image Segmentation

As mentioned earlier semantic scene segmentation is a very challenging task, where every pixel in an image has to be assigned a semantic label like “boat”, “sky” etc. In this section we show initial promising results with SEQNBEST. Note that these are not meant to be competitive with the most recent state-of-the-art advances in image segmentation but meant to showcase the flexibility of our approach in using *any* predictor.

Dataset: In PASCAL VOC 2012 segmentation challenge [Everingham et al.] the task is to mark every test image with one of 20 class labels or the **background** class. Figure 4.6 shows some example images and their annotated groundtruth labels. There are 1464 images in **train** and 1449 in the **val** set which we use as the test set in our experiments below.

Analysis: We use the Hierarchical Inference Machine (HIM) algorithm by Munoz et al. [Munoz et al. 2010] to learn 5 structured predictors in the SEQNBEST framework. We use the output of category-specific regressors of [Carreira et al. 2012] as additional features to HIM. In the

Table 4.2: As the number of predictions is increased, we observe a 10.60% gain in “oracle” accuracy over a single prediction on the PASCAL VOC 2012 `val` dataset.

Position	1	2	3	4	5
Oracle acc. (%)	42.91	45.96	46.44	47.09	47.46

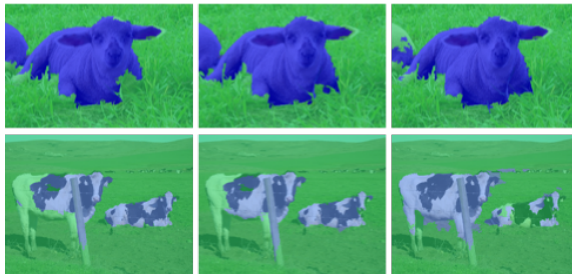


Figure 4.6: Qualitative examples of multiple semantic scene segmentations on the PASCAL VOC 2012 dataset. Each predictor tries to get right what the previous predictors have not been able to cover well. For example the cow grazing scene the first two predictors miss parts of the cow while the third one gets majority of it correct.

first position HIM achieves 42.91% average intersection/union accuracy over all 21 classes. Table 4.2 shows the “oracle” accuracy as the number of predictions is increased to 5 where the “oracle” accuracy is 47.46% which is a 10.6% gain.

Prasad et al. [Prasad et al. 2014], have proposed inference procedures for extracting diverse hypotheses in MRFs using various higher-order potentials [DeLong et al. 2012]. This is another example of the *model-dependent* category of methods as described in Chapter 4.1. Similar to us, they have demonstrated their method on the semantic segmentation challenge in PASCAL VOC 2012 `val` set. They show impressive “oracle” gains of $\sim 12\%$ over a single prediction. Since their model and code is not yet available, it is not currently possible to directly compare against SEQNBEST. We use a *different* model to achieve similar boosts. Again, this showcases the ease of use and generality of our approach. Note that we are not constrained to specific models or specific diversity terms which may be only compatible with particular model representations.

In ongoing experiments we are using recent advances in convolutional neural networks [Long et al. 2014; Hariharan et al. 2014] as the structured predictor for generating multiple segmentations using SEQNBEST.

Discussion

The three main approaches presented until now for predicting set and lists (CONSEQOPT (Chapter 2), SCP (Chapter 3), SEQNBEST (Chapter 4)) rely on “reductions” for algorithm design. The main idea in reductions is a simple but powerful one: given a challenging problem for which no obvious solution strategies exist, break it down into simpler problems with well-understood theoretical and practical solutions and then relate performance on the simpler problems to the original problem of interest. Well known reductions include: QUANTING [Langford et al. 2012] from quantile regression to classification; PROBING [Langford and Zadrozny 2005] from squared loss regression to classification; COSTING [Zadrozny et al. 2003] from importance weighted classification to binary classification by rejection sampling; SEARN [Daumé Iii et al. 2009] from structured prediction to binary classification; DAGGER [Ross et al. 2011] imitation learning and structured prediction to no-regret online learning. Such reductions leverage existing methods for classification, regression and structured prediction and allow rapid progress to be made on the new task. Furthermore, due to their modular nature, if improved techniques for classification, regression or structured prediction become available in the future, they can be readily plugged in without any change in the algorithm. This makes the approaches proposed in this work versatile and better able to weather the test of time.

[Yue and Joachims 2008] tackle the problem of predicting diverse sets of items for information retrieval settings. Their approach has two stages. In the first stage a user makes a query and a set of relevant documents are returned by an oracle (*e.g.* search engine). In the second stage, their approach then finds the subset of documents which achieves maximum approximate coverage of subtopics. This is obtained by using the GREEDY algorithm under the assumption that “covering” words in the user query will cover topics (since topics that a document covers are unknown). In contrast CONSEQOPT and SCP don’t separate the problem of list prediction into two stages, instead they train classifiers/regressors to directly predict the list that is competitive with the GREEDY algorithm that has perfect knowledge

of the user’s objective. This also bypasses the need for a relevance oracle like a search engine to find a set of relevant items in the first stage.

[Yue and Guestrin 2011] tackle the problem of personal news recommendation where a list of personalized news articles are recommended to the user. They assume a parameterized submodular reward function whose parameters are then learnt using user interaction data in a partial feedback setting using linear stochastic bandit algorithms (one copy per position of the list similar to [Streeter and Golovin 2008] and CONSEQOPT). The main limitation of this approach is the realizability assumption *i.e.* that the true user interaction model lies within their considered linear model class. In contrast the approaches proposed here explicitly consider a submodular reward function over lists and are agnostic to the reward model of the user. As a result any feature space can be used to model the reward that a particular article brings to a certain position in the list instead of only the submodular basis functions that are used in [Yue and Guestrin 2011].

As mentioned before in Chapter 4.1 determinantal point processes (DPP’s), is a model used in particle physics for optimizing for diverse but low error predictions. Given a library of items, questions regarding the probability of subsets of these items can be efficiently answered. Given a subset of the items, the probability of that subset is proportional to the value of the determinant of the sub-matrix whose rows and columns correspond to the items under consideration. This can be geometrically interpreted as the volume of the parallelepiped enclosed by the vectors representing those items. Vectors which are similar to each other will enclose less area as opposed to vectors which are much different. This naturally encourages sets consisting of diverse items to be picked. DPP’s have been used in document summarization, pose estimation and other tasks where predicting lists is important [Kulesza and Taskar 2011, 2010]. But both learning and inference in DPP’s remains approximate. In contrast we take the route of directly optimizing for the objective at hand to predict lists.

Autonomous UAV Flight through Dense Clutter

Unmanned Aerial Vehicles (UAVs) have recently received a lot of attention by the robotics community. While autonomous flight with active sensors like lidars has been well studied [Scherer et al. 2008; Bachrach et al. 2009], flight using passive sensors like cameras has relatively lagged behind. This is especially important given that small UAVs do not have the payload and power capabilities for carrying such sensors. Additionally, most of the modern research on UAVs has focussed on flying at altitudes with mostly open space [Dey et al. 2011]. Flying UAVs close to the ground through dense clutter [Ross et al. 2013a; Scherer et al. 2008] has been less explored. In this chapter we leverage the multiple prediction techniques developed in earlier chapters and apply them to the problem of pure vision-based autonomous UAV flight through dense clutter. We show that using the paradigm of multiple predictions we are able to increase average flight length by up to 71% over the single prediction case.

Receding horizon control [Kelly et al. 2006] is a classical deliberative scheme commonly used in autonomous ground vehicles including five out of the six finalists of the DARPA Urban Challenge [Buehler et al. 2008]. Figure 6.2 illustrates receding horizon control on our UAV in motion capture. In receding horizon control, a pre-selected set of dynamically feasible trajectories of fixed length (the horizon), are evaluated on a cost map of the environment around the vehicle and the trajectory that avoids collision while making most progress towards a goal location is chosen. This trajectory is traversed for a bit and the process repeated again.

We demonstrate the *first* receding horizon control with monocular vision implementation on a UAV. Figure 6.1 shows our quadrotor evaluating a set of trajectories on the projected depth image obtained from monocular depth prediction and traversing the chosen one.

This is motivated by our previous work [Ross et al. 2013a], where we used imitation learning to learn a purely reactive controller for flying a UAV using only monocular vision

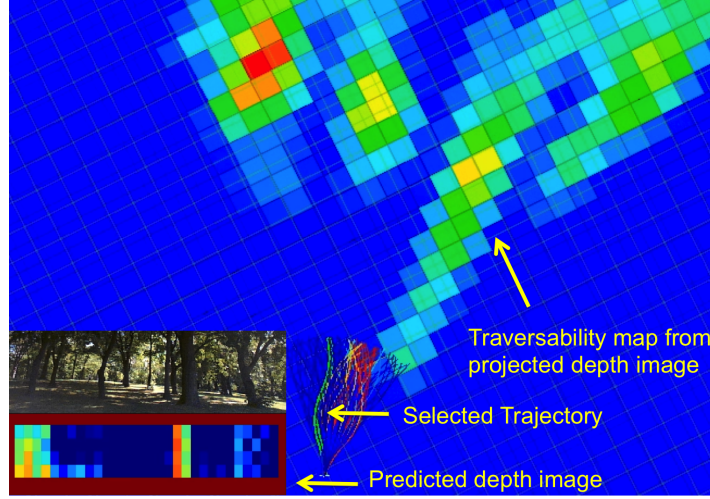


Figure 6.1: Example of receding horizon with a quadrotor using monocular vision. The lower left images show the view from the front camera and the corresponding depth images from the monocular depth perception layer. The rest of the figure shows the overhead view of the quadrotor and the traversability map (built by projecting out the depth image) where red indicates higher obstacle density. The grid is $1 \times 1 \text{ m}^2$. The trajectories are evaluated on the projected depth image and the one with the least collision score (thick green) trajectory followed.

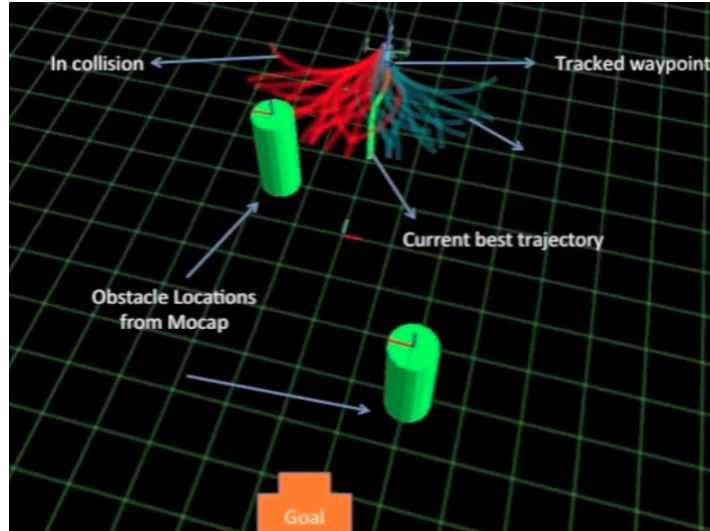


Figure 6.2: Receding horizon control on UAV in motion capture. A library of 78 trajectories of length 5 m are evaluated to find the best collision-free trajectory. This is followed for some time and the process repeated.

through dense clutter. While good obstacle avoidance behavior was obtained, there are certain limitations of a purely reactive layer that a more deliberative approach like receding horizon control can ameliorate. Reactive control is by definition myopic, i.e., it concerns itself with avoiding the obstacles closest to the vehicle. This can lead to it being easily stuck in cul-de-sacs. Since receding horizon control plans for longer horizons it achieves better plans and minimizes the chances of getting stuck [Knepper and Mason 2009]. Another limitation of pure reactive control is the difficulty to reach a goal location or direction. In a receding horizon control scheme, trajectories are selected based on a score which is the sum of two terms: first, the collision score of traversing it and second, the heuristic cost of reaching the goal from the end of the trajectory. By weighting both these terms suitably, goal-directed behavior is realized while maintaining obstacle-avoidance capability. But it is to be noted that reactive control can be integrated with receding horizon for obtaining the best of both worlds in terms of collision avoidance behavior.

Receding horizon control needs three working components

1. *A method to estimate depth:* This can be obtained from stereo vision [Schmid et al. 2014; Matthies et al. 2014] or dense structure-from-motion (SfM) [Wendel et al. 2012]. But these are not amenable for achieving higher speeds due to high computational expense. We note that in the presence of enough computation power, information from these techniques can be combined with monocular vision to improve overall perception.

Biologists have found strong evidence that birds and insects use optical flow to navigate through dense clutter [Srinivasan 2011]. Optical flow has been used for autonomous flight of UAVs [Beyeler et al. 2009]. However, it is difficult to directly derive a robust control principle from flow. Instead we follow the same data driven principle as our previous work [Ross et al. 2013a] and use local statistics of optical flow *as features* in the monocular depth prediction module. This allows the learning algorithm to derive complex behaviors in a data driven fashion.

2. *A method for relative pose estimation:* To track the trajectory chosen at every cycle, the pose of the vehicle must be tracked. We demonstrate a relative pose estimation system using a downward facing camera and a sonar, which is utilized by the controller for tracking the trajectory (Chapter 6.2.5).
3. *A method to deal with perception uncertainty:* Most planning schemes either assume that perception is perfect or make simplistic assumptions of uncertainty. We introduce the concept of making multiple, relevant yet diverse predictions for incorporating perception uncertainty into planning. The intuition is predicated on the observation that avoiding a small number of ghost obstacles is acceptable as long as true obstacles

are not missed (high recall, low precision). The details are presented in Chapter 6.2.4 and are related to the methods developed in Chapters 2 and 3. We demonstrate in experiments the efficacy of this approach as compared to making only a single best prediction.

In summary our list of contributions are:

- Budgeted near-optimal feature selection and fast non-linear regression for monocular depth prediction.
- Real time relative vision-based pose estimation.
- Multiple predictions to efficiently incorporate uncertainty in the planning stage.
- First complete receding horizon control implementation on a UAV with monocular vision.

6.1 Hardware and Software Overview

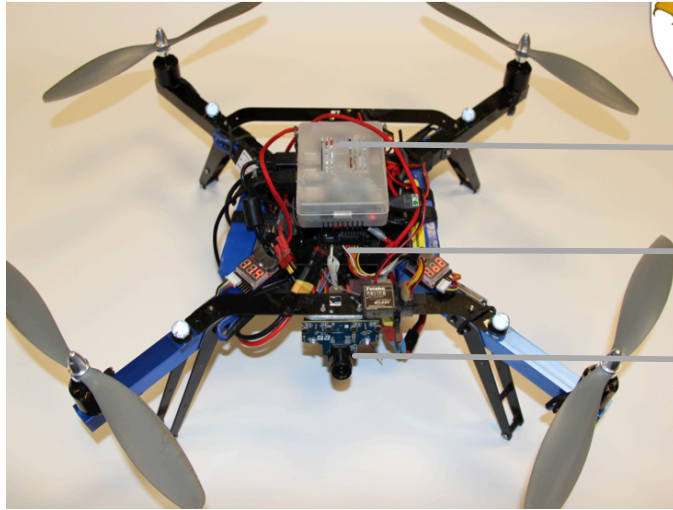
In this section we describe the hardware platforms used in our experiments. Developing and testing all the integrated modules of receding horizon is challenging. Therefore we assembled a rover (Figure 6.3) in addition to a UAV (Figure 6.3) to be able to test various modules separately. The rover also facilitated parallel development and testing of modules. Here we describe the hardware platforms and overall software architecture.

6.1.1 Rover

The skid-steered rover (Figure 6.3) uses an Ardupilot microcontroller board [Ardupilot 2015] which takes in high level control commands from the planner and controls four motors to achieve the desired motion.

Other than the low-level controllers, all other aspects of the rover are kept exactly the same as the UAV to allow seamless transfer of software. For example, the rover has a front facing PlayStation Eye color camera (640×480 at 30Hz) which is also used as the front facing camera on the UAV.

