Visual Utility

A Framework for Focusing Computer Vision Algorithms

Mark Desnoyer

CMU-RI-TR-15-32

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

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

December 2015

Thesis Committee:
David Wettergreen, Chair
Martial Hebert
Yaser Sheikh
James J. Clark, McGill University

Abstract

The real world is a rich environment, fraught with complexity. To be robust in this complex environment, computer Vision algorithms that operate in Unstructured Environments (VUE) tend to use large amounts of data or complex modeling. Unfortunately, these algorithms also require significant computational resources.

In this thesis, we examine a *visual utility* framework that we show is used regularly in an ad hoc manner. This framework uses visual utility estimators to speed up VUE algorithms with minimal performance degradation by focusing those higher level algorithms on the most relevant imagery. We formally define this framework, show that it has a submodular structure and discover under what conditions using it is valuable in practice. We find that visual utility approaches are most effective when using fast, task specific visual utility estimators and the VUE task is computationally expensive,

We also introduce SCATAT, a cascade building algorithm. SCATAT takes advantage of the submodular structure of the visual utility framework in order to build a near optimal cascade that trades off task performance and processing requirements explicitly. We then validate this algorithm in an experimental case study on object detection.

Finally, in theoretical case studies, we prove that two seminal cascade algorithms are special cases of our visual utility framework. We show they optimize a submodular visual utility function, explaining their high observed performance in practice.

Acknowledgments

This research was supported in part by the Spark Fund, Caterpillar Inc., Bombardier Inc. and the Robotics Institute of Carnegie Mellon University.

This thesis has taken longer than most. There were many dead ends along the way and many starts and stops. Without the support of many people, it would not have been possible.

First, I'd like to thank my advisor David Wettergreen. We tackled many fun problems together and without his support from the start, to ABD and back again, I wouldn't be here. I would also like to thank the rest of my committee, Yaser Sheikh, Martial Hebert and James J. Clark for being a solid sounding board and keeping me honest.

I would also like to thank all of my colleagues on the various projects throughout the journey that expanded my horizons. Starting with the X-prize team of Red Whittaker, John Thornton, Erika Bannon, Kel Guerrin, Dave Duggins, Rama Manur, Kevin Peterson and Nick Miller who reached for the stars. Ben Grocholsky and Paul Rybski rode the rails to explore automating the Bombardier trains. Brad Kriel and Dominic Jonak help study detection paradigms in construction environments. Finally, the Reefbot team of Michael Furlong, John Thornton, Scott Moreland, Ashley Kidd, Justine Kaznica and Bob Snowden made the many long hours at the aquarium exciting.

Next, I must thank the Neon team and board who have supported me to finish this journey in these last two years. There are many pressures and deadlines inherent when building a new company, but a culture with a dedication to incessant growth and learning made finishing this possible.

Of course, I must also mention Mike Kim, a dear friend who encouraged me with some wise words to *Just Finish*.

My family has also been crucial during my educational journey. From early years in a French immersion grade school all the way to defending a thesis in another country 26 years later, they have always been there for me through thick or thin. Merci Beaucoup.

Last, but not least, thank you to my wife Catherine. She put up with everything from the typical request for chores to typing away on this document during our honeymoon in Patagonia. I could not have finished without all of her love and support throughout the years, not to mention her insistence on me being addressed as Mr. Mark Desnoyer, while she can be Dr. Catherine Izard. I love you.

Contents

1 Introduction **1**

1.1 Motivation 1

1.1.1 Ever-Growing Computational Complexity of Computer Vision 1

1.1.2 Visual Search in Biologic Systems 2

1.1.3 Computer Vision != Biologic Vision 2

1.2 Visual Utility 3

1.3 Thesis Statement 4

2 Literature Review **7**

2.1 Eye Movements and How They Filter Data 8

2.2 Computational Motivation for Foveal Vision 9

2.3 Physical Implementations of Foveal Cameras 10

2.4 Visual Utility Models of the Brain 14

2.4.1 Social Robotics 14

2.4.2 Saliency 16

2.5 Generic Visual Utility Estimators 19

2.5.1 Estimators Called Saliency but Really Aren't 19

2.5.2 Objectness 21

2.6 Applications of Saliency and Other General Visual Utility Estimators 22

2.6.1 Active Vision 23

2.6.2 Reactive Vision 27

2.7 Effectiveness of Previous Visual Utility Estimators 29

2.7.1 Generic Visual Utility Measures 30

2.7.2 Task Specific Visual Utility Measures 31

2.7.3 Very Expensive Tasks 33

2.7.4 Summary of Effective Visual Utility Estimators 34

2.8 Selective Computation For Object Detection 34

2.8.1 Cascades . . . 34
2.8.2 Submodular Approaches to Speed Up Object Detection . . . 35
2.8.3 Other Approaches To Fast Object Detection . . . 36
2.9 Submodularity . . . 36
2.9.1 Submodular Minimization . . . 36
2.9.2 Submodular Maximization . . . 37
2.10 Summary . . . 37

3 Theory **39**
3.1 Defining Visual Utility . . . 39
3.2 Visual Utility for a Single Frame . . . 40
3.2.1 Setting Relative Weights . . . 40
3.3 Metrics in Detection Tasks . . . 42
3.3.1 Precision and Recall . . . 42
3.3.2 ROC Curve . . . 43
3.4 Building a Visual Utility Filter . . . 43
3.4.1 Choosing a Visual Utility Filter . . . 45
3.4.2 Cascades (AND operation) . . . 46
3.4.3 Submodularity in Cascades . . . 49
3.4.4 Parallel Filters (OR operation) . . . 52
3.5 Adaptive Submodularity . . . 54
3.5.1 Visual Utility and Adaptive Submodularity . . . 54
3.6 Summary . . . 57

4 Theoretical Case Studies **59**
4.1 Viola-Jones Cascade Detector . . . 59
4.1.1 Details of the Viola-Jones Cascade Detector . . . 59
4.1.2 Mapping to the Visual Utility Framework . . . 61
4.1.3 Theoretical Analysis . . . 61
4.1.4 Discussion . . . 62
4.2 Star Cascade for Deformable Parts Models . . . 63
4.2.1 Details of the Star Cascade . . . 63
4.2.2 Mapping to the Visual Utility Framework . . . 64
4.2.3 Theoretical Analysis . . . 65
4.2.4 Discussion . . . 67

5 SCATAT Algorithm **69**

5.1 Minimizing the Difference of Two Submodular Functions . . . 69
5.2 SCATAT Algorithm . . . 70
5.2.1 Minimizing the Supermodular Function . . . 71
5.2.2 Modular Approximation . . . 71
5.3 Conclusion . . . 73

6 Visual Utility in Deep Neural Networks **75**
6.1 Introduction . . . 75
6.2 Visual Utility Filtering . . . 76
6.3 Integrated Region Evaluation . . . 77
6.4 Adaptive Searching . . . 78
6.5 Building Near Optimal Networks . . . 79
6.5.1 Hierarchical Modular Optimization . . . 79
6.5.2 Layer-wise Learning . . . 80
6.6 Conclusions . . . 80

7 Experimental Case Study in Object Detection **81**
7.1 Problem Description . . . 81
7.2 Experimental Framework . . . 82
7.2.1 Datasets . . . 82
7.2.2 Evaluation Criteria . . . 83
7.3 Experiments . . . 84
7.3.1 A Range of Visual Utility Estimators . . . 84
7.3.2 Applying Multiple Object Detectors . . . 92
7.3.3 SCATAT Algorithm . . . 94
7.3.4 Impact of Non-Maximal Suppression . . . 100
7.4 Summary . . . 102