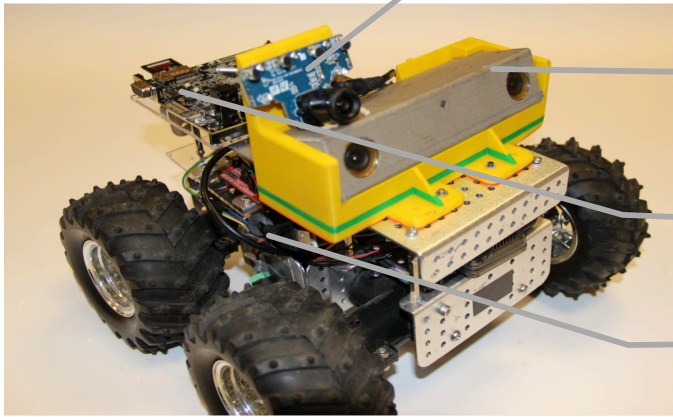
A Bumblebee color stereo camera pair (1024×768 at 20Hz) is rigidly mounted with respect to the front camera using a custom 3D printed fiber plastic encasing. This is used for collecting data with groundtruth depth values (Chapter 6.2) and validation of planning (Section 6.2.6). We calibrate the rigid body transform between the front camera and the left camera of the stereo pair using Bouget’s camera calibration toolbox [Bouguet 2004].



Arm-based
Linux computer

IMU Microstrain
3DM-GX3-25

PlayStation Eye camera
(640x480 @ 30Hz)



PlayStation Eye camera
(640x480 @ 30Hz)

Bumblebee Stereo
color camera
(1024x768 @ 20Hz)

Onboard ARM-based
Linux computer

Ardupilot for low level control.
Same chip is used on UAV.

Figure 6.3: (Top) Quadrotor used as our development platform. (Bottom) Rover assembled with the same control chips and perception software as UAV for rapid tandem development and validation of modules.

Stereo depth images and front camera images are recorded simultaneously while driving the rover around using a joystick. The depth images are then transformed to the front camera’s coordinate system to provide groundtruth depth values for every pixel. The training depth images are from a slightly different perspective than encountered by the UAV during flight, but we found in practice that the depth prediction modules generalized well to the UAV. Details in Chapter 6.2

6.1.2 UAV

Figure 6.3 shows the quadrotor we use for our experiments. Figure 6.4 shows the schematic of the various modules that run onboard and offboard. The base chassis, motors and autopilot are assembled using the Arducopter kit [Ardupilot 2015]. Due to drift and noise of the IMU integrated in the Ardupilot unit, we added a Microstrain 3DM-GX3-25 IMU which is used to aid real time pose estimation. There are two cameras: one facing downwards for real time pose estimation (PlayStation Eye color camera, 320×240 at 120Hz) and one facing forward (PointGrey Chameleon color camera 640×480 at 30Hz) for obstacle avoidance. The onboard processor is an Odroid XU-3 quad-core ARM based small board computer [Odroid 2015] which runs Ubuntu 14.04 and ROS Groovy [Quigley et al. 2009]. This unit runs the pose tracking and trajectory following modules. LidarLite, a lidar based sensor [LidarLite 2015] is used to estimate altitude. The image stream from the front facing camera is streamed to the base station where the depth prediction module processes it; the trajectory evaluation module then finds the best trajectory to follow to minimize probability of collision and transmits it to the onboard computer where the trajectory following module runs a pure pursuit controller to do trajectory tracking [Coulter 1992]. The resulting high level control commands are sent to the Ardupilot which sends low level control commands to the motor controllers to achieve the desired motion. In following sections we describe each module in detail.

6.2 Monocular Depth Prediction

In this section we detail the 3 depth prediction techniques we have developed and used in experiments, preceded first by the data collection methodology.

6.2.1 Data Collection

RGB-D sensors like the Kinect, currently do not work outdoors. Since camera and calibrated nodding lidar setup is expensive and complicated we used a rigidly mounted Bumblebee stereo

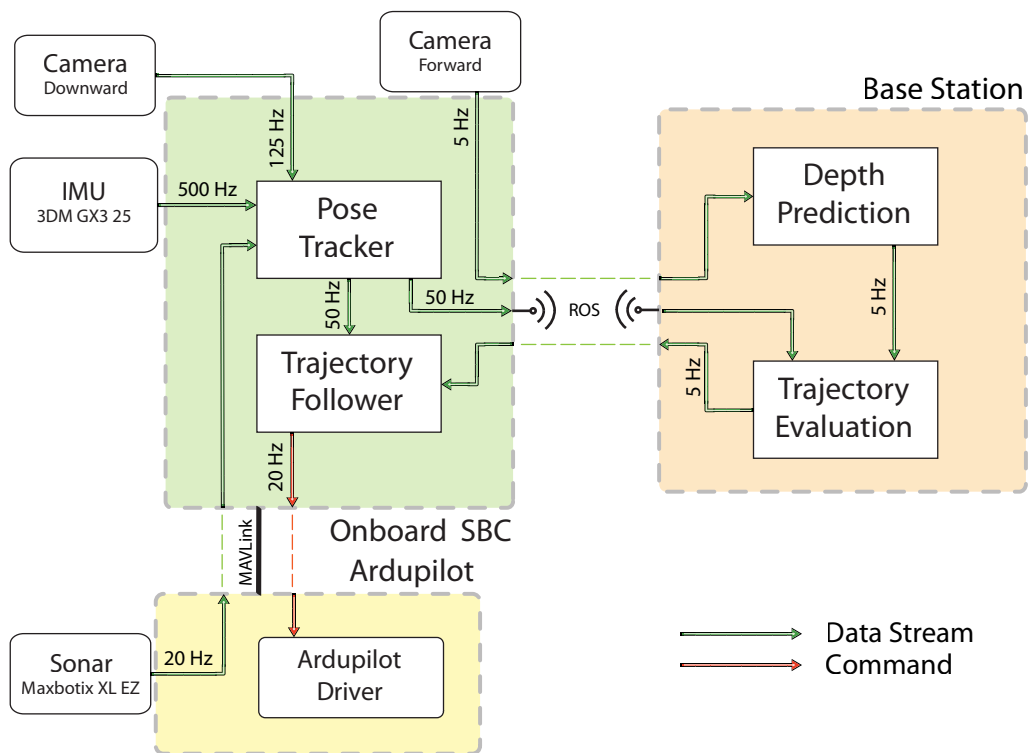


Figure 6.4: Schematic diagram of hardware and software modules

color camera and the PlayStation Eye camera for our outdoor data collection. This setup was mounted on the rover (Figure 6.3). We collected data at different neighboring locations with varying tree density, under varying illumination conditions and in both summer and winter conditions (Figure 6.5). Our corpus of imagery with stereo depth information is around 16000 images and growing. We will make this dataset publicly available in the near future.



(a) Testing and training areas near Carnegie Mellon University, (b) Additional training area with higher density of trees (approximately one tree per $6 \times 6 m^2$). The images below show a couple of examples from winter with snow on the ground. The tree density is approximately one tree per $12 \times 12 m^2$ area.

Figure 6.5: Testing and training areas.

6.2.2 Depth Prediction by Fast Non-linear Regression

In this section we describe the depth prediction approach from monocular images, and the fast non-linear regression method used for regression.

An image is first gridded up into non-overlapping patches. We predict the depth in meters at every patch of the image (Figure 6.6 yellow box). For each patch we extract features which describe the patch, features which describe the full column containing the patch (Figure 6.6

green box) and features which describe the column of three times the patch width (Figure 6.6 red box), centered around the patch. The final feature vector for a patch is the concatenation of the feature vectors of all three regions. When a patch is seen by itself it is very hard to tell the relative depth with respect to the rest of the scene. But by adding the features of the surrounding area of the patch, more context is available to aid the predictor [Divvala et al. 2009; Oliva and Torralba 2007].

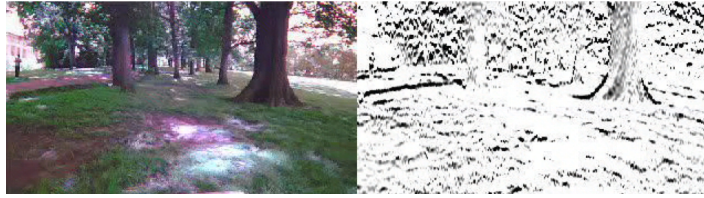


Figure 6.6: The yellow box is an example patch, the green box is the column of the same width, and the red box is the column of 3 times the patch width. Features are extracted individually at the patch, and the two columns. They are concatenated together to form the total feature representation of the patch.

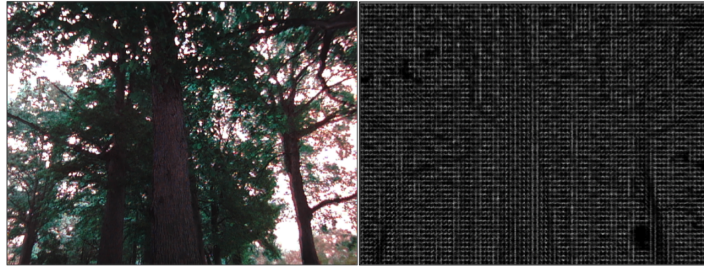
Description of features

In this part we describe in brief the features used to represent the patch. We mainly borrow the features as used in previous work on monocular imitation learning [Ross et al. 2013a] for UAVs, which are partly inspired by the work of Hoiem et al., [Hoiem et al. 2005] and Saxena et al., [Saxena et al. 2005]. We predict the depth at every patch using these features, which is then used by the planning module.

- *Optical flow*: We use the Farneback dense optical flow [Farneback 2003] implementation in OpenCV to compute for every patch the average, minimum and maximum optical flow values. By using optical flow statistics as features, temporal information is also available to the learning algorithm. While related works have used optical flow information directly to derive control policies [Srinivasan 2011; Beyeler et al. 2009], but in practice we found the estimation of flow itself to be noisy, even when a very computationally expensive optimization based algorithm using a GPU is used [Wedel et al. 2009]. Additionally for the area of the image directly in front of the UAV, optical flow is the zero by definition. This makes distinguishing between obstacles and free space



(a) Radon transform features help capture strong edges in images



(b) Histogram of oriented gradients (HoG) features help capture local gradient orientation information



(c) We used the per pixel hand detector of Li et al. [Li and Kitani 2013] to detect trees in images and its output as features to the learning algorithm for predicting depth



(d) Temporal information is maintained by computing optical flow statistics over the image, statistics of which are then used as features in the learning algorithm for depth prediction.

Figure 6.7: Illustration of image features used in the learning algorithm for monocular depth prediction.

directly in front of the UAV difficult. Instead we have taken the aforementioned data driven approach where the estimation of the local scene depth is left to the learning algorithm which is *aided* by optical flow information but is not completely dependent on it. See Figure 6.7d.

- *Radon Transform*: The radon transform [Helgason 1980] of an image is computed by summing up the pixel values along a discretized set of lines in the image, resulting in a 2D matrix where the axes are the two parameters of a line in 2D: angle θ of the line and s the distance along the line. We discretize this matrix in to 15×15 bins. For each angle θ the two highest values are recorded. This encodes the orientations of strong edges in the image. See Figure 6.7a
- *Structure Tensor*: At every point in a patch the structure tensor [Harris and Stephens 1988] is computed and the angle between the two eigenvectors is used to index in to a 15-bin histogram for the entire window. The corresponding eigenvalues are accumulated in the bins. In contrast to the radon transform, the structure tensor is a more local descriptor of texture. Together with radon features the texture gradients are captured, which are strong monocular depth cues [Wu et al. 2004].
- *Laws' Masks*: These describe the texture intensities [Davies 2004]. We use six masks obtained by pairwise combinations of one dimensional masks: (L)evel, (E)dge and (S)pot. The image is converted to the YCrCb colorspace and the LL mask is applied to all three channels. The remaining five masks are applied to the Y channel only. The results are computed for each window and the mean absolute value of each mask response is recorded.

For further details on radon transform, structure tensor and Laws' masks usage see [Ross et al. 2013a].

- *Histogram of Oriented Gradients (HoG)*: This feature has been used widely in the computer vision community for capturing texture information for object detection [Dalal and Triggs 2005]. The HoG descriptor computes the histogram of local gradient orientations over local patches in an image. In our implementation we compute the HoG feature over 9 orientation bins. See Figure 6.7b
- *Tree feature*: We use the per pixel fast classifier by Li et al. [Li and Kitani 2013] to train a supervised tree detector. Li et al. originally used this for real time hand detection in ego-centric videos. They use a random forest to predict whether each pixel in an image belongs to a human hand. We adapted this fast per pixel labeling method to predict for us the probability of each pixel belonging to a tree, in an image patch. This information is then used as a feature for that patch. See Figure 6.7c

Fast Non-linear Prediction

Due to harsh real-time constraints an accurate but fast predictor is needed. Recent linear regression implementations are very fast and can operate on millions of features in real time [Langford et al. 2007] but are limited in predictive performance by the inherent linearity assumption. In very recent work Agarwal et al. [Agarwal et al. 2013] develop fast iterative methods which use linear regression in the inner loop to obtain overall non-linear behavior. This leads to fast prediction times while obtaining much better accuracy. We implemented Algorithm 2 in [Agarwal et al. 2013] and found that it lowered the error by 10% compared to just linear regression, while still allowing real time prediction.

Budgeted Feature Selection

While many different visual features can be extracted on images, they need to be computed in real time. The faster the desired speed of the vehicle, the faster the perception and planning modules have to work to maintain safety. Additionally the limited computational power onboard a small UAV imposes a budget within which to make a prediction. Each kind of feature requires different time periods to extract, while contributing different amounts to the prediction accuracy. For example, radon transforms might take relatively less time to compute but contribute a lot to the prediction accuracy, while another feature might take more time but also contribute relatively less or vice versa. This problem is further complicated by the “grouping” effects where a particular feature’s performance is affected by the presence or absence of other features.

Given a time budget, the naive but obvious solution is to enumerate all possible combinations of features within the budget and find the group of features which achieve minimum loss. This is exponential in the number of available features. Instead we use the efficient approach developed by Hu et al. [Hu et al. 2014] to select the near-optimal set of features which meet the imposed budget constraints. Their approach uses a simple greedy algorithm that first whitens feature groups and then recursively chooses groups by the reduction in explained variance divided by the time to achieve that reduction. A more efficient variant of this with equivalent guarantees, chooses features by computing gradients to approximate the reduction in explained variance, eliminating the need to “try” all feature groups sequentially. For each specified time budget, the features selected by this procedure are within a constant factor of the optimal set of features which respect that budget. Since this holds across all time budgets, this procedure provides a recursive way to generate feature sets across time steps.

Figure 6.8 shows the sequence of features that was selected by Hu et al.’s [Hu et al. 2014]

feature selection procedure. For any given budget only the features on the left up to the specified time budget need to be computed.

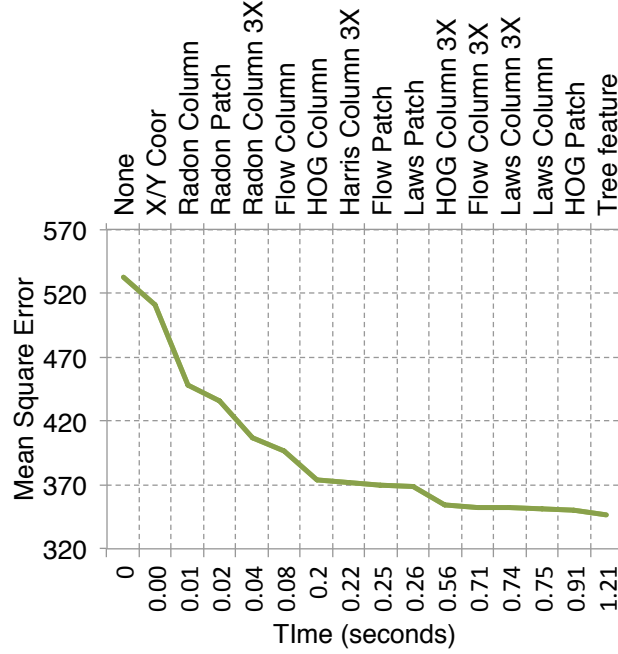


Figure 6.8: On the upper x-axis the sequence of features selected by Hu et al.’s method [Hu et al. 2014] and the lower x-axis shows the cumulative time taken for all features up to that point. The near-optimal sequence of features rapidly decrease the prediction error. For a given time budget, the sequence of features to the left of that time should be used.

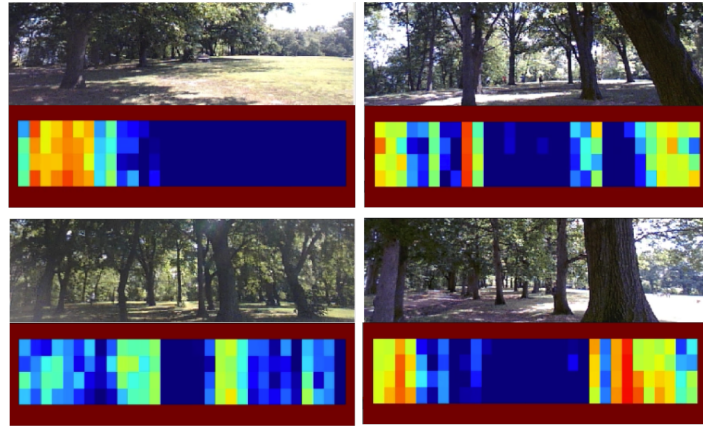


Figure 6.9: Depth prediction examples on real outdoor scenes. Closer obstacles are indicated by red.

6.2.3 Visual Odometry based Depth Prediction

Monocular visual odometry (VO) is the process of jointly estimating the camera pose and 3D scene geometry given a sequence of camera images. Generally, feature-based methods for VO [Davison et al. 2007; Klein and Murray 2007] consist of two separate steps: First, a set of feature observations is obtained from the given image. Second, camera pose and scene geometry are obtained as a function of these features only. While, this abstraction greatly reduces the complexity of the problem, it comes with several drawbacks. While feature-based methods allow us to estimate camera pose in real-time, the resulting feature based maps provide a very sparse representation of the scene geometry to have any reliable collision avoidance. Only image information conforming to the respective feature type and parametrization – typically image corners and blobs or line segments – is utilized. To overcome this limitation, direct approaches [Stühmer et al. 2010; Wendel et al. 2012; Templeton 2009; Geyer et al. 2006] for scene geometry reconstruction have become increasingly popular in the last few years.

Direct Methods for VO

Instead of operating solely on visual features, direct methods directly work on the images instead of a set of extracted features, for both mapping and tracking: The world is modeled as a dense surface while in turn new frames are tracked using whole-image alignment. This concept removes the need for discrete features, and allows the exploitation of all information present in the image. In addition to higher accuracy and robustness, in environments with little interesting points to extract features on, this provides substantially more information about the geometry of the environment. We utilize the framework of [Engel et al. 2013]: a method for semi-dense direct depth map estimation i.e. a dense depth map covering all image regions with non-negligible gradient.

Semi-dense Depth Map Estimation

The depth measurements are obtained by [Engel et al. 2013]’s proposed probabilistic approach for adaptive-baseline stereo. This method explicitly takes into account the knowledge that in video, small baseline frames occurs before large baseline frames. A subset of pixels is selected for which the disparity is sufficiently large and for each selected pixel a suitable reference frame is selected. A one dimensional disparity search is performed. The obtained disparity is converted to an inverse-depth representation, where the inverse depth is directly proportional to the disparity. The map is then updated using this inverse depth estimate.

The inverse depth map is propagated from to subsequent frames, once the pose of the

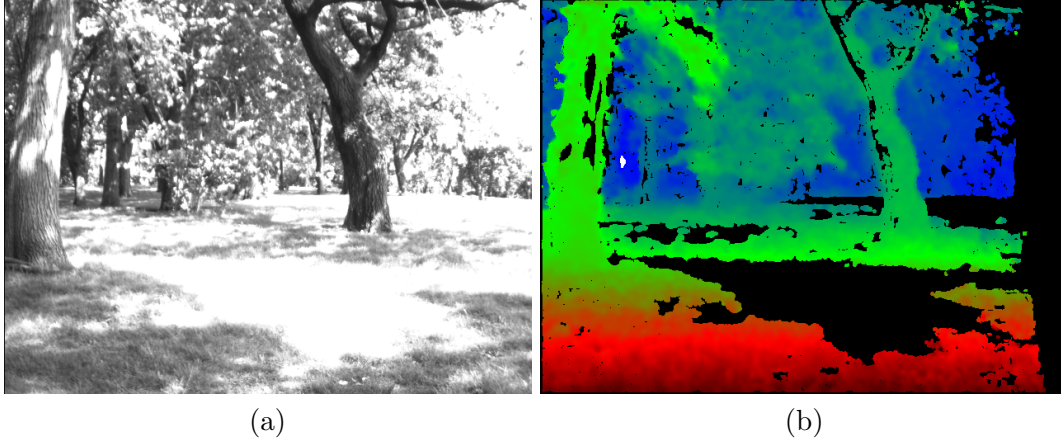


Figure 6.10: Semi-dense Depth Map Estimation. (a) Sample image and (b) Inverse Depth Map (red is near, blue is far) of a sample image during flight as generated by monocular visual odometry based approach. It is to be noted that only reliable depths are propagated and the rest are discarded (black regions), hence resulting in a semi-dense representation. Note: Best seen in color.

following frames have been determined and refined with new stereo depth measurements. Based on the inverse depth estimate d_0 for the pixel, the corresponding 3D point is calculated and projected into the new frame and assigned to the closest integer pixel position providing the new inverse depth estimate d_1 . We assume the camera rotation to be small, thus the new inverse depth map can be approximated by

$$d_1(d_0) = (d_0^{-1} - t_z)^{-1},$$

where t_z is the camera translation along the optical axis. Now, for each frame, after the depth map has been updated, a regularization step is performed by assigning each inverse depth value the average of the surrounding inverse depths, weighted by their respective inverse variance (σ^2). An example of the obtained depth estimates has been shown in Figure 6.10 Note: In order to prevent sharp edges, which can be critical in detecting trees, we only perform this step if two adjacent depth values are statistically similar i.e. their variances are within 2σ .

Dense Tracking

We represent an image as $I : \Omega \rightarrow \mathbf{R}$, the inverse depth map and inverse depth variance map as $D : \Omega_D \rightarrow \mathbf{R}^+$ and $V : \Omega_D \rightarrow \mathbf{R}^+$, where Ω_D contains all pixels which have a valid depth hypothesis. Note that D and V denote mean and variance of the inverse depth, as this approximates the uncertainty of stereo much better than assuming a Gaussian-distributed

depth [Montiel et al. 2006].

Given a semi-dense inverse depth map for the current image, the camera pose of the new frames is estimated using direct image alignment: given the current map $\{I_M, D_M, V_M\}$, the relative pose $\xi \in SE(3)$ of a new frame I is obtained by directly minimizing the photometric error

$$E(\xi) := \sum_{x \in \Omega_{D_M}} \|I_M(x) - I(w(x, D_m(x), \xi))\|_\delta,$$

where $w : \Omega_{D_M} \times \mathbf{R} \times SE(3) \rightarrow \omega$ projects a point from the reference frame image into the new frame and $\|\cdot\|_\delta$ is the Huber norm to account for outliers. The minimum is computed using iteratively re-weighted Levenberg-Marquardt minimization [Engel et al. 2014].

Scale Estimation

Scale ambiguity is inherent to all monocular visual odometry based methods. This is not critical in visual mapping tasks, where the external scale can be obtained using either fiducial markers [Daftry et al. 2015], or known dimension of objects in the scene as a post processing step. However, for obstacle avoidance in real-time, it is required to accurately recover the current scale so that the distance to the object is known in real world units. We resolve the absolute scale $\lambda \in \mathbf{R}^+$ by leveraging motion estimation from a highly accurate single beam laser lite sensor [LidarLite 2015] onboard. We measure, at regular intervals (operating at 15Hz), the 3-dimensional distance travelled according to the visual odometry $\mathbf{x}_i \in \mathbf{R}^3$ and the metric sensors $\mathbf{y}_i \in \mathbf{R}^3$. Given such sample pairs $(\mathbf{x}_i, \mathbf{y}_i)$, we obtain a scale $\lambda(t_i) \in \mathbf{R}$ as the running arithmetic average of the quotients $\frac{\|\mathbf{x}_i\|}{\|\mathbf{y}_i\|}$ over a small window size. We further pass the obtained set of scale measurements through a low-pass filter in order to avoid erroneous measurements due to sensor noise. The true scale λ is thus obtained and used to scale the depth map to real world units.

6.2.4 Multiple Predictions

The monocular depth estimates are often noisy and inaccurate due the challenging nature of the problem. A planning system must incorporate this uncertainty to achieve safe flight. Figure 6.11 illustrates the difficulty of trying to train a predictive method for building a perception system for collision avoidance. Figure 6.11 (left) shows a ground truth location of trees in the vicinity of an autonomous UAV. Figure 6.11 (middle) shows the location of the trees as predicted by the perception system. In this prediction the trees on the left and far away in front are predicted correctly but the tree on the right is predicted close to the UAV. This will cause the UAV to dodge a ghost obstacle. While this is bad, it is not fatal because

the UAV will not crash but make some extraneous motions. But the prediction of trees in Figure 6.11 (right) is potentially fatal. Here the trees far away in front and on the right are correctly predicted where as the tree on the left originally close to the UAV is mis-predicted to be far away. This type of mistake will cause the UAV to crash into an obstacle it does not know is there.

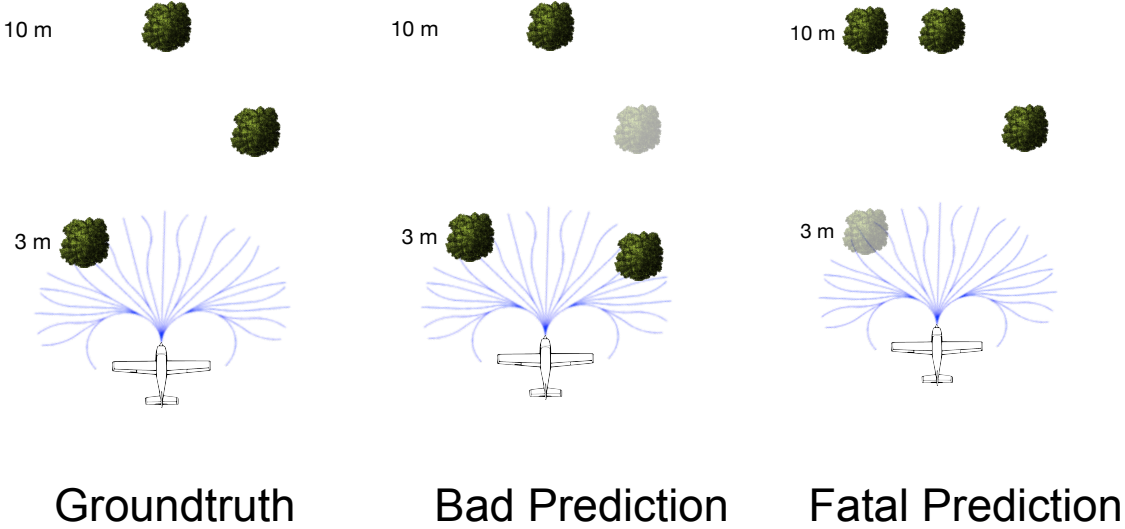


Figure 6.11: Illustration of the complicated nature of the loss function for collision avoidance. (Left) Groundtruth tree locations (Middle) Bad prediction where a tree is predicted closer than it actually is located (Right) Fatal prediction where a tree close by is mispredicted further away.

Ideally, a vision-based perception system should be trained to minimize loss functions which will penalize such fatal predictions more than other kind of predictions. But even writing down such a loss function is difficult. Therefore most monocular depth perception systems try to minimize easy to optimize surrogate loss functions like regularized L_1 or L_2 loss [Saxena et al. 2005]. We try to reduce the probability of collision by generating multiple interpretations of the scene to hedge against the risk of committing to a single potentially fatal interpretation as illustrated in Figure 6.11. Specifically we generate 3 interpretations of the scene and evaluate the trajectories in all 3 interpretations simultaneously. The trajectory which is least likely to collide on average in all interpretations is then chosen as the trajectory to traverse.

One way of making multiple predictions is to just sample the posterior distribution of a learnt predictor. In order to truly capture the uncertainty of the predictor, a lot of interpretations have to be sampled and trajectories evaluated on each of them. A large number of samples will be from around the peaks of this distribution leading to wasted samples. This is not feasible given the real time constraints of the problem.

In previous chapters (Chapters 2 and 3), we have developed techniques for predicting a budgeted number of interpretations of an environment with applications to manipulation, planning and control. [Batra et al., \[Batra et al. 2012\]](#) have also applied similar ideas to structured prediction problems in computer vision. These approaches try to come up with a small number of relevant but diverse interpretations of the scene so that at least one of them is correct. In this work, we adopt a similar philosophy and use the error profile of the fast non-linear regressor described in Chapter 6.2 to make two additional predictions: The non-linear regressor is first trained on a dataset of 14500 images and its performance on a held-out dataset of 1500 images is evaluated. For each depth value predicted by it, the average over-prediction and under-prediction error is recorded. For example the predictor may say that an image patch is at 3 meters while it is actually either, on average, at 4 meters or at 2.5 meters. We round each prediction depth to the nearest integer, and record the average over and under-predictions as in the above example in a look-up table (LUT). At test time the predictor produces a depth map and the LUT is applied to this depth map, producing two additional depth maps: one for over-prediction error, and one for the under-prediction error.

Similarly for the Direct VO based depth image prediction, we make multiple predictions by utilizing the variance of the estimated inverse depth which is already calculated in the framework of [\[Engel et al. 2013\]](#). At every pixel the variance of the inverse depth is used to find the inverse depth value one standard deviation away from the mean (both lower than and higher than the mean value) and inverted to obtain a depth value. So as before a total of 3 depth predictions are made: 1) mean depth estimate 2) depth estimate at one standard deviation lower than the mean depth at every pixel and 3) depth estimate at one standard deviation greater than the mean depth at every pixel.

Figure 6.12 shows an example in which making multiple predictions is clearly beneficial compared to the single best interpretation (using the non-linear regression depth estimation method). We provide more experimental details and statistics in Chapter 6.3.

6.2.5 Pose Estimation

As discussed before, a relative pose-estimation system is needed to follow the trajectories chosen by the planning layer. We use a downward looking camera in conjunction with a downward facing single beam lidar [\[LidarLite 2015\]](#) for determining relative pose. Looking forward to determine pose is ill-conditioned due to a lack of parallax as the camera faces the direction of motion. There are still significant challenges involved when looking down. Texture is often very self similar making it challenging for traditional feature based methods [\[Newcombe et al. 2011; Klein and Murray 2007\]](#) to be employed.

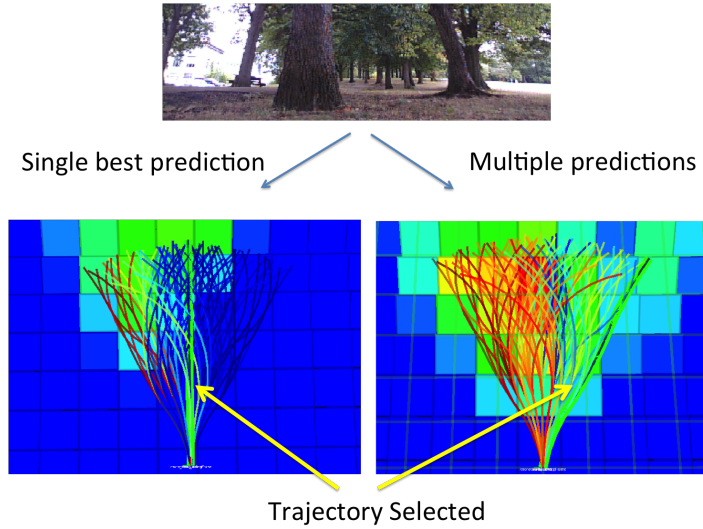


Figure 6.12: The scene at top is an example from the front camera of the UAV. On the left is shown the predicted traversability map (red is high cost, blue is low cost) resulting from a single interpretation of the scene. Here the UAV has selected the straight path (thick, green) which will make it collide with the tree right in front. While on the right the traversability map is constructed from multiple interpretations of the image, leading to the trajectory in the right being selected which will make the UAV avoid collision.