8 Conclusions **107**
8.1 Contributions . . . 108
8.2 Future Work . . . 109
8.3 Parting Thoughts . . . 109

9 Bibliography **111**

A Reefbot **129**
A.1 Introduction . . . 129

A.2 Related Work . . . 130
A.3 Robotic Platform . . . 131
A.4 Marine Species Classification . . . 131
A.4.1 Dataset . . . 132
A.4.2 Fish Extraction . . . 132
A.4.3 Fish Classification . . . 134
A.5 Evaluation . . . 135
A.5.1 Quality of Segmentation . . . 135
A.5.2 Impact of Segmentation Quality . . . 135
A.5.3 Feature Types . . . 137
A.6 Conclusions . . . 137
A.7 Acknowledgements . . . 138

List of Figures

1.1 Processing time of object detection algorithms in the last 15 years . . . 2
1.2 Comparing cost and computation capacity of GPUs and CPUs . . . 3
1.3 Visual utility conceptual framework . . . 4

2.1 Typical human fixation pattern. . . . 9
2.2 Log-polar mapping. . . . 11
2.3 Various proposed multifoveal geometries. . . . 12
2.4 Bandera's method of mapping a foveal sensor. . . . 13
2.5 A selection of social robots. . . . 15
2.6 Model of saliency from [129]. . . . 17
2.7 Examples of visual utility maps. . . . 19
2.8 Example of a next-best-view 3D reconstruction. . . . 24
2.9 Example of how excess data can degrade classification. . . . 28
2.10 Generic visual utility cascade. . . . 34

3.1 Typical Precision-Recall and ROC curves. . . . 42
3.2 Base case of no visual utility filtering. . . . 43
3.3 A simple visual utility filter . . . 44
3.4 Typical ROC curve for detection tasks . . . 47
3.5 Combining two visual utility filters in an AND operation . . . 48
3.6 Combining two visual utility filters in an OR operation . . . 52
3.7 A toy motivation for parallel cascades. . . . 53
3.8 Alignment is crucial in adaptive search . . . 56

4.1 Example Haar like features. . . . 60
4.2 A generic objection detection cascade. . . . 61
4.3 The star cascade for a deformable parts model. . . . 65

6.1 Visual utility filtering for deep neural networks . . . 76

6.2 Architecture of a detection deep neural network . . . 77
6.3 Glimpse extraction with a deep neural network . . . 78

7.1 Correlation between the AP of pedestrian and object detection tasks . . . 82
7.2 Examples from the Heimonen HIMA dataset. . . . 83
7.3 Examples from the ETH dataset. . . . 83
7.4 Rahtu's boundary edge feature. . . . 87
7.5 F-Score vs. Speedup when using visual utility with the HOG detector. . . . 91
7.6 Precision vs. Speedup when using visual utility with the HOG detector. . . . 91
7.7 Recall vs. Speedup when using visual utility with the HOG detector. . . . 91
7.8 The precision recall curves of different visual utility estimators. . . . 92
7.9 Number of windows processed after visual utility filtering. . . . 94
7.10 Number of object detectors needed to be as fast as the resampling baseline. . . . 95
7.11 F-Score vs. Speedup for the SCATAT algorithm compared to other visual utility estimators. 98
7.12 Precision vs. Speedup for the SCATAT algorithm compared to other visual utility estimators. 98
7.13 Recall vs. Speedup for the SCATAT algorithm compared to other visual utility estimators. 99
7.14 Precision vs. Recall when using visual utility with the HOG detector including building an integral HOG cascade using SCATAT. . . . 99
7.15 F-Score vs. Speedup for the SCATAT algorithm compared to the integral HOG cascade from [334]. . . . 100
7.16 Precision vs. Speedup for the SCATAT algorithm compared to the integral HOG cascade from [334]. . . . 101
7.17 Recall vs. Speedup for the SCATAT algorithm compared to the integral HOG cascade from [334]. . . . 101
7.18 Precision vs. Recall for the SCATAT algorithm compared to the integral HOG cascade from [334]. . . . 102
7.19 The F-score vs. speedup when using non-maximal suppression on the ETH dataset . . . 103
7.20 The PR curve when using non-maximal suppression on the ETH dataset . . . 103
7.21 Correlation between ground truth and two detectors. . . . 104

8.1 Using context in an image increases computational complexity. . . . 108

A.1 The Reefbot robot. Image ©Paul A. Selvaggio . . . 129
A.2 The Reefbot exhibit console. Image ©Paul A. Selvaggio . . . 131
A.3 An example frame from Reefbot. The scene can have many fish, they can appear in arbitrary orientations and the lighting makes them difficult to identify. . . . 132
A.4 Summary of the number of instances for each species in the human and machine extracted datasets . . . 133

A.5 Histogram of the sizes of the two datasets. The automatically extracted fish patches are significantly smaller than the human labeling because it identifies less of the fish. 135

A.6 Precision Recall curve of the fish extractor. Even though we are only using motion, we are still able to achieve over 70% precision while still being able to extract 20% of the fish. . . . 136

A.7 Histogram of the sizes of the two datasets. The automatically extracted fish patches are significantly smaller than the human labeling because it identifies less of the fish. 136

A.8 The accuracy of the classifier for different fractions of queries returning confident results. Results are shown for both the machine and human datasets 137

List of Tables

2.1 Applications where visual utility approaches have been applied. 38

5.1 Requirements of the modular approximation function. 70

7.1 CPU time of different visual utility estimators. 90
7.2 Average precision of different visual utility estimators 93
7.3 Computation time for calculating the integral HOG feature. 96

Chapter 1

Introduction

In my field, dreams are easy to come by.

Chris Hadfield, Canadian Astronaut

1.1 Motivation

1.1.1 Ever-Growing Computational Complexity of Computer Vision

Applying vision techniques to real world problems can be challenging. The real world is a rich environment, fraught with complexity. To be robust in this complex environment, computer vision algorithms that operate in unstructured environments tend to use large amounts of data or complex modeling. Even in the simplest case, an algorithm must operate on all of the pixels in an image, while the trend in vision is to increase the computational complexity further. For example, fig. 1.1 shows the processing time of algorithms proposed for the well studied problem of object detection. For this problem, the computational time has been increasing exponentially despite Moore's law as the algorithms become more complex and are able to handle more object categories. In problems like scene understanding, pose estimation and others, newer techniques rely on modeling the relationships between portions of images and objects, adding extra dimensions to the search space [113]. Other approaches require many searches into a large scale database, which further increases computation time and memory requirements [258].

Moore's law is not sufficient to overcome this increasing complexity because the same process that increases the transistor density on chips is increasing the number of pixels in camera sensors. Sometimes, for applications where latency is important, we can compensate for this extra complexity using GPUs or by parallelizing across many machines. However, this does not change the underlying computational costs and GPUs are not orders of magnitude cheaper than CPUs. This is shown in fig. 1.2 where even though the computational capacity of a single chip is high in GPU instances, the overall cost per teraflop-hour is only 10-20% cheaper. Another approach is to identify structures in specific algorithms that allow them to be reformulated to run in sub-linear time, but this is not always possible and often requires massive research and engineering effort. Our only other option is to limit the amount of data that must be processed in the first place. It is this approach that we will explore in this thesis. In particular, we will start by examining a visual system that effectively limits the amount of data processed, while still enabling complex inference and operations: the biological visual system.

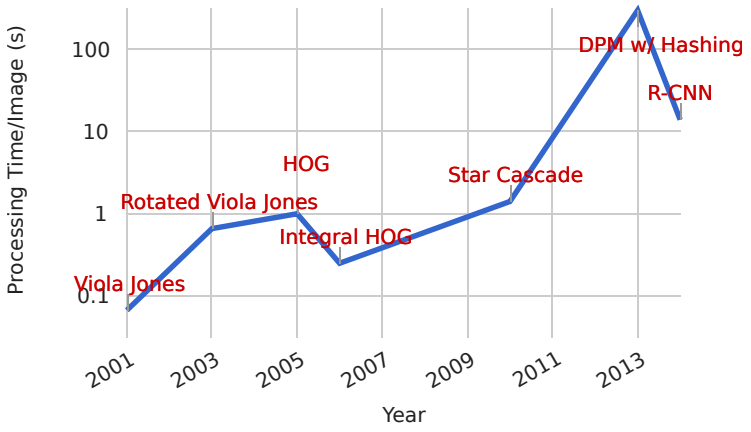


Figure 1.1: The processing time of object detection algorithms over the last 15 years as reported by their authors. The computation time has been increasing exponentially over the last 15 years despite Moore's law. The data points in order are from [298], [141], [60], [334], [84], [66], [97].