In receding horizon, absolute pose with respect to some fixed world coordinate system is not needed, as one needs to follow trajectories for short durations only. So as long as one has a relative, consistent pose estimation system for this duration (3 seconds in our implementation), one can successfully follow trajectories.

We used a variant of a simple algorithm that has been presented quite often, most recently in [Honegger et al. 2013]. This approach uses a Kanade-Lucas-Tomasi (KLT) tracker [Tomasi and Kanade 1991] to detect where each pixel in a grid of pixels moves over consecutive frames, and estimating the mean flow from these after rejecting outliers. We do the outlier detection step by comparing the variation of the flow vectors obtained for every pixel on the grid to a specific threshold. Whenever the variance of the flow is high, we do not calculate the mean flow velocity, and instead decay the previous velocity estimate by a constant factor.

This estimate of flow however tries to find the best planar displacement between the two patches, and does not take into account out-of-plane rotations, due to motion of the camera. Camera ego-motion is compensated using motion information from the IMU. Finally the metric scale is estimated from sonar. We compute instantaneous relative velocity between the camera and ground which is integrated over time to get position.

This process is computationally inexpensive, and can be run at very high frame rates. Higher frame rates lead to smaller displacements between pairs of images, which in turn

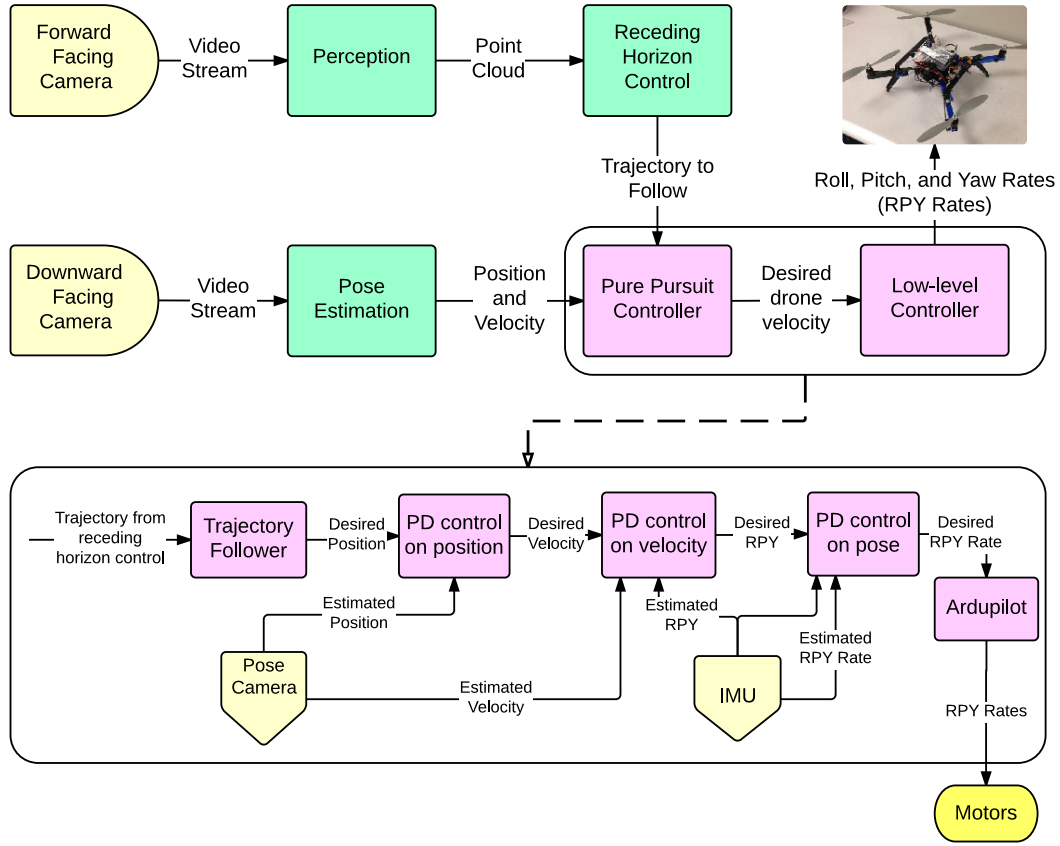


Figure 6.13: The overall flow of data and control commands between various modules. The pure pursuit trajectory follower and low-level controller (purple boxes) are shown in greater detail at the bottom.

makes tracking easier.

We evaluated the performance of the flow based tracker in motion capture and compared the true motion capture tracks to the tracks returned by flow based tracker. The resulting tracks are as shown in Figure 6.14

6.2.6 Planning and Control

Figure 6.13 shows the overall flow of data and control commands. The front camera video stream is fed to the perception module which predicts the depth of every pixel in a frame, projects it to a point cloud representation and sends it to the receding horizon control module. A trajectory library of 78 trajectories of length 5 meters is budgeted and picked from a much larger library of 2401 trajectories using the maximum dispersion algorithm by Green et al. [Green and Kelly 2006]. This is a greedy procedure for selecting trajectories, one at a time,

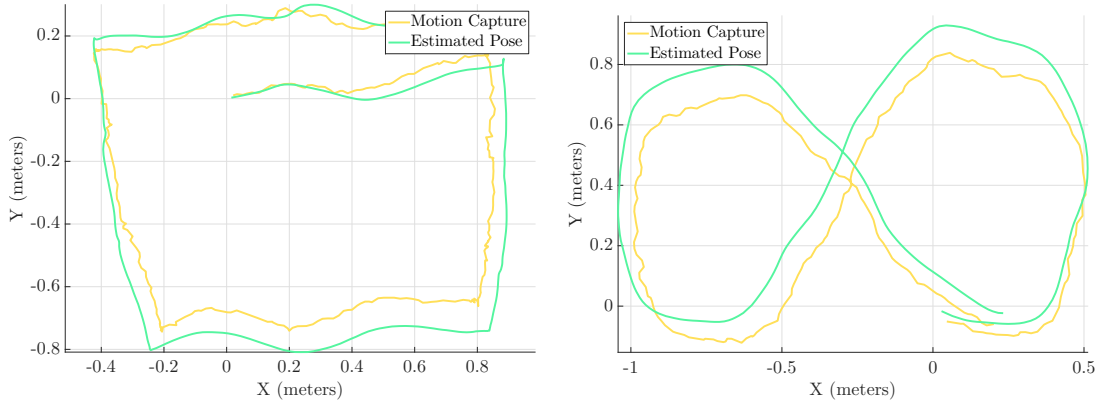


Figure 6.14: Comparison of the differential flow tracker performance vs ground truth in motion capture. Yellow tracks are the true trajectories as determined by the very accurate motion capture system, green are those determined by the algorithm. Note that due to constant replanning every 3 second, small drift in following a specific trajectory can be easily tolerated. So as long as the drift is not more than a few centimeters over a trajectory, collision avoidance is not compromised.

so that each subsequent trajectory spans maximum area between it and the rest of the trajectories. The receding horizon module maintains a score for every point in the point cloud. The score of a point decays exponentially the longer it exists. After some time when it drops below a user set threshold, the point is deleted. The decay rate is specified by setting the time constant of the decaying function. This fading memory representation of the local scene layout has two advantages: 1) It prevents collisions caused by narrow field-of-view issues where the quadrotor forgets that it has just avoided a tree, sees the next tree and dodges sideways, crashing into the just avoided tree. 2) It allows emergency backtracking maneuvers to be safely executed if required, since there is some local memory of the obstacles it has just passed.

Our system accepts a goal direction as input and ensures that the vehicle makes progress towards the goal while avoiding obstacles along the way. The score for each trajectory is the sum of three terms: 1) A sphere of the same radius as the quadrotor is convolved along a trajectory and the score of each point in collision is added up. The higher this term is relative to other trajectories, the higher the likelihood of this trajectory being in collision. 2) A term which penalizes a trajectory whose end *direction* deviates from goal direction. This is weighted by a user specified parameter. This term induces goal directed behavior and is tuned to ensure that the planner always avoids obstacles as a first priority. 3) A term which penalizes a trajectory for deviating in *translation* from the goal direction.

The pure pursuit controller module (Figure 6.13) takes in the coordinates of the trajectory to follow and the current pose of the vehicle from the optical flow based pose estimation system

(Chapter 6.2.5). We use a pure pursuit strategy [Coulter 1992] to track it. Specifically, this involves finding the closest point on the trajectory from the robot’s current estimated position and setting the target waypoint to be a certain fixed lookahead distance further along the trajectory. The lookahead distance can be tuned to obtain the desired smoothness while following the trajectory; a larger lookahead distance leads to smoother motions, at the cost of not following the trajectory exactly. Using the pose updates provided by the pose estimation module, we head towards this moving waypoint using a generic PD controller. Since the receding horizon control module continuously replans (at 5 hz) based on the image data provided by the front facing camera, we can choose to follow arbitrary lengths along a particular trajectory before switching over to the latest chosen one.

Validation of Modules

We validated each module separately as well as in tandem with other modules where each validation was progressively integrated with other modules. This helped reveal bugs and instabilities in the system.

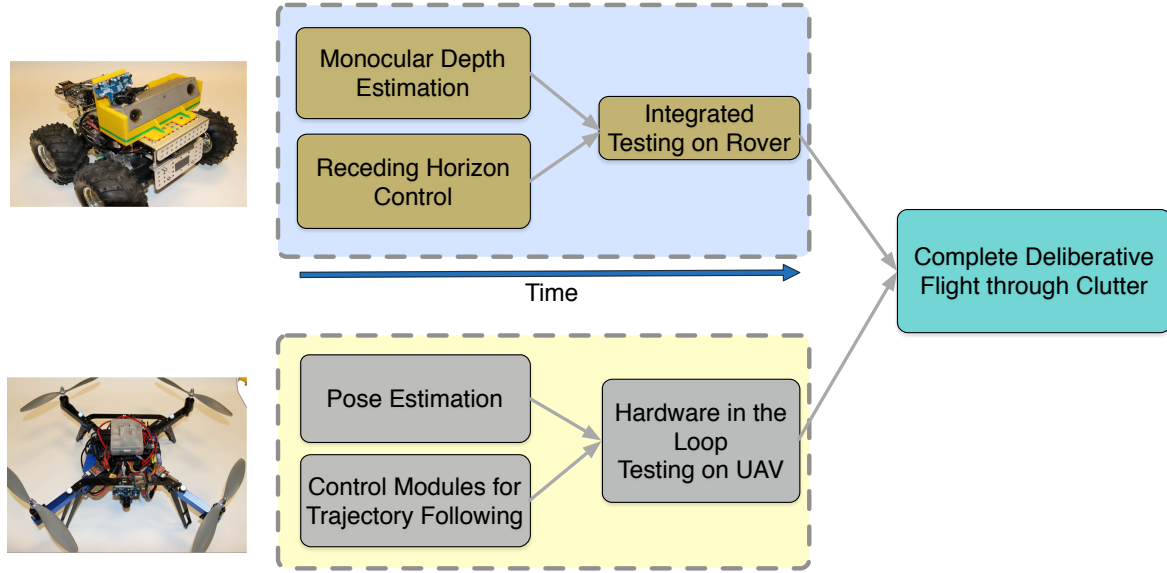


Figure 6.15: Development and testing of different modules occurred in parallel. The depth estimation and planning modules were developed and tested on the rover while the pose estimation and control modules required for following trajectories were developed and tested on the UAV. Hardware-in-the-loop (HWIL) testing of all planning and control modules was done on the UAV before perception and planning modules from the rover were integrated and then tested on the UAV for complete autonomous flight through clutter.

- *Trajectory Evaluation and Pure Pursuit Validation with Stereo Data on Rover:* We tested the trajectory evaluation and pure pursuit control module by running the entire

pipeline (other than monocular depth prediction) with stereo depth images on the rover. Figure 6.16.

- *Trajectory Evaluation and Pure Pursuit Validation with Monocular Depth on Rover:* This test is the same as above but instead of using depth images from stereo we used the monocular depth prediction. This allowed us to tune the parameters for scoring trajectories in the receding horizon module to head towards goal without colliding with obstacles.
- *Trajectory Evaluation and Pure Pursuit Validation with Known Obstacles in Motion Capture on UAV:* While testing of modules progressed on the rover we assembled and developed the pose estimation module (Chapter 6.2.5) for the UAV. We tested this module in a motion capture lab where the position of the UAV as well of the obstacles was known and updated at 120 Hz. (See Figure 6.2)
- *Trajectory Evaluation and Pure Pursuit Validation with Hardware-in-the-Loop (HWIL):* In this test we ran the UAV in an open field, fooled the receding horizon module to think it was in the midst of a point cloud and ran the whole system (except perception) to validate planning and control modules. Figure 6.17 shows an example from this setup.
- *Whole System:* After validating each module following the evaluation protocol described above, we ran the whole system end-to-end. Figure 6.1 shows an example scene of the quadrotor in full autonomous mode avoiding trees outdoors. We detail the results of collision avoidance in Section 6.3.

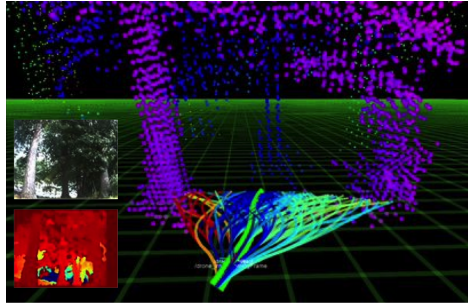


Figure 6.16: Receding horizon control validation with rover using depth images from stereo. The bright green trajectory is the currently selected trajectory to follow. Red trajectories indicate that they are more likely to be in collision.

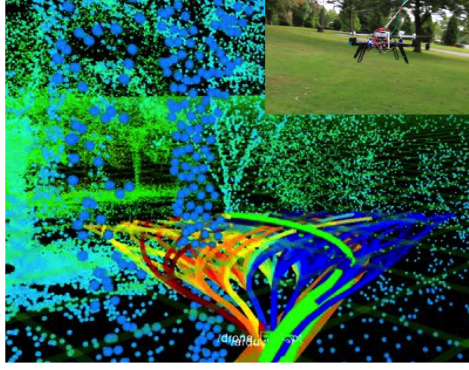


Figure 6.17: Hardware-in-the-loop testing with UAV in open field. The receding horizon module was fooled into thinking that it was in the midst of a real world point cloud while it planned and executed its way through it. This allowed us to validate planning and control without endangering the UAV.

6.3 Experiments

In this section we analyze the performance of the deliberative approach using the three depth prediction approaches detailed and discuss pros and cons of each method. All experiments were conducted in a wooded area with dense trees and a light-weight tether to the UAV for safety. Figure 6.5 shows the location of the test site (Schenley Park, near Carnegie Mellon University, Pittsburgh, PA). There is approximately 1 tree per $12 \times 12 \text{ m}^2$ in this area.

We separately evaluate performance of the perception module and the ability of the entire system to fly in dense clutter.

6.3.1 Perception Evaluation

Figure 6.18 shows the average depth error against depth images obtained from stereo processing. The average error values for non-linear regression are obtained from a held-out set of 100 images. We readily observe that direct visual odometry performs really well with low error values up to $[15, 20]$ m. Please note that direct visual odometry is not a learning based method while non-linear regression is trained on a dataset of stereo depth images. This graph nevertheless serves to show the accuracy of the visual odometry based method.