1.1.2 Visual Search in Biologic Systems

Except for some gastropods, every animal on Earth with eyes has a foveal visual system. This system combines a low resolution, wide field of view region in the eye, with a high resolution, narrow field of view. In humans, these two regions work together to produce the visual grasp reflex [254] where a stimulus in the periphery causes the eye to direct the foveal region to the area of interest in order to obtain more information [169]. The eye will then gaze on that location for a short time before moving on.

There is evidence that the foveal visual system is optimized for three things. First, motion blur is minimized. As the eye is gazing at a location, the image is stabilized, while during the fast saccades, the blurred imagery is ignored by the brain [171]. Second, the effective resolution of the eye is increased because as the eye is gazing, it dithers slightly, enabling the creation of a super-resolution image [44]. Finally, the amount of processing required is minimized. It is this last characteristic that we will examine throughout this thesis.

We can see that biologic systems reduce the amount of processing required simply by examining the retinotropic mapping of the visual system. Each area in the retina can be mapped directly to a specific portion of the visual cortex where it is processed. Though the fovea only covers 1% of the retina, it maps to over 50% of the visual cortex [147]. Therefore, our brain would have to be significantly larger if our retina were covered with photoreceptors at the same density as it is in the fovea.

A larger brain would require more energy to maintain and from an evolutionary point of view, would thus be undesirable. Therefore, we can infer that the foveal visual system enables an organism to efficiently filter the rich visual world and distill it to that information that is most useful for the organism's survival. In this thesis, we question whether computer vision systems can use a similar approach to operate efficiently.

1.1.3 Computer Vision != Biologic Vision

There are significant structural differences in computer vision systems and biologic vision systems. First, computer vision systems use a camera with a sensor that has a uniform resolution across the array, while biologic systems have a foveal camera. Some foveal cameras have been built (section 2.3), but any computer vision algorithm must be reformulated to the different spatial mapping. It would be ideal not to have to rebuild over 30 years of computer vision research that has created algorithms for cartesian sensors.

The second major difference between computer and biological vision systems is that at the low level,

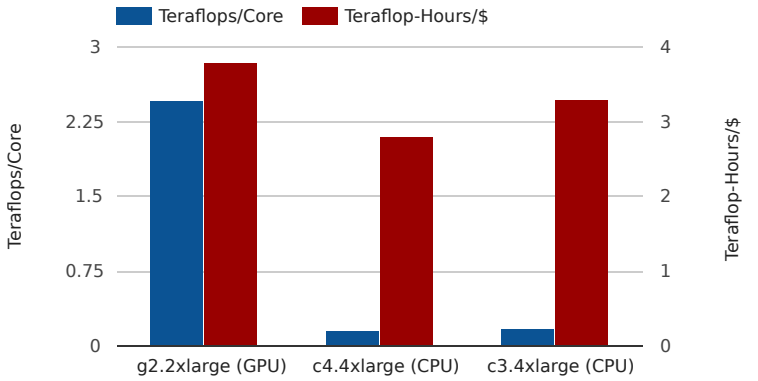


Figure 1.2: The computational capacity and cost for different GPU and CPU instances on Amazon Web Services as of September 2015. Even though GPUs have a lot of computational capacity per chip, the overall capacity per dollar is only 10-20% better than CPUs.

the biologic vision system is extremely parallelized. In the the V1, the first stage of the visual cortex, each small spatial region in the eye is attached to different neurons that fire on very simple cues, like Gabor filters [63]. As signals travel further along the visual cortex, the spatial extent associated with each neuron increases. Furthermore, there are multiple feedback mechanisms throughout the visual cortex where activity in the higher level regions will modulate responses in the lower regions [122]. In contrast, most computer vision systems still execute on sequential processors primarily using a pipeline architecture. This is slowly changing with the use of GPUs and other highly parallel co-processor architectures. These architectures enable simple, local features to be calculated very quickly in parallel across a visual frame, while higher level reasoning must execute on a sequential processor.

The last major difference is that for many applications, the vision system cannot actively control the camera to capture specific information. Instead, it is inundated with whatever visual data is available and must react to it. In contrast, biologic systems have the ability to predict where the useful visual data will appear and can thus act to limit the data input into the system [324].

1.2 Visual Utility

In this thesis, we present a *visual utility* framework that is inspired by the biological vision system with its grasp reflex, but is applicable to many computer vision approaches. In this conceptual framework (fig. 1.3), the rich visual world is filtered by a visual utility process. This visual utility process uses simple features in order to reduce the amount of data processed by a higher level algorithm. This framework assumes that most of the visual data is not useful to the high-level algorithm. Thus, by removing most of the data in the visual utility process, the overall computational cost of the entire system can be reduced.

Visual utility is a metric for each location in an image of whether or not imagery from that location will be used by the higher level algorithm. It is inherently task specific and cannot be calculated a priori without a complete understanding of the algorithm as that would be seeing the future. However, we can examine algorithms that produce *estimates* of visual utility and compare that estimate after the fact with those locations actually used by the algorithm.

Saliency is a type of visual utility *estimator*. Saliency is the ability of an entity to stand out from its neighbours and thus attract the attention of the human eye [157]. It is usually generated as a *saliency map* where each pixel has a value proportional to its saliency. In its early computational formulation, saliency was defined as solely a bottom-up process computed from the raw imagery and independent of the task being performed [129]. More recent work has modified this definition to allow top-down task specific

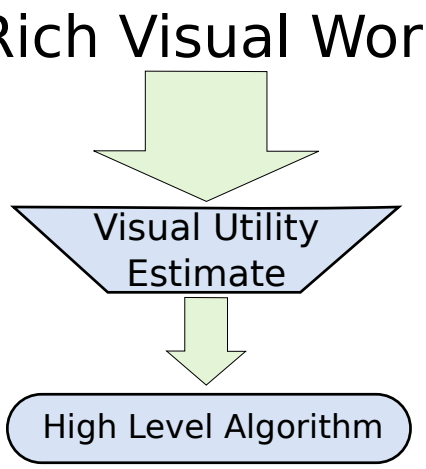


Figure 1.3: The generic visual utility framework. The rich visual world is filtered using a visual utility estimate before being processed by the complex, high-level task. In this thesis, we examine the computation time and task accuracy trade-offs of the entire system.

information to bias the saliency measure for finding a desired object [128, 211].

Unfortunately, the term *saliency* has been overloaded because it is difficult to compare different saliency algorithms objectively. Multiple evaluation techniques have been used including comparing with human eye tracking data [34, 95, 221, 230, 236], the ability to segment objects from the background [2, 121] and the ability to improve object recognition tasks [94]. Therefore, for clarity, we will only use the term *saliency* in its biologically-inspired context of where the human eye will travel in a scene. Furthermore, *bottom-up saliency* refers to where the eye will move in absence of a specific task, while *top-down saliency* refers to where the eye will move if it is looking for a specific object. Using this definition of saliency, we can see that saliency algorithms will always produce an estimate of visual utility. However, the accuracy of this estimate with vary depending on the task.

An estimate of visual utility can be used to build a filter on the visual data that must be processed by the higher level task. If the estimate is fast to compute, and/or the complex task is very expensive, then the resulting process could speed up. However, if useful data is incorrectly filtered, the high-level task accuracy could decrease.