6.3.2 System Performance Evaluation

We evaluate performance by recording the average distance flown autonomously by the UAV over several runs, before an intervention. An intervention, in this context, is defined as the

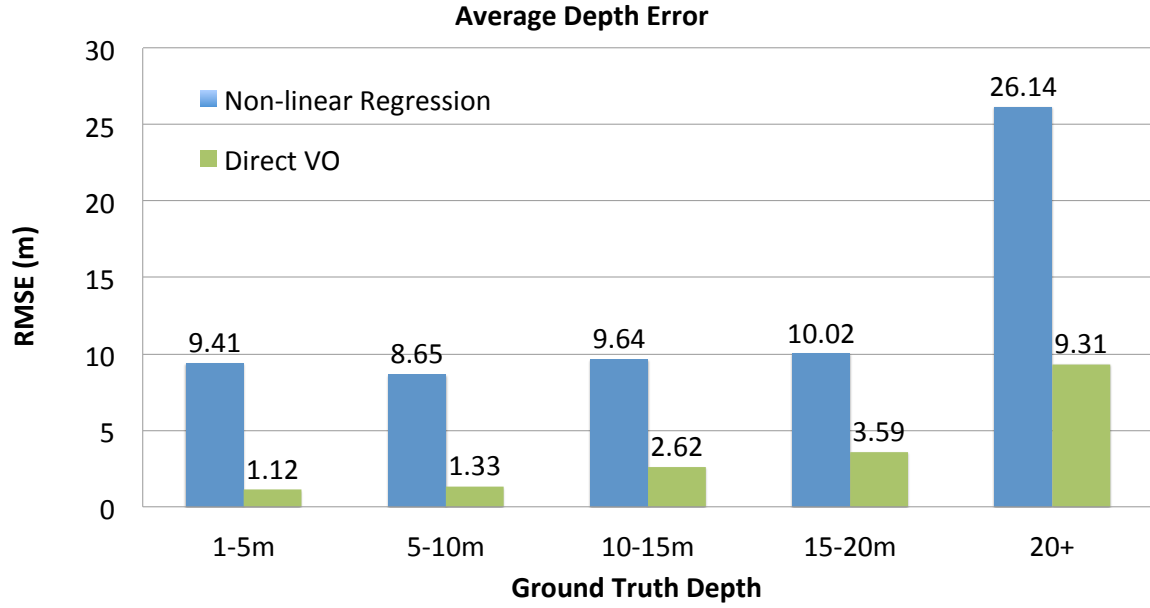


Figure 6.18: Average root-mean-squared-error (RMSE) binned over groundtruth depth buckets of $[1, 5]$ m, $[5, 10]$ m, etc. Groundtruth depth images are obtained from stereo image processing.

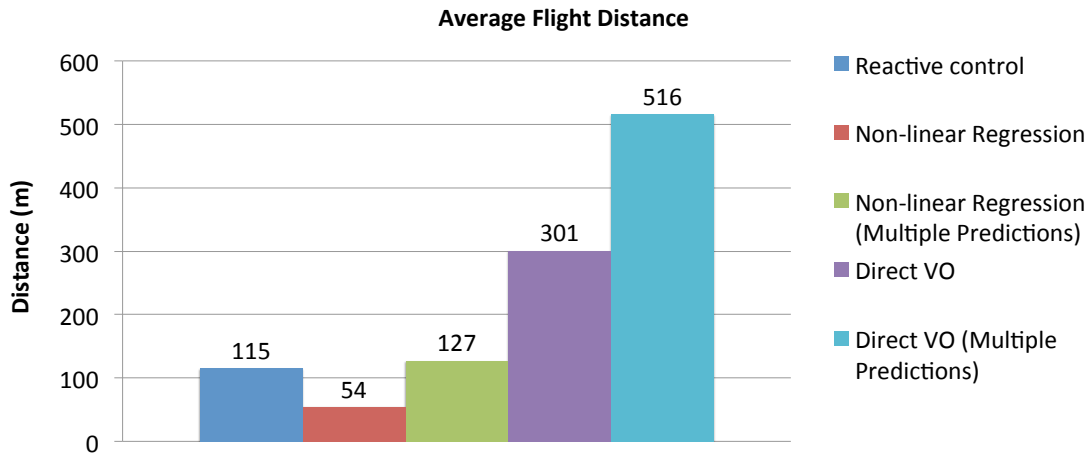


Figure 6.19: Average flight distance per intervention. For each method, the corresponding multiple prediction variant performs significantly better. We also plot the performance of the pure reactive approach from [Ross et al. 2013a] as a baseline to highlight that being deliberative significantly raises collision avoidance performance.

pilot overriding the autonomous system to prevent the UAV from an inevitable crash. Experiments were performed using both the multiple predictions approach and single best prediction, for each depth prediction method. Figure 6.19 shows the average distance traversed per intervention and Figure 6.20 shows the average number of trees avoided by the UAV per intervention using each perception method. We make several immediate observations:

- **Multiple predictions gives a significant boost to the average flight distance for each depth prediction method over corresponding single prediction approach.** In the case of non-linear regression based depth prediction, making multiple predictions gives an 135% improvement, and in the case of direct visual odometry the improvement is 71%. This validates our intuition from Chapter 6.2.4 that by avoiding a small number of extra ghost obstacles, we can significantly reduce crashes due to uncertainty.
- **Deliberative approaches perform much better than pure reactive obstacle avoidance.** This is not surprising since by definition deliberative approaches can reason further and make better decisions than reactive approaches. Direct visual odometry with multiple predictions can fly, on average 3.48 times longer than pure reactive control used in our previous work [Ross et al. 2013a].
- **Direct visual odometry based depth prediction significantly outperforms non-linear regression based depth prediction and pure reactive control.** While this is not surprising given the accuracy of the depth maps produced by direct visual odometry in Chapter 6.2.3, it is surprising that even when we are moving forward, which is the direction of least parallax, good depth maps can be realized. This can be attributed to the fact that geometric constraints provided by the ground and trees on the periphery of the field of view over multiple frames provide enough constraints for an accurate depth map.
- **Overall with our best performing approach (Direct visual odometry with multiple predictions we can fly 516 meters on average before an intervention)**

In Figure 6.21 failures are broken down by the type of obstacle the UAV failed to avoid, or whether the obstacle was not in the field-of-view (FOV) for the non-linear regression based depth prediction method (both single best and multiple prediction approaches). Overall, 39% of the failures were due to large trees and 33% on hard to perceive obstacles like branches and leaves. As expected, failures due to FOV issues are now the least contributor to overall interventions compared to the reactive control strategy [Ross et al. 2013a] which had 29.3% FOV related interventions, while deliberative approach has only 9%. This is intuitive, since the reactive control is myopic in nature and our deliberate approach helps overcome this problem

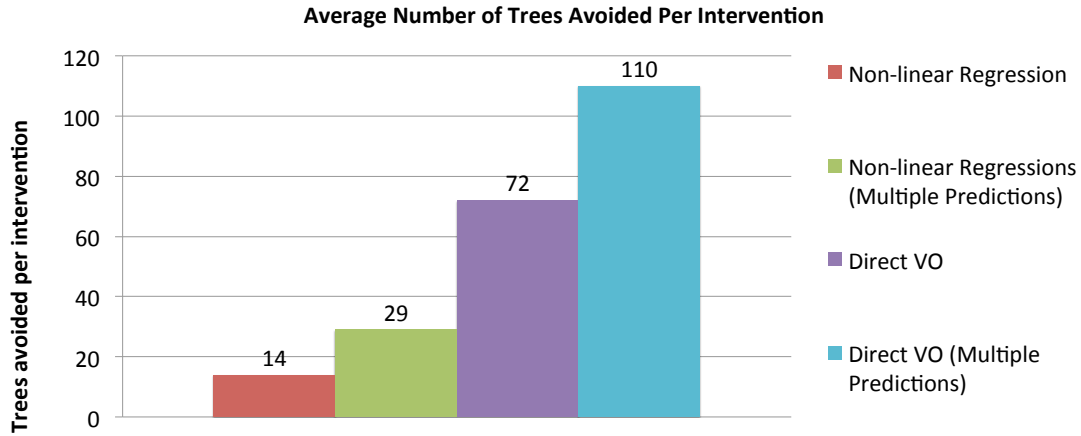


Figure 6.20: Average number of trees avoided by the UAV per intervention. This provides an additional idea of the density of the test site.

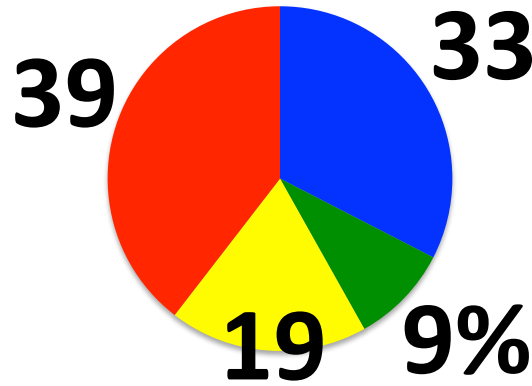


Figure 6.21: Percentage of failure for each type. Red: Large Trees, Yellow: Thin Trees, Blue: Foliage, Green: Narrow FOV.

as described in previous sections. Figure 6.22 shows some typical intervention examples.

6.4 Conclusion

While we have obtained promising results, a number of challenges remain: better handling of sudden strong wind disturbances and control schemes for leveraging the full dynamic envelope of the vehicle. In ongoing work we are moving towards complete onboard computing of all modules to reduce latency. We can leverage other sensing modes like sparse, but more accurate depth estimation from stereo, which can be used as “anchor” points to improve dense monocular depth estimation. Similarly low power, light weight lidars can be actively

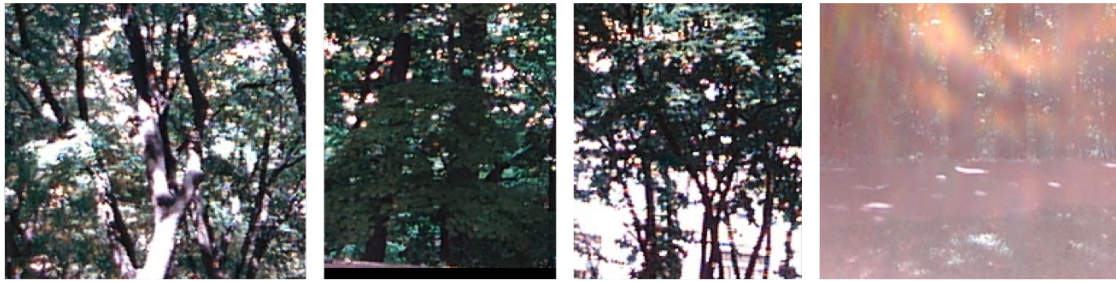


Figure 6.22: Examples of interventions: (Left) Bright trees saturated by sunlight from behind (Second from left) Thick foliage (Third from left) Thin trees (Right) Flare from direct sunlight. Camera/lens with higher dynamic range and more data of rare classes should improve performance.

foveated to high probability obstacle regions to reduce false positives and get exact depth. Another central future effort is to integrate the purely reactive [Ross et al. 2013a] approach with the deliberative scheme detailed here, for better performance.

Open Issues and Future Work

In this work we have proposed methods for predicting lists which maximize monotone submodular objectives. Recent advances in literature [Gupta et al. 2010; Buchbinder et al. 2014; Pan et al. 2014] have proposed methods for optimizing *non-monotone* submodular objectives in the context-free setting. Similar reductions as demonstrated in this work can be harnessed for contextualizing the optimization of non-monotone submodular objectives as well. This will find ready use in robotics, computer vision and general decision-making scenarios.

In all the methods presented here, we have assumed that executing the list of actions does not change the current environment. A concrete example of this is to consider the manipulation planning case study 2.1 where a list of initial seed trajectories is proposed by CONSEQOPT or SCP for evaluation by the manipulator. As the manipulator executes this list on the environment, it is possible that it will modify the environment by unintentionally moving the objects. This changes the environment and affects the probability of success of subsequent seed trajectories. If the change is large it invalidates the performance guarantees of the list.

In ongoing work we are exploring connections to adaptive submodularity [Golovin and Krause 2010] and imitation learning [Ross et al. 2011] to produce decision policies which observe the result of the current action and then suggest the next action to try.

Another limitation we would like to highlight is our methods can't propose novel actions that are not present in the library. Muelling *et al.* [Muelling et al. 2010] generalize actions already present in a motion library to synthesize new actions in an online manner which are more suited for the task at hand. By incorporating such action synthesis methods into our framework we can propose better action lists for the task at hand.

In the case of multiple structured output prediction (Section 4, it might be possible to modify the training procedure of Structured-SVMs (specifically the decoding stage) [Tsochantaridis et al. 2005; Ratliff et al. 2007c] for getting the exact same performance guarantees

as CONSEQOPT in the non-structured output case. We will investigate this direction in the near future.

Appendix

Proofs of SCP Theoretical Results

This appendix contains the proofs of the various theoretical results presented in this paper.

A.1 Preliminaries

We begin by proving a number of lemmas about monotone submodular functions, which will be useful to prove our main results.

Lemma 1. *Let \mathcal{A} be a set and f be a monotone submodular function defined on list of items from \mathcal{A} . For any lists A, B , we have that:*

$$f(A \oplus B) - f(A) \leq |B|(\mathbb{E}_{s \sim U(B)}[f(A \oplus s)] - f(A))$$

for $U(B)$ the uniform distribution on items in B .

Proof. For any list A and B , let B_i denote the list of the first i items in B , and b_i the i^{th} item in B . We have that:

$$\begin{aligned} & f(A \oplus B) - f(A) \\ &= \sum_{i=1}^{|B|} f(A \oplus B_i) - f(A \oplus B_{i-1}) \\ &\leq \sum_{i=1}^{|B|} f(A \oplus b_i) - f(A) \\ &= |B|(\mathbb{E}_{b \sim U(B)}[f(A \oplus b)] - f(A)) \end{aligned}$$

where the inequality follows from the submodularity property of f . □

Lemma 2. *Let \mathcal{A} be a set, and f a monotone submodular function defined on lists of items in \mathcal{A} . Let A, B be any lists of items from \mathcal{A} . Denote A_j the list of the first j items in A , $U(B)$ the uniform distribution on items in B and define $\epsilon_j = \mathbb{E}_{s \sim U(B)}[f(A_{j-1} \oplus s)] - f(A_j)$, the additive error term in competing with the average marginal benefits of the items in B*

when picking the j^{th} item in A (which could be positive or negative). Then:

$$f(A) \geq (1 - (1 - 1/|B|)^{|A|})f(B) - \sum_{i=1}^{|A|} (1 - 1/|B|)^{|A|-i} \epsilon_i$$

In particular if $|A| = |B| = k$, then:

$$f(A) \geq (1 - 1/e)f(B) - \sum_{i=1}^k (1 - 1/k)^{k-i} \epsilon_i$$

and for $\alpha = \exp(-|A|/|B|)$ (i.e. $|A| = |B| \log(1/\alpha)$):

$$f(A) \geq (1 - \alpha)f(B) - \sum_{i=1}^{|A|} (1 - 1/|B|)^{|A|-i} \epsilon_i$$

Proof. Using the monotone property and previous lemma 1, we must have that: $f(B) - f(A) \leq f(A \oplus B) - f(A) \leq |B|(\mathbb{E}_{b \sim U(B)}[f(A \oplus b)] - f(A))$.

Now let $\Delta_j = f(B) - f(A_j)$. By the above we have that

$$\begin{aligned} \Delta_j &\leq |B|[\mathbb{E}_{s \sim U(B)}[f(A_j \oplus s)] - f(A_j)] \\ &= |B|[\mathbb{E}_{s \sim U(B)}[f(A_j \oplus s)] - f(A_{j+1}) \\ &\quad + f(A_{j+1}) - f(B) + f(B) - f(A_j)] \\ &= |B|[\epsilon_{j+1} + \Delta_j - \Delta_{j+1}] \end{aligned}$$

Rearranging terms, this implies that $\Delta_{j+1} \leq (1 - 1/|B|)\Delta_j + \epsilon_{j+1}$. Recursively expanding this recurrence from $\Delta_{|A|}$, we obtain:

$$\Delta_{|A|} \leq (1 - 1/|B|)^{|A|} \Delta_0 + \sum_{i=1}^{|A|} (1 - 1/|B|)^{|A|-i} \epsilon_i$$

Using the definition of $\Delta_{|A|}$ and rearranging terms, we obtain $f(A) \geq (1 - (1 - 1/|B|)^{|A|})f(B) - \sum_{i=1}^{|A|} (1 - 1/|B|)^{|A|-i} \epsilon_i$. This proves the first statement of the theorem. The following two statements follow from the observations that $(1 - 1/|B|)^{|A|} = \exp(|A| \log(1 - 1/|B|)) \leq \exp(-|A|/|B|) = \alpha$. Hence $(1 - (1 - 1/|B|)^{|A|})f(B) \geq (1 - \alpha)f(B)$. When $|A| = |B|$, $\alpha = 1/e$ and this proves the special case where $|A| = |B|$. \square

For the greedy list construction strategy, the ϵ_j in the last lemma are always ≤ 0 , such that Lemma 2 implies that if we construct a list of size k with greedy, it must achieve at least 63% of the value of the optimal list of size k , but also that it must achieve at least 95% of

the value of the optimal list of size $\lfloor k/3 \rfloor$, and at least 99.9% of the value of the optimal list of size $\lfloor k/7 \rfloor$.

A more surprising fact that follows from the last lemma is that constructing a list stochastically, by sampling items from a particular fixed distribution, can provide the same guarantee as greedy:

Lemma 3. *Let \mathcal{A} be a set, and f a monotone submodular function defined on lists of items in \mathcal{A} . Let B be any list of items from \mathcal{A} and $U(B)$ the uniform distribution on elements in B . Suppose we construct the list A by sampling k items randomly from $U(B)$ (with replacement). Denote A_j the list obtained after j samples, and P_j the distribution over lists obtained after j samples. Then:*

$$\mathbb{E}_{A \sim P_k}[f(A)] \geq (1 - (1 - 1/|B|)^k)f(B)$$

In particular, for $\alpha = \exp(-k/|B|)$:

$$\mathbb{E}_{A \sim P_k}[f(A)] \geq (1 - \alpha)f(B)$$

Proof. The proof follows a similar proof to the previous lemma. Recall that by the monotone property and lemma 1, we have that for any list A : $f(B) - f(A) \leq f(A \oplus B) - f(A) \leq |B|(\mathbb{E}_{b \sim U(B)}[f(A \oplus b)] - f(A))$. Because this holds for all lists, we must also have that for any distribution P over lists A , $f(B) - \mathbb{E}_{A \sim P}[f(A)] \leq |B|(\mathbb{E}_{A \sim P}[\mathbb{E}_{b \sim U(B)}[f(A \oplus b)] - f(A)]$. Also note that by the way we construct sets, we have that $\mathbb{E}_{A_{j+1} \sim P_{j+1}}[f(A_{j+1})] = \mathbb{E}_{A_j \sim P_j}[\mathbb{E}_{s \sim U(B)}[f(A_j \oplus s)]]$

Now let $\Delta_j = f(B) - \mathbb{E}_{A_j \sim P_j}[f(A_j)]$. By the above we have that:

$$\begin{aligned} \Delta_j &\leq |B|(\mathbb{E}_{A_j \sim P_j}[\mathbb{E}_{s \sim U(B)}[f(A_j \oplus s)] - f(A_j)) \\ &= |B|(\mathbb{E}_{A_j \sim P_j}[\mathbb{E}_{s \sim U(B)}[f(A_j \oplus s)] - f(B) \\ &\quad + f(B) - f(A_j)] \\ &= |B|(\mathbb{E}_{A_{j+1} \sim P_{j+1}}[f(A_{j+1})] - f(B) \\ &\quad + f(B) - \mathbb{E}_{A_j \sim P_j}[f(A_j)]) \\ &= |B|[\Delta_j - \Delta_{j+1}] \end{aligned}$$

Rearranging terms, this implies that $\Delta_{j+1} \leq (1 - 1/|B|)\Delta_j$. Recursively expanding this recurrence from Δ_k , we obtain:

$$\Delta_k \leq (1 - 1/|B|)^k \Delta_0$$

Using the definition of Δ_k and rearranging terms we obtain $\mathbb{E}_{A \sim P_k}[f(A)] \geq (1 - (1 -$

$1/|B|)^k)f(B)$. The second statement follows again from the fact that $(1 - (1 - 1/|B|)^k)f(B) \geq (1 - \alpha)f(B)$ \square

Corollary 3. *There exists a distribution that when sampled k times to construct a list, achieves an approximation ratio of $(1 - 1/e)$ of the optimal list of size k in expectation. In particular, if A^* is an optimal list of size k , sampling k times from $U(A^*)$ achieves this approximation ratio. Additionally, for any $\alpha \in (0, 1]$, sampling $\lceil k \log(1/\alpha) \rceil$ times must construct a list that achieves an approximation ratio of $(1 - \alpha)$ in expectation.*

Proof. Follows from the last lemma using $B = A^*$. \square

This surprising result can also be seen as a special case of a more general result proven in prior related work that analyzed randomized set selection strategies to optimize submodular functions (Lemma 2.2 in [Feige et al. 2011]).

A.2 Proofs of Main Results

We now provide the proofs of the main results in this paper. We provide the proofs for the more general contextual case where we learn over a hypothesis class $\tilde{\Pi}$. All the results for the context-free case can be seen as special cases of these results when $\Pi = \tilde{\Pi} = \{\psi | a \in \mathcal{A}\}$ and $\psi(\text{computeFeatures}(S, \mathbf{d})) = a$ for any example \mathbf{d} and list S .

We refer the reader to the notation defined in Chapter 2 and 3 for the definitions of the various terms used.

Theorem 1. *Let $\alpha = \exp(-N/K)$ and $K' = \min(N, K)$. After T iterations, for any $\delta, \delta' \in (0, 1)$, we have that with probability at least $1 - \delta$:*

$$F(\bar{\psi}, N) \geq (1 - \alpha)F(S_{\psi, K}^*) - \frac{R}{T} - 2\sqrt{\frac{2 \ln(1/\delta)}{T}}$$

and similarly, with probability at least $1 - \delta - \delta'$:

$$F(\bar{\psi}, N) \geq (1 - \alpha)F(S_{\psi, K}^*) - \frac{\mathbb{E}[R]}{T} - \sqrt{\frac{2K' \ln(1/\delta')}{T}} - 2\sqrt{\frac{2 \ln(1/\delta)}{T}}$$

Proof.

$$\begin{aligned}
& F(\bar{\psi}, N) \\
&= \frac{1}{T} \sum_{t=1}^T F(\psi_t, N) \\
&= \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{S_{\psi, N} \sim \psi_t} [\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, N}, \mathbf{d})]] \\
&= (1 - \alpha) \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, K}^*, \mathbf{d})] \\
&\quad - (1 - \alpha) \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, K}^*, \mathbf{d})] \\
&\quad - \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{S_{\psi, N} \sim \psi_t} [\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, N}, \mathbf{d})]]
\end{aligned}$$

Now consider the sampled examples $\{d_t\}_{t=1}^T$ and the hypotheses $\psi_{t,i}$ sampled i.i.d. from ψ_t to construct the lists $\{S_t\}_{t=1}^T$ and denote the random variables $X_t = (1 - \alpha)(\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, K}^*, \mathbf{d})] - f(S_{\psi, K}^*, \mathbf{d}_t)) - \mathbb{E}_{S_{\psi, N} \sim \psi_t} [\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, N}, \mathbf{d})]] - f(S_t, \mathbf{d}_t)$. If ψ_t is deterministic, then simply consider all $\psi_{t,i} = \psi_t$. Because the \mathbf{d}_t are i.i.d. from \mathcal{D} , and the distribution of hypotheses used to construct S_t only depends on $\{\mathbf{d}_\tau\}_{\tau=1}^{t-1}$ and $\{S_\tau\}_{\tau=1}^{t-1}$, then the X_t conditioned on $\{X_\tau\}_{\tau=1}^{t-1}$ have expectation 0, and because $f \in [0, 1]$ for all $d \in \mathcal{D}$, X_t can vary in a range $r \subseteq [-2, 2]$. Thus the sequence of random variables $Y_t = \sum_{i=1}^t X_i$, for $t = 1$ to T , forms a martingale where $|Y_t - Y_{t+1}| \leq 2$. By the Azuma-Hoeffding's inequality, we have that $P(Y_T/T \geq \epsilon) \leq \exp(-\epsilon^2 T/8)$. Hence for any $\delta \in (0, 1)$, we have that with probability at least $1 - \delta$, $Y_T/T \leq 2\sqrt{\frac{2 \ln(1/\delta)}{T}}$. Hence we have that with probability at least $1 - \delta$:

$$\begin{aligned}
& F(\bar{\psi}, N) \\
&= (1 - \alpha) \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, K}^*, \mathbf{d})] \\
&\quad - [(1 - \alpha) \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, K}^*, \mathbf{d})] \\
&\quad - \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{S_{\psi, N} \sim \psi_t} [\mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, N}, \mathbf{d})]]] \\
&= (1 - \alpha) \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, K}^*, \mathbf{d})] \\
&\quad - [(1 - \alpha) \frac{1}{T} \sum_{t=1}^T f(S_{\psi, K}^*, \mathbf{d}_t) \\
&\quad - \frac{1}{T} \sum_{t=1}^T f(S_t, \mathbf{d}_t)] - Y_T/T \\
&= (1 - \alpha) \mathbb{E}_{\mathbf{d} \sim \mathcal{D}} [f(S_{\psi, K}^*, \mathbf{d})] \\
&\quad - [(1 - \alpha) \frac{1}{T} \sum_{t=1}^T f(S_{\psi, K}^*, \mathbf{d}_t) \\
&\quad - \frac{1}{T} \sum_{t=1}^T f(S_t, \mathbf{d}_t)] - 2\sqrt{\frac{2 \ln(1/\delta)}{T}}
\end{aligned}$$

Let $w_i = (1 - 1/K)^{N-i}$. From Lemma 2, we have:

$$\begin{aligned}
& (1 - \alpha) \frac{1}{T} \sum_{t=1}^T f(S_{\psi,K}^*, \mathbf{d}_t) - \frac{1}{T} \sum_{t=1}^T f(S_t, \mathbf{d}_t) \\
& \leq \frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N w_i (\mathbb{E}_{\psi \sim U(S_{\psi,K}^*)} [f(S_{t,i-1} \oplus \psi(S_{t,i-1}, \mathbf{d}_t))] \\
& \quad - f(S_{t,i}, \mathbf{d}_t)) \\
& = \mathbb{E}_{\psi \sim U(S_{\psi,K}^*)} \left[\frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N w_i (f(S_{t,i-1} \oplus \psi(S_{t,i-1}, \mathbf{d}_t)) \right. \\
& \quad \left. - f(S_{t,i}, \mathbf{d}_t)) \right] \\
& \leq \max_{\psi \in \Pi} \left[\frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N w_i (f(S_{t,i-1} \oplus \psi(S_{t,i-1}, \mathbf{d}_t)) \right. \\
& \quad \left. - f(S_{t,i}, \mathbf{d}_t)) \right] \\
& \leq \max_{\psi \in \tilde{\Pi}} \left[\frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N w_i (f(S_{t,i-1} \oplus \psi(S_{t,i-1}, \mathbf{d}_t)) \right. \\
& \quad \left. - f(S_{t,i}, \mathbf{d}_t)) \right] \\
& = R/T
\end{aligned}$$

Hence combining with the previous result proves the first part of the theorem.

Additionally, for the sampled examples $\{d_t\}_{t=1}^T$ and the hypotheses $\psi_{t,i}$, consider the random variables $Q_{N(t-1)+i} = w_i \mathbb{E}_{\psi \sim \psi_t} [\mathbf{f}(S_{t,i-1} \oplus \psi(S_{t,i-1}, \mathbf{d}_t))] - w_i f(S_{t,i}, \mathbf{d}_t)$. Because each draw of $\psi_{t,i}$ is i.i.d. from ψ_t , we have that again the sequence of random variables $Z_j = \sum_{i=1}^j Q_i$, for $j = 1$ to TN forms a martingale and because each Q_i can take values in a range $[-w_j, w_j]$ for $j = 1 + \text{mod}(i-1, N)$, we have $|Z_i - Z_{i-1}| \leq w_j$. Since $\sum_{i=1}^{TN} |Z_i - Z_{i-1}|^2 \leq T \sum_{i=1}^N (1 - 1/K)^{2(N-i)} \leq T \min(K, N) = TK'$, by Azuma-Hoeffding's inequality, we must have that $P(Z_{TN} \geq \epsilon) \leq \exp(-\epsilon^2/2TK')$. Thus for any $\delta' \in (0, 1)$, with probability at least $1 - \delta'$, $Z_{TN} \leq \sqrt{2TK' \ln(1/\delta')}$. Hence combining with the previous result, it must be the case that with probability at least $1 - \delta - \delta'$, both $Y_T/T \leq 2\sqrt{\frac{2 \ln(1/\delta)}{T}}$ and $Z_{TN} \leq \sqrt{2TK' \ln(1/\delta')}$ holds.

Now note that:

$$\begin{aligned}
& \max_{\psi \in \tilde{\Pi}} \left[\frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N w_i (f(S_{t,i-1} \oplus \psi(S_{t,i-1}, \mathbf{d}_t)) - f(S_{t,i}, \mathbf{d}_t)) \right] \\
& = \max_{\psi \in \tilde{\Pi}} \left[\frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N w_i (\mathbf{f}(S_{t,i-1} \oplus \psi(S_{t,i-1}, \mathbf{d}_t)) \right. \\
& \quad \left. - \mathbb{E}_{\psi' \sim \psi_t} [f(S_{t,i-1} \oplus \psi'(S_{t,i-1}, \mathbf{d}_t))] \right] + Z_{TN}/T \\
& = \mathbb{E}[R]/T + Z_{TN}/T
\end{aligned}$$

Using this additional fact, and combining with previous results we must have that with

probability at least $1 - \delta - \delta'$:

$$\begin{aligned}
& F(\bar{\psi}, N) \\
& \geq (1 - \alpha)F(S_{\psi, K}^*) - [(1 - \alpha)\frac{1}{T} \sum_{t=1}^T f(S_{\psi, K}^*, \mathbf{d}_t) \\
& \quad - \frac{1}{T} \sum_{t=1}^T f(S_t, \mathbf{d}_t)] - 2\sqrt{\frac{2\ln(1/\delta)}{T}} \\
& \geq (1 - \alpha)F(S_{\psi, K}^*) - \mathbb{E}[R]/T - Z_{TN}/T - 2\sqrt{\frac{2\ln(1/\delta)}{T}} \\
& \geq (1 - \alpha)F(S_{\psi, K}^*) - \mathbb{E}[R]/T - \sqrt{\frac{2K'\ln(1/\delta')}{T}} \\
& \quad - 2\sqrt{\frac{2\ln(1/\delta)}{T}}
\end{aligned}$$

□

We now show that the expected regret must grow with $\sqrt{K'}$ and not K' , when using Weighted Majority with the optimal learning rate (or with the doubling trick).