1.3 Thesis Statement

Computer vision algorithms are computationally expensive and difficult to use efficiently in real world settings. Therefore, in this thesis, we explore the visual utility framework that is inspired by biological vision systems but appropriate for computer vision systems. In particular, we show that:

The visual utility framework can be used to reduce the computational cost of computer vision algorithms in a lossy way such that the degradation of task accuracy is minimized.

Stepping through this statement,

The visual utility framework

refers to the framework just described informally in the previous section and specified formally in chapter 3.

reduce the computational cost of computer vision algorithms

We will show that the visual utility framework reduces the computational cost of some computer vision

algorithms including the cost of computing the visual utility estimator. Our theoretical formulation in chapter 3 specifies under what scenarios the cost can decrease. Then we will examine in detail two theoretical case studies in chapter 4 where processing time, and thus computational cost, is reduced. Finally, we will perform some experimental case studies in chapter 7 that examine different visual utility estimators for object detection.

lossy way

Often, one goal of processes that reduce the cost of algorithms is to have an insignificant impact on task results. However, in many scenarios, it might not be possible to meet the computational requirements either in latency or in hardware a guaranteed approach. In this thesis, we are examining a class of approaches that have the potential to reduce the cost even more, albeit at risk of changing the high-level algorithm's results.

degradation of task accuracy is minimized

There are many ways to reduce the cost of a visual process if task accuracy is allowed to degrade. In an extreme case, one could just avoid executing the process. It will have a run time of zero, but the task accuracy will also be zero. We will show in this thesis that it is possible to minimize the amount of accuracy degradation for a given cost by utilizing the submodular structure of the problem discovered in chapter 3 to create the SCATAT algorithm (chapter 5), which generates a near optimal cascade of visual utility filters. We will then validate this algorithm in an experimental case study on object detection in chapter 7.

The rest of the document is structured so that in chapter 2, we explore the biological inspiration of visual utility in more detail and examine the many scenarios where a visual utility like approach has been used in the literature. chapter 3 defines visual utility formally and details the theoretical foundation of the thesis. chapter 4 presents theoretical case studies of two seminal cascade algorithms. In these studies, we analyze the algorithms in light of our theoretical findings on visual utility and show that these processes are near optimal. chapter 6 examines how visual utility and submodularity relate to deep neural networks. chapter 5 details the SCATAT algorithm, which generates a near-optimal cascade of visual utility filters that explicitly trades off computational cost and task accuracy. chapter 7 walks through our experimental case studies. Finally, chapter 8 summaries the conclusions of this thesis.

Chapter 2

Literature Review

> The best time to plant a tree was 20 years ago. The second best time is now.
>
> Chinese Proverb

This chapter discusses the prior work related to this thesis and presents an analysis of the effectiveness of prior visual utility implementations. Visual utility is inspired by an understanding of how the biological visual system functions. Therefore, in section 2.1 we will discuss in detail the models for the biological vision system along with the computational motivation for foveal vision (section 2.2) and some physical implementations of foveal cameras (section 2.3). Then, we will examine visual utility systems for computer vision problems. There have been a number of researchers that study visual utility problems in order to learn more about the brain. This can be broken down into two categories. First, work in social robotics tries to mimic human eye movement so that the robots react more naturally with the test subjects (section 2.4.1). Second, the work in saliency tries to predict the target of human eye saccades in different scenarios (section 2.4.2). Both of these streams show that it is possible to reasonably predict human eye motion for a generic task, but there is no analysis on the processing time or computational costs required because their goals are to better understand biological systems, not improve artificial ones.

Object detection is probably the most well studied application of visual utility because it is the most natural fit; a visual utility process attempts to find those regions that are of interest, while a object detector tries to find those regions that contain a given object. We can thus examine successful object detectors using the visual utility lens. The most effective detectors can be described as one or more visual utility estimators, trained to find the specific object. Cascades (section 2.8.1) are great examples.

Using a generic visual utility estimator is a different object detection approach that has been extensively proposed. The generic visual utility estimator can potentially be applied to any kind of object and is used to either limit the regions processed by a specific object detector or to segment out the objects of interest. In the literature, this kind of work has appeared under the names of *saliency* (section 2.4.2), and *objectness* (section 2.5.2). A full analysis of these approaches is given in section 2.7, but we find that there is evidence these approaches filter the candidate regions better than random. This is crucial for the object discovery task where the system learn new objects and be able to find them again reliably. However, there has been no analysis that explicitly looks at the computational cost of the entire object detection system and takes into account the computational cost of the visual utility estimator.

A slightly different form of visual utility can be found for active and reactive vision tasks. In active vision tasks, the visual utility estimator uses the current image, along with a model of the world (map) in order to plan subsequent measurements. This is the next-best-view problem and is detailed in section 2.6.1. A significant feature of these problems is that the processing cost of the high-level algorithm is expensive because an agent must move around the scene in order to get the next measurement. This enables more

complex processing in the visual utility estimator. Reactive visions systems, discussed in section 2.6.2 are those that react to changes in the environment to direct a sensor to gather new information.

Finally, at the end of this chapter, we will outline the submodularity. In chapter 3 of this thesis, we identify that building a cascade of visual utility estimators is a submodular process. We then use that insight in order to develop the SCATAT algorithm in chapter 5.

2.1 Eye Movements and How They Filter Data

The neuroscience and cognitive psychology communities have tried to understand how the mechanisms in animal visual systems function that define eye movements. These studies are interested in examining and reproducing the animal's behaviour and many of them are conducted in laboratory settings [54]. Therefore, their results are not necessarily useful for solving complex tasks in the unstructured environments that we are interested in. However, humans also solve complex visual tasks and our visual system has been fine tuned over millions of year. Therefore, this body of work provides an excellent source of models for the structure presented in this thesis for their ability to help solve the tasks we are interested in.

Early observations noticed that the eyes do not scan a scene evenly. The eye will fixate on an area for a while and then jump to another location using a ballistic motion called a saccade [136]. The eye will even fixate when it is traveling relative to the scene in order to keep the focus of attention stable. As a result, measured eye movements show a smooth path when tracking interrupted by fast saccades [169].

This fixate and saccade behaviour is common throughout the animal kingdom. Land presents a comparative study of eye motions for different animals and notices that except for a few insects and gastropods, animals have evolved this behaviour. He shows that the main reason for keeping the eye still while fixating is to minimize motion blur that occurs in the photoreceptors. In this view, the saccades serve to increase coverage of the scene and they are performed quickly in order to minimize the visual information that will be discarded as being too blurry [169].

Yeshurun and Carrasco demonstrate that fixating also increases the effective resolution of the eye. They prove this result by running an experiment where subjects must segment artificial textures. Performance on this task will peak at a specific spatial resolution. In the experiment, they ask subjects to gaze at a particular spot and then flash the artificial textures for different amounts of time. In the area seen by the fovea, observing the scene for a longer period of time decreased performance, while in the areas seen by the periphery, performance increases. This paradoxical result shows that eye fixations serve to increase the perceived resolution [326].

Researchers have also worked to determine what factors affect the target of a saccade. Eye tracking experiments where subjects are not pre-cued show that the eye will move towards areas that are outliers in some low-level cue like orientation, color, contrast and motion [193, 215, 221]. However, in experiments where the subject is given a task, we see examples where some higher level mechanism in the brain is influencing the eye motions. For example, Wolfe et al. use a visual search task where subjects must find simple objects like red horizontal lines in a group of distractors like green vertical lines. Subjects were able to find the red lines more quickly if they were first told to find the red lines as opposed to being uncued [315].