Corollary 1. *Under the event where Theorem 1 holds (the event that occurs w.p. $1 - \delta - \delta'$), if $\tilde{\Pi}$ is a finite set of hypotheses, using Weighted Majority with the optimal learning rate guarantees that after T iterations:*

$$\begin{aligned}
\mathbb{E}[R]/T & \leq \frac{4K'\ln|\tilde{\Pi}|}{T} + 2\sqrt{\frac{K'\ln|\tilde{\Pi}|}{T}} \\
& \quad + 2^{9/4}(K'/T)^{3/4}(\ln(1/\delta'))^{1/4}\sqrt{\ln|\tilde{\Pi}|}
\end{aligned}$$

For large enough T in $\Omega(K'(\ln|\tilde{\Pi}| + \ln(1/\delta')))$, we obtain that:

$$\mathbb{E}[R]/T \leq O\left(\sqrt{\frac{K'\ln|\tilde{\Pi}|}{T}}\right)$$

Proof. We use a similar argument as Streeter & Golovin in Lemma 4 in [Streeter and Golovin 2008] to bound $\mathbb{E}[R]$ in the result of Theorem 1. Consider the sum of the benefits accumulated by the learning algorithm at position i in the list, for $i \in 1, 2, \dots, N$, i.e. let $y_i = \sum_{t=1}^T b(\psi_{t,i}(S_{t,i-1}, \mathbf{d}_t) | S_{t,i-1}, \mathbf{d}_t)$, where $\psi_{t,i}$ corresponds to the particular sampled hypothesis by Weighted Majority for choosing the item at position i , when constructing the list S_t for example \mathbf{d}_t . Note that $\sum_{i=1}^N (1 - 1/K)^{N-i} y_i \leq \sum_{i=1}^N y_i \leq T$ by the fact that the monotone submodular function f is bounded in $[0, 1]$ for all examples \mathbf{d} . Now consider the sum of the benefits you could have accumulated at position i , had you chosen the best fixed hypothesis in hindsight to construct the entire list, keeping the hypothesis fixed as the list is constructed, i.e. let $z_i = \sum_{t=1}^T b(\psi^*(S_{t,i-1}, \mathbf{d}_t) | S_{t,i-1}, \mathbf{d}_t)$, for $\psi^* = \arg \max_{\psi \in \tilde{\Pi}} \sum_{i=1}^N (1 - 1/K)^{N-i} \sum_{t=1}^T b(\psi(S_{t,i-1}, \mathbf{d}_t) | S_{t,i-1}, \mathbf{d}_t)$ and let $r_i = z_i - y_i$. Now denote $Z = \sqrt{\sum_{i=1}^N (1 - 1/K)^{N-i} z_i}$. We have $Z^2 = \sum_{i=1}^N (1 - 1/K)^{N-i} z_i = \sum_{i=1}^N (1 - 1/K)^{N-i} (y_i + r_i) \leq T + R$, where R is the sample regret incurred by the learning algorithm. Under the event where theorem 1 holds (i.e. the event that occurs with probability at least

$1 - \delta - \delta'$), we had already shown that $R \leq \mathbb{E}[R] + Z_{TN}$, for $Z_{TN} \leq \sqrt{2TK' \ln(1/\delta')}$, in the second part of the proof of theorem 1. Thus when theorem 1 holds, we have $Z^2 \leq T + \sqrt{2TK' \ln(1/\delta')} + \mathbb{E}[R]$. Now using the generalized version of weighted majority with rewards (i.e. using directly the benefits as rewards) [Arora et al. 2012], since the rewards at each update are in $[0, K']$, we have that with the best learning rate in hindsight ¹: $\mathbb{E}[R] \leq 2Z\sqrt{K' \ln |\tilde{\Pi}|}$. Thus we obtain $Z^2 \leq T + \sqrt{2TK' \ln(1/\delta')} + 2Z\sqrt{K' \ln |\tilde{\Pi}|}$. This is a quadratic inequality of the form $Z^2 - 2Z\sqrt{K' \ln |\tilde{\Pi}|} - T - \sqrt{2TK' \ln(1/\delta')} \leq 0$, with the additional constraint $Z \geq 0$. This implies Z is less than or equal to the largest non-negative root of the polynomial $Z^2 - 2Z\sqrt{K' \ln |\tilde{\Pi}|} - T - \sqrt{2TK' \ln(1/\delta')}$. Solving for the roots, we obtain

$$\begin{aligned} Z &\leq \sqrt{K' \ln |\tilde{\Pi}|} + \sqrt{K' \ln |\tilde{\Pi}| + T + \sqrt{2TK' \ln(1/\delta')}} \\ &\leq 2\sqrt{K' \ln |\tilde{\Pi}|} + \sqrt{T} + (2TK' \ln(1/\delta'))^{1/4} \end{aligned}$$

Plugging back Z into the expression $\mathbb{E}[R] \leq 2Z\sqrt{K' \ln |\tilde{\Pi}|}$, we obtain:

$$\begin{aligned} \mathbb{E}[R] &\leq 4K' \ln |\tilde{\Pi}| + 2\sqrt{TK' \ln |\tilde{\Pi}|} \\ &\quad + 2(2T \ln(1/\delta'))^{1/4} (K')^{3/4} \sqrt{\ln |\tilde{\Pi}|} \end{aligned}$$

Thus the average regret:

$$\begin{aligned} \frac{\mathbb{E}[R]}{T} &\leq \frac{4K' \ln |\tilde{\Pi}|}{T} + 2\sqrt{\frac{K' \ln |\tilde{\Pi}|}{T}} \\ &\quad + 2^{9/4} (K'/T)^{3/4} (\ln(1/\delta'))^{1/4} \sqrt{\ln |\tilde{\Pi}|} \end{aligned}$$

For T in $\Omega(K'(\ln \tilde{\Pi} + \ln(1/\delta')))$, the dominant term is $2\sqrt{\frac{K' \ln |\tilde{\Pi}|}{T}}$, and thus $\frac{\mathbb{E}[R]}{T}$ is $O(\sqrt{\frac{K' \ln |\tilde{\Pi}|}{T}})$. \square

Corollary 2. *Let $\alpha = \exp(-N/K)$ and $K' = \min(N, K)$. If we run an online learning algorithm on the sequence of convex loss C_t instead of ℓ_t , then after T iterations, for any $\delta \in (0, 1)$, we have that with probability at least $1 - \delta$:*

$$F(\bar{\psi}, N) \geq (1 - \alpha)F(S_{\psi, K}^*) - \frac{\tilde{R}}{T} - 2\sqrt{\frac{2 \ln(1/\delta)}{T}} - \mathcal{G}$$

where \tilde{R} is the regret on the sequence of convex loss C_t , and $\mathcal{G} = \frac{1}{T}[\sum_{t=1}^T (\ell_t(\psi_t) - C_t(\psi_t)) + \min_{\psi \in \tilde{\Pi}} \sum_{t=1}^T C_t(\psi) - \min_{\psi' \in \tilde{\Pi}} \sum_{t=1}^T \ell_t(\psi')]$ is the “convex optimization gap” that measures how close the surrogate losses C_t is to minimizing the cost-sensitive losses ℓ_t .

Proof. Follows immediately from Theorem 1 using the definition of R , \tilde{R} and \mathcal{G} , since $\mathcal{G} =$

¹if not a doubling trick can be used to get the same regret bound within a small constant factor [Cesa-Bianchi et al. 1997]

$$\frac{R-\tilde{R}}{T}$$

□

Proof of Monotone Submodularity of Quality Function

Consider the quality function which scores a set of predictions for an input image by the loss of the best prediction in the set. Using the same notation in Chapter 4, the quality function is reproduced as:

$$f(Y_S(I), \mathbf{y}_{gt}) = \max_{i \in 1, \dots, N} \{g(\tilde{h}_i(I), \mathbf{y}_{gt})\}, \quad (\text{B.1})$$

$$= 1 - \min_{i \in 1, \dots, N} \{l(\tilde{h}_i(I), \mathbf{y}_{gt})\} \quad (\text{B.2})$$

The above equation scores the sequence of structured predictions $Y_S(I) = \{\tilde{h}_i(I)\}_{i \in 1 \dots N}$ by the score of the best prediction produced by the predictors $S = \{\tilde{h}_1, \tilde{h}_2, \dots, \tilde{h}_N\}$. Such a function f was proved to be monotone, submodular by Dey *et al.* in [Dey et al. 2012]. We reproduce the proof here for convenience while adapting the exposition to the specific usage in our case:

A set function f which maps subsets $A \subseteq \mathcal{A}$ of a finite set \mathcal{A} to the real numbers. f is called submodular if, for all $A \subseteq B \subseteq \mathcal{A}$ and $S \in \mathcal{A} \setminus B$ it holds that

$$f(A \oplus S) - f(A) \geq f(B \oplus S) - f(B) \quad (\text{B.3})$$

where \oplus is the concatenation operator. Such a function is monotone if it holds that for any sets $S_1, S_2 \in \mathcal{A}$, we have

$$f(S_1) \leq f(S_1 \oplus S_2) \quad (\text{B.4})$$

$$f(S_2) \leq f(S_1 \oplus S_2)$$

We want to prove that f (B.2) is monotone, submodular. We make B.2 more general by replacing the loss of a particular prediction $l(h_i(I))$ with $\text{cost}(a_i)$ where a_i is a particular item. The simplified equation is:

$$f \equiv 1 - \min_{a_i \in \mathcal{A}} \{\text{cost}(a_i)\} \quad (\text{B.5})$$

where \mathcal{A} is the set of allowed items.

This can be proved if $\min_{a_i \in \mathcal{A}} \{\text{cost}(a_i)\}$ is monotone supermodular. A function f is supermodular if it holds that

$$f(A \oplus S) - f(A) \leq f(B \oplus S) - f(B) \quad (\text{B.6})$$

Theorem 2. *The function $\min_{a_i \in \mathcal{A}} \{\text{cost}(a_i)\}$ is monotone, supermodular where a_i are predictions.*

Proof. Submodularity: Assume that we are given sets $A \subseteq B \subseteq \mathcal{A}$, $S \in \mathcal{A} \setminus B$. We want to prove the inequality in B.6. Let $R = B \setminus A$, the set of elements that are in B but not in A . Since $A \oplus R = B$ we can now rewrite B.6 as

$$f(A \oplus S) - f(A) \leq f(A \oplus R \oplus S) - f(A \oplus R) \quad (\text{B.7})$$

We refer to the left and right sides of B.7 as LHS and RHS respectively. Define a^* as the prediction which has the least cost. Hence there can be three cases:

- Case 1: $a^* \in A$ In this case $LHS = RHS = 0$
- Case 2: $a^* \in R$ In this case $RHS \geq LHS$
- Case 3: $a^* \in S$ In this case $RHS \geq LHS$

Since in all possible cases it can be seen that RHS is greater than or equal to LHS it is proved that $\min_{a_i \in \mathcal{A}} \{\text{cost}(a_i)\}$ is supermodular. Note that if there are multiple predictions which have the same minimum cost as a^* then similar arguments still hold and even in the worst case when they are distributed across S , R and A , Case 1 holds.

Monotonicity Consider two sequences S_1 and S_2 . Define a^* as the predictions which has the least cost. We want to prove that $\min_{a_i \in \mathcal{A}} \{\text{cost}(a_i)\}$ is monotone decreasing, i.e.

$$\begin{aligned} f(S_1) &\geq f(S_1 \oplus S_2) \\ f(S_2) &\geq f(S_1 \oplus S_2) \end{aligned} \quad (\text{B.8})$$

There are three possible cases:

- Case 1: $a^* \in S_1 \implies f(S_1) = f(S_1 \oplus S_2)$ and $f(S_2) \geq f(S_1 \oplus S_2)$
- Case 2: $a^* \in S_2 \implies f(S_1) \geq f(S_1 \oplus S_2)$ and $f(S_2) = f(S_1 \oplus S_2)$
- Case 3: $a^* \in S_1 \oplus S_2 \implies f(S_1) = f(S_1 \oplus S_2)$ and $f(S_2) = f(S_1 \oplus S_2)$

Since in all possible cases the conditions in B.8 are satisfied $\min_{a_i \in \mathcal{A}} \{\text{cost}(a_i)\}$ is monotone decreasing. \square

Corollary 3. *The function f of Equation 4.3 in the paper is monotone, submodular due to $\min_{a_i \in \mathcal{A}} \{\text{cost}(a_i)\}$ being monotone, supermodular by Theorem 2.*

Corollary 4. *The function $F(S, \mathcal{D}) = \mathbb{E}_{(I, \mathbf{y}_{gt}) \sim \mathcal{D}} [f(Y_S(I), \mathbf{y}_{gt})]$ (Equation 4.4 in the paper) is also monotone submodular since non-negative sums of monotone submodular functions is also monotone submodular.*

Bibliography

- Agarwal, A., S. M. Kakade, N. Karampatziakis, L. Song, and G. Valiant
2013. Least squares revisited: Scalable approaches for multi-class prediction. *arXiv preprint arXiv:1310.1949*. 6.2.2
- Alina Beygelzimer, J. L. and B. Zadrozny
2009. Tutorial on Reductions in Machine Learning. http://hunch.net/~reductions_tutorial. 2.2.2
- Ardupilot
2015. <http://dev.ardupilot.com>. 6.1.1, 6.1.2
- Arora, S., E. Hazan, , and S. Kale
2012. The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing*, 8:121–164. A.2
- Auer, P., N. Cesa-Bianchi, Y. Freund, and R. Schapire
2003. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77. 3, 3, 3
- Bachrach, A., R. He, and N. Roy
2009. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217–228. 6
- Batra, D., A. Kowdle, D. Parikh, J. Luo, and T. Chen
2010. icoseg: Interactive co-segmentation with intelligent scribble guidance. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, Pp. 3169–3176. IEEE. 4.2, 4.3.2
- Batra, D., P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich
2012. Diverse m-best solutions in markov random fields. In *Computer Vision–ECCV 2012*, Pp. 1–16. Springer. 1, 2.1, 4, 4.1, 4.3, 4.3.1, 6.2.4
- Beyeler, A., J.-C. Zufferey, and D. Floreano
2009. Vision-based control of near-obstacle flight. *Autonomous robots*, 27(3):201–219. 1, 6.2.2

- Beygelzimer, A., V. Dani, T. Hayes, J. Langford, and B. Zadrozny
 2005. Error limiting reductions between classification tasks. In *Proceedings of the 22nd international conference on Machine learning*, Pp. 49–56. ACM. [2.2.2](#), [3](#)
- Bishop, C. M. et al.
 2006. *Pattern recognition and machine learning*, volume 4. Springer New York. [1](#)
- Bouguet, J.-Y.
 2004. Camera calibration toolbox for matlab. [6.1.1](#)
- Buchbinder, N., M. Feldman, J. S. Naor, and R. Schwartz
 2014. Submodular maximization with cardinality constraints. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, Pp. 1433–1452. SIAM. [7](#)
- Buehler, M., K. Iagnemma, and S. Singh
 2008. Special issue on the 2007 darpa urban challenge, part i, ii, iii. *Journal of Field Robotics*. [6](#)
- Carbonell, J. and J. Goldstein
 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, Pp. 335–336. ACM. [1](#), [1](#)
- Carreira, J., R. Caseiro, J. Batista, and C. Sminchisescu
 2012. Semantic segmentation with second-order pooling. In *Computer Vision–ECCV 2012*, Pp. 430–443. Springer. [4.3.3](#)
- Carreira, J. and C. Sminchisescu
 2010. Constrained parametric min-cuts for automatic object segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, Pp. 3241–3248. IEEE. [4](#)
- Caruana, R., A. Niculescu-Mizil, G. Crew, and A. Ksikes
 2004. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, P. 18. ACM. [4](#)
- Cesa-Bianchi, N., Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth
 1997. How to use expert advice. *Journal of the ACM*, 44(3):427–485. [1](#)
- Coulter, R. C.
 1992. Implementation of the pure pursuit path tracking algorithm. Technical report, DTIC Document. [6.1.2](#), [6.2.6](#)

- Dafttry, S., C. Hoppe, and H. Bischof
 2015. Building with drones: Accurate 3d facade reconstruction using mavs. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. 6.2.3
- Dalal, N. and B. Triggs
 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, Pp. 886–893. IEEE. 6.2.2
- Dang, H. T.
 2005. Overview of duc 2005. In *Proceedings of the Document Understanding Conference*. 3.2.3
- Daumé Iii, H., J. Langford, and D. Marcu
 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325. 5
- Davies, E. R.
 2004. *Machine vision: theory, algorithms, practicalities*. Elsevier. 6.2.2
- Davison, A. J., I. D. Reid, N. D. Molton, and O. Stasse
 2007. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067. 6.2.3
- DeLong, A., A. Osokin, H. N. Isack, and Y. Boykov
 2012. Fast approximate energy minimization with label costs. *International journal of computer vision*, 96(1):1–27. 4.3.3
- Dey, D., C. Geyer, S. Singh, and M. Digioia
 2011. A cascaded method to detect aircraft in video imagery. *The International Journal of Robotics Research*, P. 0278364911412807. 6
- Dey, D., T. Y. Liu, M. Hebert, and J. A. Bagnell
 2013. Contextual sequence prediction with application to control library optimization. *Robotics*, P. 49. 4.2
- Dey, D., T. Y. Liu, B. Sofman, and J. A. Bagnell
 2012. Efficient optimization of control libraries. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*. 1, 3, 4.2, B
- Divvala, S. K., D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert
 2009. An empirical study of context in object detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, Pp. 1271–1278. IEEE. 6.2.2