Task-based biasing of the saccade target seems to be stronger as the task becomes more complex. A set of experiments track the eyes of subjects while they were performing everyday tasks that require hand-eye coordination like making a sandwich, pouring tea or playing cricket [170, 171, 224]. In the sandwich and tea experiments, subjects would often look at an object several seconds before it was needed and then look somewhere else (fig. 2.1). When that object was subsequently used, the eye and the hand would move towards that object simultaneously. This observation implies that there must exist some short term spatial representation of the scene that helps to drive both hand and eye movements [114]. In concert with this short term memory, there is evidence from the cricket experiment to show that the human eye will learn

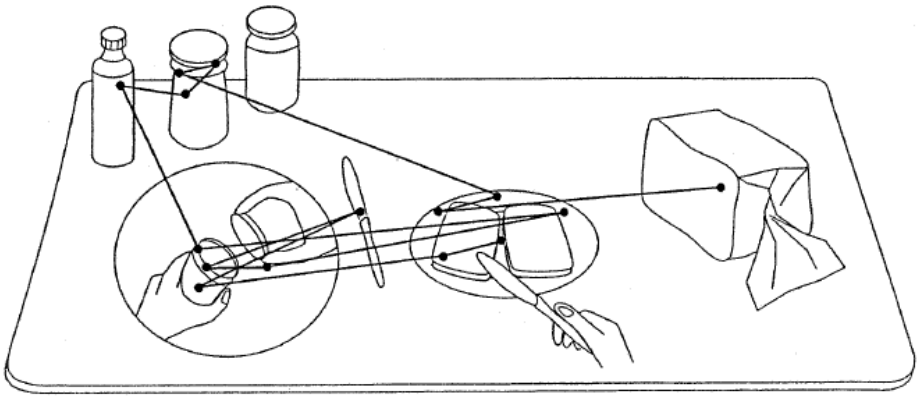


Figure 2.1: Typical human fixation pattern measured when a person is making a sandwich. Dots represent where the gaze will stop momentarily before executing another saccade. Figure from [170]

where to look over longer periods of time. In this experiment, cricket players were observed to look not at the ball, but at the ground where the ball was expected to land. The ground is not visually interesting at the time of the saccade, but it is crucial for the player to know the timing of the ball hitting the ground in order to swing the bat and thus the eye fixates on that location [171].

The human visual system also exhibits the phenomenon called the inhibition of return. When the eye executes a saccade away from a location, it is less likely to return to that location on a subsequent saccade [229]. Klein has shown that the inhibition of return is a necessary mechanism for the kind of serial visual search that humans perform. In a serial search, the eye first looks at one location to search for the desired object. If it is is not found, then the eye executes a saccade and looks at a new location. The process continues until the target is found. Without an inhibition of return, the eye would become stuck bouncing between a set of targets and never find the object. In a parallel search, where the entire scene is processed simultaneously, an inhibition of return is not necessary [154].

Though we do perform serial searches, there is also evidence of parallel processing of low level cues. Desimone and Duncan present a meta-analysis of multiple psychological studies and conclude that visual attention is an emergent behaviour of many different neural mechanisms that compete for finite visual processing. In their analysis, they found evidence that low level cues are extracted from the scene and grouped into objects in parallel, but evaluating those objects for a task usually occurs serially. In this view, the order of evaluation is driven by the amount of neural processing required for each object and thus, visual attention emerges from this minimization [69].

2.2 Computational Motivation for Foveal Vision

Many researchers that have used a foveal vision paradigm motivate their use of the approach as a way to lower the amount of processing required. There is both biologic and computational complexity evidence for this view. Biologically, we see evidence of the processing in the human visual pathway. Each area in the retina can be mapped directly to a specific portion of visual cortex where it is processed. Though the fovea only covers 1% of the retina, it maps to over 50% of the visual cortex [147]. Therefore, our brain would have to be significantly larger if our retina were covered with photoreceptors at the same density as it is in the fovea.

Two researchers have also performed computational complexity analyzes for foveal vision systems. Bandera, in his thesis, presents a first order analysis where he uses a complexity measure of the amount of data captured multiplied by the computation time. Using this metric, a Cartesian array of NxN pixels

has an $O(n^4)$ complexity while an exponential foveal array has a complexity of $O((logN)^3)$ [19].

Tsotsos has also studied computational complexity of visual search, which is one of the most common visual tasks. In his analysis, he differentiates between bounded search where the desired object is completely known, like a template search, and unbounded search where the designed object is unknown. He proves that unbounded search, which is more common in practice, is NP-complete while bounded search is linear in the number of pixels. Even though unbounded search is NP-complete, Tsotsos identifies five optimizations that reduce the computational complexity of visual search to a small exponent making solving it more practical. The first optimization is a high degree of parallelism. The second is a hierarchical abstraction of visual prototypes that can cut search time logarithmically. The third is a hierarchical organization of raw input tokens to maintain semantic content and minimize the number of receptors. The fourth optimization is to take advantage of a physical world that is spatio-temporally localized and thus organize receptors locally. The last optimization is to be able to pool responses from different stimuli into a single interest map [285].

In a follow-up study, Tsotsos compares the computational complexity of active visual search compared to passive visual search. In this case, active search uses a hypothesize and test approach to control where to acquire more information but takes time to do so, while passive search waits for all the information and then searches through it. In this analysis, active search will always be more efficient for unbounded search that uses a brute force approach. However, if the unbounded search is performed using the 5 optimizations previously mentioned, then active search is only more efficient if the hypothesis pruning is good [286]. Applying this analysis to our framework, we can treat our concept of visual utility as a method of fast hypothesis pruning. Therefore, with an accurate enough visual utility estimate, we can except a computational performance increase for the overall task.

Empirically, we can also see that using large data bandwidth from high resolution imagery will cause complex algorithms to perform too slowly. For example, Hubner and Pajorola ran a timing experiment for calculating stable features across frames, a technique used in many high-level vision tasks. Using a 2.0 Ghz Core 2 Duo machine, SURF features [23] were calculated in 50.1 fps for a 256x192 image, but only 4.8 fps on a 1024x768 image [124]. Unfortunately, the ever-increasing processing power in computers cannot handle high resolution imagery because camera detectors and microprocessors are fabricated using the same technology: semi-conductor photo-lithography [42]. Thus, the same advances that decrease the size of a transistor also decrease the size of a pixel. So, while processing power increases, so too does the data captured.

2.3 Physical Implementations of Foveal Cameras

In order to achieve the computational savings of foveal vision, foveal cameras have been implemented in hardware and using optics. Most of these implementations use a log-polar mapping which was developed to mimic primate eyes [325]. The log-polar mapping defines a pixel to be an arc slice between two radii while the spacing of radii distances follows a logarithmic relationship (fig. 2.2). The log-polar mapping creates high resolution pixels at the center of the frame and low resolution pixels on the periphery. This representation also has the advantage that it is rotationally and scale invariant. However, in contrast to traditional square arrays, this representation is not location-invariant [137].

Rojer and Schwartz use the continuous log-polar transformation in a design trade-off model for if the transformation were to be implemented on chip as a visual sensor. This model allows the designer to select the desired field of view, the number of pixels in the outermost periphery, and the parameter a which displaces the singularity at the center of the field of view. Using this model, they show how to duplicate the field of view and resolution found in the human visual system [238].

Sandini and Tagliasco created another model that was simpler to implement in chip fabrication. Their model two distinct regions: the fovea and the periphery. The fovea is a high-resolution, uniformly sampled rectangle in the center of the field of view, while the periphery uses the log-polar mapping [246]. Van der

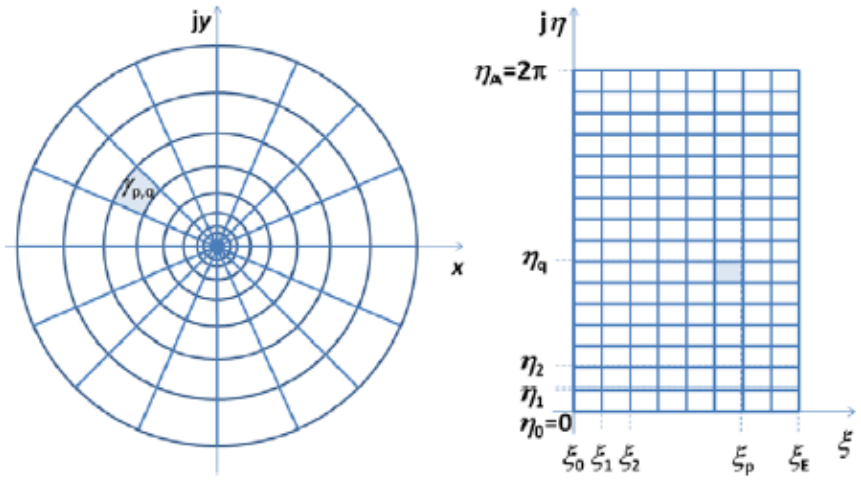


Figure 2.2: The-log polar mapping converts a point in the image plane (defined as $z = x + jy$) to log-polar coordinates (defined as $w = \xi + j\eta$) using the relationship $w = log(z)$. In practice, to avoid the singularity at the center of the field of view, a $w = log(z + a)$ mapping is used. Figure from [137]

Spiegel et al. first implemented this model as a 2000 pixel CCD using VLSI technology [294]. Wodnicki et al. used the same two-region approach to create a CMOS sensor [313]. Simultaneously, Sandini's group created a 8000 pixel CMOS implementation that took advantage of newer fabrication methods to remove the discontinuity between the fovea and the periphery [86]. Finally Sandini et al. created a 33,000 pixel color CMOS sensor [247]. A similar CMOS implementation has also been created as a retinal implant for people suffering from photoreceptor degeneration [252]. For a more detailed review of these fabrication efforts, see [28].

As opposed to creating arc-shaped pixels, Bandera uses a different approach. In his thesis, he creates sensors using traditional square pixels, but whose size grow as the move away from the periphery. He analyzes growing the pixel size both linearly and exponentially and the impact of that decision on the data bandwidth generated [19].

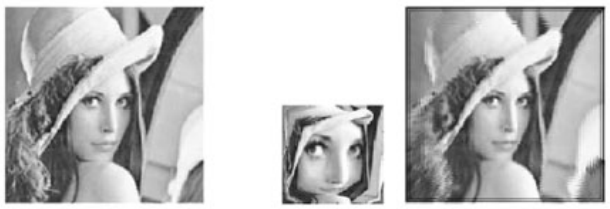
Instead of creating new sensors, Kuniyoshi et al. designed a new lens that when projected on a Cartesian sensor, simulates a log-polar mapping [163]. They then used two of the lenses in a robotic head and demonstrated stereo tracking [164].

The log-polar mapping has also been simulated using data from traditional Cartesian sensors that has been processed either in software [68, 89] or using specialized hardware [251]. Due to the extra computation to do the transform, these approaches are not valuable for real-time systems, however, they are useful to run proof-of-concept experiments and to develop vision algorithms in this space [137].

Though a log-polar mapping has some advantageous properties including biological plausibility and rotational invariance, algorithms designed for Cartesian sensors will not work directly on a log-polar image because pixel adjacency no longer has the same meaning [325]. Therefore, we cannot take advantage of most the tools developed in computer vision from the last 30 years.

Some algorithms have been reformulated for use on a log-polar image and take advantage of the new structure to execute more efficiently. For example, optical flow can be computed quickly, especially when trying to predict the time to impact of an object approaching the camera along its boresight [61, 220, 280, 325] and object tracking is very accurate because of the high resolution in the center of the view [180, 251]. Other algorithms are not necessarily more efficient in log-polar space, but have been demonstrated. Such algorithms include: image registration [235, 314], stereo scene reconstruction [27], object matching [43], interest operator calculation [319] and saliency calculation [273].

(a) Reciprocal-wedge transform [281]



(b) Cartesian foveal geometry [196]

(c) Multifoveal geometry with fovea placed on the outer faces. [21]

Figure 2.3: Various proposed multifoveal geometries.

Though not applicable to the general log-polar model, Bandera's approach of using square pixels sized using a log-polar model allows the use of some Cartesian vision algorithms. By using square pixels, Bandera picture is equivalent to sampling from an image pyramid along a 3D manifold (fig. 2.4) [19]. Thus, Cartesian algorithms that operate on image pyramids can be easily transformed to operate on this kind of foveal sensor. For example, Young demonstrated a silhouette classification algorithm that operates on Bandera's data structure [327] while Arrebola demonstrated simple segmentation and object detection [12].

Other foveal transforms have also been proposed for different tasks. Tong introduced a reciprocal-wedge transform that is effective for road following and recovering depth in motion stereo (fig. 2.3(a)) [281]. Martinez implemented a Cartesian Foveal Geometry using FPGAs and demonstrated faster object tracking (fig. 2.3(b)) [196]. Finally, multi-foveal Cartesian systems have been simulated in software for image and video compression where there are multiple regions of interest in the scene (fig. 2.3(c)) [21, 41].

As opposed to using different mappings of a single plane to create foveal vision, some researchers have used two coordinated Cartesian cameras with different fields of view. One camera has a small field of view and high resolution like the fovea, while the other has a large field of view and low resolution like the periphery. This approach means that all of the traditional computer vision algorithms can be used, while still reducing the total data bandwidth that needs to be processed. We are using the same approach and evaluating its impact on overall task performance when coupled with visual utility based control.

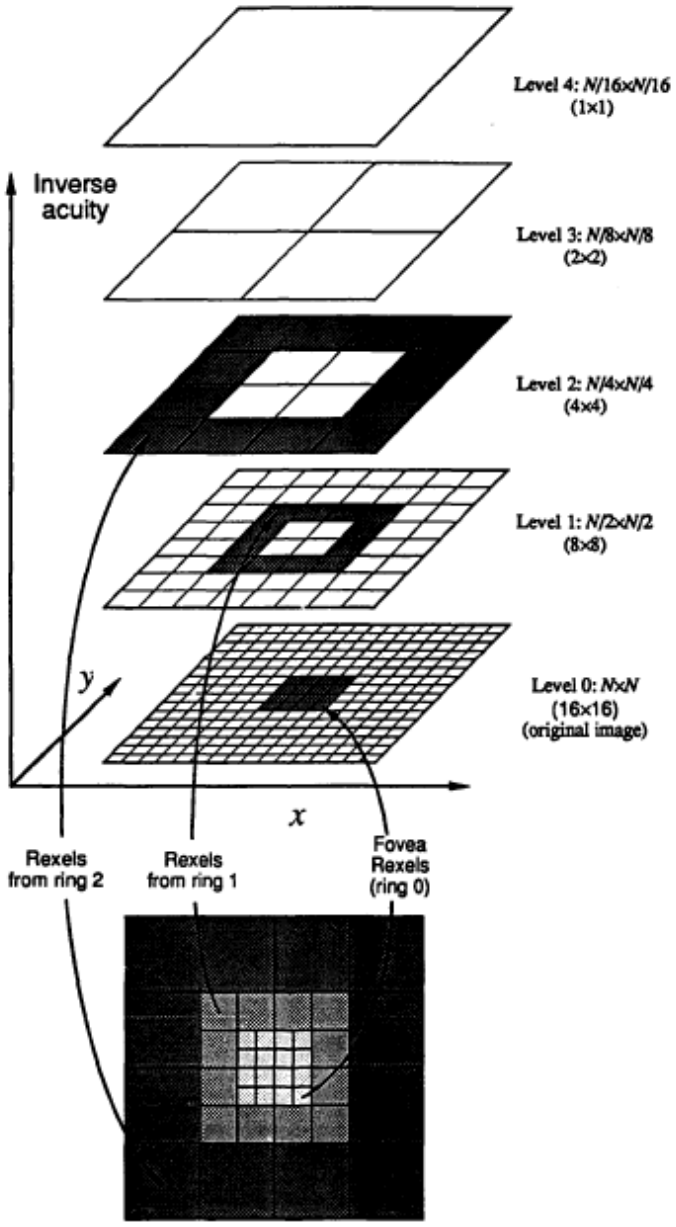


Figure 2.4: Bandera's method of mapping a foveal sensor is equivalent to sampling a manifold from an image pyramid. Figure from [19].