- Dragan, A., G. Gordon, and S. Srinivasa
 2011. Learning from experience in manipulation planning: Setting the right goals. In *International Symposium of Robotics Research*. 2.1
- Engel, J., T. Schöps, and D. Cremers
 2014. Lsd-slam: Large-scale direct monocular slam. In *Computer Vision–ECCV 2014*, Pp. 834–849. Springer. 6.2.3
- Engel, J., J. Sturm, and D. Cremers
 2013. Semi-dense visual odometry for a monocular camera. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, Pp. 1449–1456. IEEE. 6.2.3, 6.2.3, 6.2.4
- Everingham, M., L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman
 . The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>. 4.3.3
- Farneback, G.
 2003. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, Pp. 363–370. Springer. 6.2.2
- Feige, U.
 1998. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652. 2.2.1, 3.1
- Feige, U., L. Lovász, and P. Tetali
 2004. Approximating min sum set cover. *Algorithmica*, 40(4):219–234. 2.2.1, 2.2.1
- Feige, U., V. S. Mirrokni, and J. Vondrak
 2011. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153. A.1
- Felzenszwalb, P. F. and D. A. McAllester
 2007. The generalized a* architecture. *J. Artif. Intell. Res.(JAIR)*, 29:153–190. 4
- Freund, Y., R. Schapire, and N. Abe
 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612. 4.2
- Geyer, C., T. Templeton, M. Meingast, and S. S. Sastry
 2006. The recursive multi-frame planar parallax algorithm. In *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, Pp. 17–24. IEEE. 6.2.3

- Girshick, R., J. Donahue, T. Darrell, and J. Malik
 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, Pp. 580–587. IEEE. [4](#)
- Golovin, D. and A. Krause
 2010. Adaptive submodularity: A new approach to active learning and stochastic optimization. In *COLT*, Pp. 333–345. [1](#), [7](#)
- Gould, S., O. Russakovsky, I. Goodfellow, P. Baumstarck, A. Y. Ng, and D. Koller
 2010. The stair vision library (v2.4). [\(LINK\)](#). [4.3.2](#)
- Green, C. and A. Kelly
 2006. Optimal sampling in the space of paths: Preliminary results. Technical Report CMU-RI-TR-06-51, Robotics Institute, Pittsburgh, PA. [6.2.6](#)
- Gupta, A., A. Roth, G. Schoenebeck, and K. Talwar
 2010. Constrained non-monotone submodular maximization: Offline and secretary algorithms. *CoRR*, abs/1003.1517. [\(LINK\)](#). [7](#)
- Guzman-Rivera, A., D. Batra, and P. Kohli
 2012. Multiple choice learning: Learning to produce multiple structured outputs. In *Advances in Neural Information Processing Systems*, Pp. 1799–1807. [4](#), [4.1](#), [4.3](#), [4.3.2](#)
- Guzman-Rivera, A., P. Kohli, D. Batra, and R. Rutenbar
 2014a. Efficiently enforcing diversity in multi-output structured prediction. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, Pp. 284–292. [4.1](#)
- Guzman-Rivera, A., P. Kohli, B. Glocker, J. Shotton, T. Sharp, A. Fitzgibbon, and S. Izadi
 2014b. Multi-output learning for camera relocalization. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, Pp. 1114–1121. IEEE. [1](#), [4.1](#), [4.3](#), [4.3.1](#), [4.3.2](#)
- Hariharan, B., P. Arbeláez, R. Girshick, and J. Malik
 2014. Simultaneous detection and segmentation. In *Computer Vision–ECCV 2014*, Pp. 297–312. Springer. [4.3.3](#)
- Harris, C. and M. Stephens
 1988. A combined corner and edge detector. In *Alvey vision conference*, volume 15, P. 50. Manchester, UK. [6.2.2](#)

- Hazan, E., A. Agarwal, and S. Kale
 2007. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192. [2](#)
- Helgason, S.
 1980. Support of radon transforms. *Advances in Mathematics*. [6.2.2](#)
- Hoiem, D., A. A. Efros, and M. Hebert
 2005. Geometric context from a single image. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, Pp. 654–661. IEEE. [6.2.2](#)
- Honegger, D., L. Meier, P. Tanskanen, and M. Pollefeys
 2013. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, Pp. 1736–1741. IEEE. [6.2.5](#)
- Horvitz, E.
 2001. Principles and applications of continual computation. *Artificial Intelligence*, 126(1):159–196. [1](#)
- Hu, H., A. Grubb, J. A. Bagnell, and M. Hebert
 2014. Efficient feature group sequencing for anytime linear prediction. *arXiv:1409.5495*. [6.2.2](#), [6.8](#)
- Jetchev, N. and M. Toussaint
 2009. Trajectory prediction: learning to map situations to robot trajectories. In *Proceedings of the 26th annual international conference on machine learning*, Pp. 449–456. ACM. [2.1](#)
- Joachims, T.
 2005. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, Pp. 377–384. ACM. [3](#)
- Kalai, A. and S. Vempala
 2005. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307. [3](#)
- Kalal, Z., K. Mikolajczyk, and J. Matas
 2012. Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1409–1422. [4](#)
- Kavraki, L. E., P. Svestka, J.-C. Latombe, and M. H. Overmars
 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580. [2.1](#)

- Kelly, A., A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, et al.
2006. Toward reliable off road autonomous vehicles operating in challenging environments. *The International Journal of Robotics Research*, 25(5-6):449–483. 6
- Khuller, S., A. Moss, and J. Naor
1999. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45. 2.2.1
- Klein, G. and D. Murray
2007. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, Pp. 225–234. IEEE. 6.2.3, 6.2.5
- Knepper, R. A. and M. T. Mason
2009. Path diversity is only part of the problem. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, Pp. 3224–3229. IEEE. 6
- Kohli, P., A. Osokin, and S. Jegelka
2013. A principled deep random field model for image segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, Pp. 1971–1978. IEEE. 4
- Krause, A. and C. Guestrin
2008. Beyond convexity: Submodularity in machine learning. *ICML Tutorials*, 4. 1
- Kuffner, J. J. and S. M. LaValle
2000. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, Pp. 995–1001. IEEE. 2.1, 2.1
- Kulesza, A. and B. Taskar
2010. Structured determinantal point processes. In *Advances in neural information processing systems*, Pp. 1171–1179. 4, 5
- Kulesza, A. and B. Taskar
2011. Learning determinantal point processes. In *In Proceedings of Uncertainty in Artificial Intelligence*. 3.2.3, 3.2.3, 9, 4.1, 5
- Lafferty, J., A. McCallum, and F. C. Pereira
2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 4

- Langford, J. and A. Beygelzimer
 2005. Sensitive error correcting output codes. In *Learning Theory*, Pp. 158–172. Springer. 2, 2.2.2
- Langford, J., L. Li, and A. Strehl
 2007. Vowpal Wabbit. 6.2.2
- Langford, J., R. Oliveira, and B. Zadrozny
 2012. Predicting conditional quantiles via reduction to classification. *arXiv preprint arXiv:1206.6860*. 5
- Langford, J. and B. Zadrozny
 2005. Estimating class membership probabilities using classifier learners. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, Pp. 198–205. 5
- Li, C. and K. M. Kitani
 2013. Pixel-level hand detection in ego-centric videos. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, Pp. 3570–3577. IEEE. 6.7c, 6.2.2
- LidarLite
 2015. <http://pulsedlight3d.com/products/lidar-lite>. 6.1.2, 6.2.3, 6.2.5
- Lin, C.-Y.
 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, Pp. 74–81. 3.2.3
- Lin, H. and J. Bilmes
 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Pp. 912–920. Association for Computational Linguistics. 3.2.3, 3.2.3
- Lin, H. and J. Bilmes
 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Pp. 510–520. Association for Computational Linguistics. 3.2.3
- Littlestone, N. and M. K. Warmuth
 1994. The weighted majority algorithm. *Information and computation*, 108(2):212–261. 3, 3, 3.1, 2

- Long, J., E. Shelhamer, and T. Darrell
 2014. Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1411.4038*. 4.3.3
- Manning, C. D., P. Raghavan, and H. Schütze
 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge. 1
- Matthies, L., R. Brockers, Y. Kuwata, and S. Weiss
 2014. Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, Pp. 3242–3249. IEEE. 1
- Misra, I., A. Shrivastava, and M. Hebert
 2014. Data-driven exemplar model selection. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, Pp. 339–346. IEEE. 4
- Montiel, J., J. Civera, and A. J. Davison
 2006. Unified inverse depth parametrization for monocular slam. *analysis*, 9:1. 6.2.3
- Muelling, K., J. Kober, and J. Peters
 2010. Learning table tennis with a mixture of motor primitives. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, Pp. 411–416. IEEE. 7
- Munagala, K., S. Babu, R. Motwani, and J. Widom
 2005. The pipelined set cover problem. In *Database Theory-ICDT 2005*, Pp. 83–98. Springer. 1, 2.2.1
- Munoz, D., J. A. Bagnell, and M. Hebert
 2010. Stacked hierarchical labeling. In *Computer Vision–ECCV 2010*, Pp. 57–70. Springer. 4, 4.3.2, 4.3.3
- Nemhauser, G., L. Wolsey, and M. Fisher
 1978. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294. 2.2.1, 4, 4.2
- Newcombe, R. A., S. J. Lovegrove, and A. J. Davison
 2011. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, Pp. 2320–2327. IEEE. 6.2.5
- Nilsson, D.
 1998. An efficient algorithm for finding the m most probable configurations in probabilistic expert systems. *Statistics and Computing*, 8(2):159–173. 4.1

Odroid

2015. http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127. 6.1.2

Oliva, A. and A. Torralba

2007. The role of context in object recognition. *Trends in cognitive sciences*, 11(12):520–527. 6.2.2

Pan, X., S. Jegelka, J. E. Gonzalez, J. K. Bradley, and M. I. Jordan

2014. Parallel double greedy submodular maximization. In *Advances in Neural Information Processing Systems*, Pp. 118–126. 7

Park, D. and D. Ramanan

2011. N-best maximal decoders for part models. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, Pp. 2627–2634. IEEE. 1, 4, 4.1, 4.3, 4.3.1

Pirsiavash, H., D. Ramanan, and C. C. Fowlkes

2011. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, Pp. 1201–1208. IEEE. 1

Prasad, A., S. Jegelka, and D. Batra

2014. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. In *Advances in Neural Information Processing Systems*, Pp. 2645–2653. 4.3.3

Quigley, M., K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng

2009. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*. 6.1.2

Radlinski, F., R. Kleinberg, and T. Joachims

2008. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, Pp. 784–791. ACM. 1, 7, 4

Ratliff, N., J. Bagnell, and S. Srinivasa

2007a. Imitation learning for locomotion and manipulation. Technical Report CMU-RI-TR-07-45, Robotics Institute, Pittsburgh, PA. 2.1

Ratliff, N. D., J. A. Bagnell, and M. Zinkevich

2007b. (approximate) subgradient methods for structured prediction. In *International Conference on Artificial Intelligence and Statistics*, Pp. 380–387. 2

- Ratliff, N. D., J. A. Bagnell, and M. Zinkevich
 2007c. (approximate) subgradient methods for structured prediction. In *International Conference on Artificial Intelligence and Statistics*, Pp. 380–387. 7
- Ross, S., G. J. Gordon, and D. Bagnell
 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, Pp. 627–635. 3, 2, 5, 7
- Ross, S., N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert
 2013a. Learning monocular reactive uav control in cluttered natural environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, Pp. 1765–1772. IEEE. 6, 6, 1, 6.2.2, 6.2.2, 6.19, 6.3.2, 6.3.2, 6.4
- Ross, S., J. Zhou, Y. Yue, D. Dey, and D. Bagnell
 2013b. Learning policies for contextual submodular prediction. In *Proceedings of The 30th International Conference on Machine Learning*, Pp. 1364–1372. 1
- Saxena, A., S. H. Chung, and A. Y. Ng
 2005. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*, Pp. 1161–1168. 6.2.2, 6.2.4
- Scherer, S., S. Singh, L. Chamberlain, and M. Elgersma
 2008. Flying fast and low among obstacles: Methodology and experiments. *The International Journal of Robotics Research*, 27(5):549–574. 6
- Schmid, K., P. Lutz, T. Tomić, E. Mair, and H. Hirschmüller
 2014. Autonomous vision-based micro air vehicle for indoor and outdoor navigation. *Journal of Field Robotics*, 31(4):537–570. 1
- Srinivasan, M. V.
 2011. Visual control of navigation in insects and its relevance for robotics. *Current opinion in neurobiology*, 21(4):535–543. 1, 6.2.2
- Streeter, M. and D. Golovin
 2008. An online algorithm for maximizing submodular functions. In *Advances in Neural Information Processing Systems*, Pp. 1577–1584. 1, 2.2.1, 2.2.1, 2.2.2, 3, 3.1, 3.1, 5, A.2
- Stühmer, J., S. Gumhold, and D. Cremers
 2010. Real-time dense geometry from a handheld camera. In *Pattern Recognition*, Pp. 11–20. Springer. 6.2.3

Taskar, B., C. Guestrin, and D. Koller

2003. Max-margin markov networks. In *Advances in Neural Information Processing Systems*, P. None. 4

Templeton, T.

2009. Accurate real-time reconstruction of distant scenes using computer vision: The recursive multi-frame planar parallax algorithm. 6.2.3

Tomasi, C. and T. Kanade

1991. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. 6.2.5

Tsochantaridis, I., T. Joachims, T. Hofmann, and Y. Altun

2005. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, Pp. 1453–1484. 4, 4.1, 7

Viola, P. and M. Jones

2001. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, Pp. I–511. IEEE. 4

Wedel, A., T. Pock, C. Zach, H. Bischof, and D. Cremers

2009. An improved algorithm for tv-l 1 optical flow. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, Pp. 23–45. Springer. 6.2.2

Wendel, A., M. Maurer, G. Graber, T. Pock, and H. Bischof

2012. Dense reconstruction on-the-fly. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, Pp. 1450–1457. IEEE. 1, 6.2.3

Wu, B., T. L. Ooi, and Z. J. He

2004. Perceiving distance accurately by a directional process of integrating ground information. *Nature*, 428(6978):73–77. 6.2.2

Yang, Y. and D. Ramanan

2011. Articulated pose estimation with flexible mixtures-of-parts. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, Pp. 1385–1392. IEEE. 4, 4.3.1

Yanover, C. and Y. Weiss

2004. Finding the ai most probable configurations using loopy belief propagation. *Advances in neural information processing systems*, 16:289. 4.1

Yue, Y. and C. Guestrin

2011. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems*, Pp. 2483–2491. 3.2.2, 5

Yue, Y. and T. Joachims

2008. Predicting diverse subsets using structural svms. In *Proceedings of the 25th international conference on Machine learning*, Pp. 1224–1231. ACM. 5

Zadrozny, B., J. Langford, and N. Abe

2003. Cost-sensitive learning by cost-proportionate example weighting. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, Pp. 435–442. IEEE. 5

Zhou, J., S. Ross, Y. Yue, D. Dey, and J. A. Bagnell

2013. Knapsack constrained contextual submodular list prediction with application to multi-document summarization. *CoRR*, abs/1308.3541. (LINK). 8

Zucker, M.

2009. A data-driven approach to high level planning. Technical Report CMU-RI-TR-09-42, Robotics Institute, Pittsburgh, PA. 2.1

Zucker, M., N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa

2013. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193. 2.1, 3.2.1