Ude describes how to calibrate these two cameras along with a control scheme that allows the high resolution camera to focus on an object found in the low resolution camera [288]. This calibration is useful for object classification and tracking tasks. In these tasks, the low resolution image is used to identify the location of the objects and the high resolution feed is used for the classification. This has been demonstrated in three different settings. First, this approach was used on a social robot to track human eyes. The low resolution feed searched for faces in the scene, while the high resolution feed focused on the eyes [248]. Second, Gould used this approach in an office setting to identify objects like cups and staplers and then minimize their position uncertainty over time [103]. Finally, both Merger and Ude use this approach in a laboratory setting to find and classify objects like stuffed animals [199, 287].

Achieving speedups by processing imagery at multiple resolutions has also been demonstrated on a single sensor. Klarquist simulates a moving fovea to create a multi-resolution 3D map of the environment using stereo cameras. The fovea is controlled using an active strategy preferentially processes higher resolution data where there is the largest uncertainty in the 3D reconstruction [153]. In a more general early study, Burt analyzes processing an image pyramid and demonstrates value using industrial inspection, surveillance and object recognition [37].

2.4 Visual Utility Models of the Brain

Much of the work in computer visual systems that implement a visual utility process are attempting to model the neural processes in the brain. These researchers model these processes in order to better understand the brain. In particular this work is prevalent in social robotics and saliency.

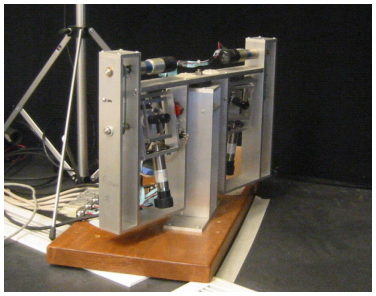
2.4.1 Social Robotics

Visual utility processes have been implemented in social robots. Sometimes, these implementations are used to simulate a theory of the human visual system. By comparing the robotic response with known human responses, the neural model is tested. The Harvard Head (fig. 2.5(a)) was one of the first robots for these experiments. This head consisted of two cameras and a fast control system to allow the head to simulate saccadic motion [87]. It was used to simulate a few saliency models that incorporated a winner-take-all strategy [52, 53]. Another head, developed at the University of Toronto around the same time (fig. 2.5(b))[201] has been used to develop and test a neural model for visual search that uses pyramids of simple features and top-down biasing [329].

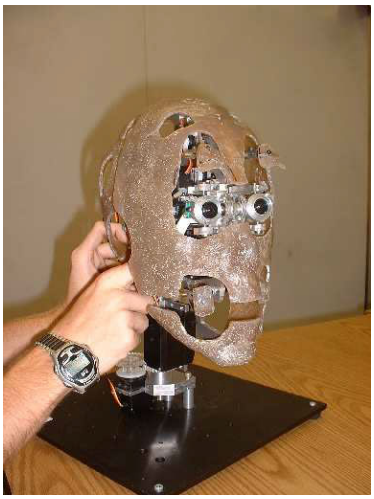
Another class of social robots are developed to simulate human learning. For instance, Lopes et al. use a robotic torso with one arm (fig. 2.5(g))[188]. They demonstrate a framework for imitation learning where the eye and hand movements are learning from people holding objects up to it. They robot learns how to look for an object and how to track it when interacting with the object using its arm [187]. Another robotic torso, called Babybot that has one arm and log-polar foveal eyes (fig. 2.5(h))[200]. In an experiment Natale et al. enable Babybot to learn how its actions affects what it sees. The robot executes pushing and pulling actions on objects in front of it and then learns to predict where the objects will move to and where to move the eyes in order to most accurately move the objects [210]. Orabona et al. demonstrate a humanoid robot that learns an metric for the likelihood that an area in its view is an object. The robot explores its workspace and tries to grab objects. When an object is successfully grabbed, it is identified as an object and incorporated into the visual model. The eyes of the robot are then directed towards areas more likely to contain objects [217]. Finally, Kim et al. enable a cheap robotic head to track objects using cues that change over time (fig. 2.5(c))[152].

Neural models in social robots is also used to allow the robot to interact in a more natural manner with the people around it. For instance, Scassellati uses a two camera foveated system where the wide field of view camera finds a person's head and then the robot's eyes, which have a narrow field of view, are directed towards the human's eyes [248]. Jasso uses a reinforcement learning method to allow a robot to

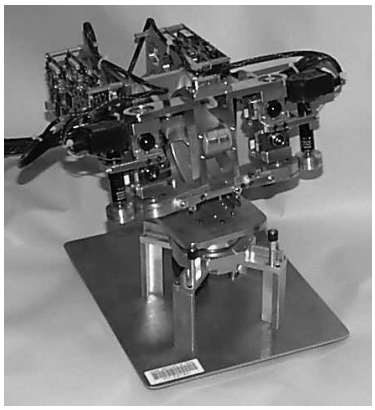
(a) [87]



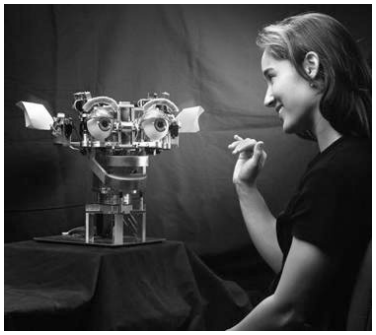
(b) [329]



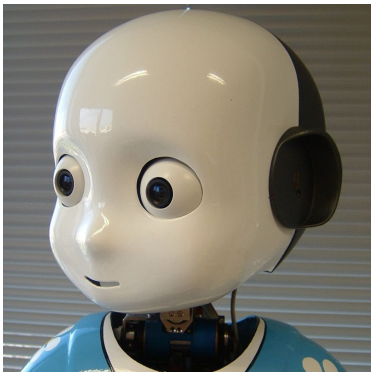
(c) [152]



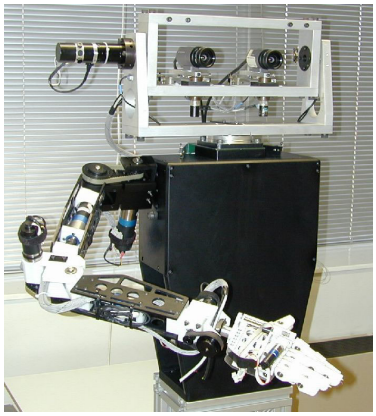
(d) [248]



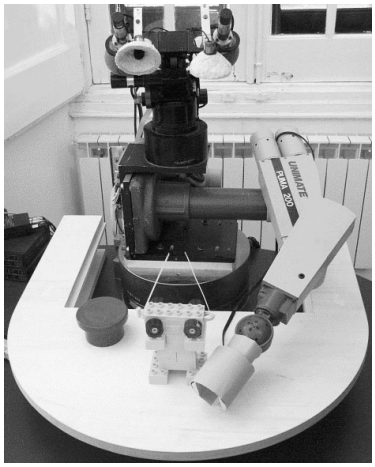
(e) [32]



(f) [242]



(g) [188]



(h) [210]

Figure 2.5: A selection of social robots.

infer where people are looking and thus be able to react naturally when a person directs the robot to look at something [135]. Breazeal and Scassellati use the a saliency model from [315] to simulate human-like visual attention behaviour on the robot Kismet (fig. 2.5(e))[32]. Butko et al. enable a robot to have natural eye movements using a saliency metric. They speed up the calculation of the saliency from (Zhang et al. 2008) and then use the saliency map to direct camera saccades [38]. Finally, Ruesch et al. fuse both audio and visual saliency in order to direct the head and eye movements (fig. 2.5(f))[242].

2.4.2 Saliency

Many different computational saliency models have been proposed to try and recreate the observed behaviours of the eye and understand the potential neural mechanisms behind them. A recent review of this field is presented in [30]. Many of these models stem from Koch and Ullman's seminal work in 1985 where they introduced the concept of a saliency map. In their formulation, a saliency map is a map across the field of view built by combining multiple feature maps. These feature maps are low-level features like color, motion and orientation where areas are singled out that differ significantly from their local surroundings. High values in the saliency map indicate regions to be targeted by saccades [157].

One of the most influential evolutions of Koch and Ullman's saliency model was presented by Itti et al. In this seminal paper, Itti et al. apply center-surround operators to the raw feature maps at different scales before they are combined into the saliency map (fig. 2.6). This approach simulates the neural response found in the human visual system where a stimulus in the center of the field will trigger a response, while a stimulus in the immediately surrounding region will inhibit that response. Therefore, this configuration will trigger when the stimulus in the center is significantly different than that in the surround. Unsurprisingly, this model can be implemented easily using convolutional neural networks. In addition to including center-surround operators, Itti et al. implemented their model as a computer vision algorithm to be run on still images and analyzed the results qualitatively [129].

Though evaluating the results of a saliency model qualitatively can provide insight, a more quantitative approach is needed to be able to compare different models. The de facto standard for those who wish to understand the human brain has been to compare the results with eye-tracking data from human subjects. The comparison can be made in two ways. First, a list of fixations can be generated from the saliency map by ordering peaks in the saliency function. This list is then compared to the fixations measured by the eye tracker [230]. Alternatively, the eye tracking data can be used to create a heat map of where the eye traveled by treating each eye fixation movement as a gaussian centered at the point of fixation. These gaussians are then combined spatially to create a map which is compared directly with the saliency map [34, 219]. Comparing different approaches also relies on the data used. Most researchers have used their own datasets, however, recently, some standard datasets of natural pictures have been made available [142, 176, 183, 234, 293].

In addition to the center-surround model, Itti et al. also demonstrated a simplistic inhibition of return mechanism. This mechanism would zero the saliency map in a circle of fixed width around the points that have been seen before, thus causing the attention to move to another part of the picture [129]. Parkhurst et al. extended this model in two ways. First, he smoothed the inhibiting signal to avoid the harsh discontinuity in the saliency map at the edge of the circle. Second, he multiplied the saliency map by a gaussian centered at the current point of fixation. This simulates the belief that points further in the periphery are less likely to be noticed by the human visual system [221].

Perrerira Da Silva et al. use a different mechanism to create an inhibition of return. They simulate a prey/predator ecosystem where the "prey" breed in proportion to the static saliency map and then the "predators" move towards those areas of higher "prey" concentration. Then, at any given time, the focus of attention becomes the location of maximal "prey" population. By fiddling with the parameters of this model, it is possible to generate a range of behaviour from one that explores significantly to one that fixates on a specific location [226].

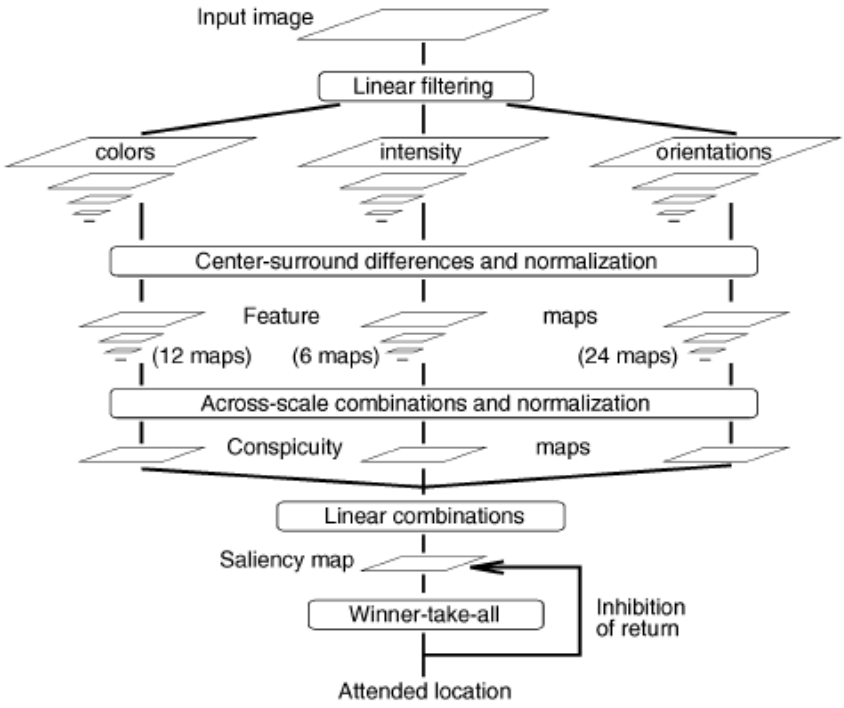


Figure 2.6: Model of saliency from [129].

One of the problems with Itti's initial saliency model is that the resulting saliency maps could be quite noisy. Noticing this, Harel et al. proposed a Markov-based smoothing method that operates on a graph sitting on top of the saliency map. In this method, they treat the graph like a Markov chain and allow saliency to collect in local areas. The resulting map is smoother and has more distinct points of attention, which can predict the human gaze more accurately [111].

Though Itti's method only used color, orientation and intensity as low level features, some researchers have predicted that the human mind is naturally hardwired to see other shapes like faces [47] and text [179]. In the face experiment, the response from a face detector [299] was added as another feature map in Itti's framework. Interestingly, the resulting saliency estimator showed improved correlation with human eye tracking even on images without faces because spurious face detections still triggered the human eye [47].

While Itti implemented a simple version of Koch and Ullman's theory, Le Meur et al. propose a more complicated saliency metric that models more of the second order psycho visual phenomenon found in the human visual system. For example, instead of a simple contrast metric, the non-linear human contrast sensitivity function is used. The resulting predicted attention points are then compared with those found from a human experiment [173].

Thus far, all the saliency models we've discussed are purely bottom-up approaches that try to predict the motion of the human eye when it views a scene but has no task. However, Yarbus demonstrated in 1967 that the task has a significant effect on the motion of the eye [324]. Navalpakkam and Itti have developed two different methods of influencing a saliency map with top-down information. First, for visual search tasks, the different feature maps in Itti's saliency model can be weighted before they are combined. These weightings are set based the feature responses of representative pictures of the objects being searched for. For instance, say we are looking for a blue sign, then the feature responses would tend to be strong in the blue channel and thus the blue feature map would have a higher weighting [211]. The second top-down

method uses retrospective feedback information about what part of the image is useful for the task at hand. In their experiment, a human identifies the relevance of the current focus of attention and then this information is used to create a Task Relevance Map which is multiplied by the saliency map [212].

Another top-down approach is presented by Torralba et al. In this work, a bottom-up saliency response is combined with the expected location of a particular class of object based on its context. For example, a person is more likely to be found walking on a road that in a tree. The authors create a model of context for each object class using a large dataset of annotated images. In each case, a distribution of the object's location in the frame is determined based on large scale features that encode the large structures of the scene [205, 282].

In addition to predicting human eye movement on a static picture, some studies have tried to replicate the properties of human attention when looking at a video sequence. Itti and Baldi propose a model that extends Itti's 1998 method by including motion features and Bayesian suprise [126]. In a more recent approach, Rudoy et al. explicitly model the salency map in a given frame to be conditional on the one before it [241]. Finally, Rudoy et al. generate saliency maps for a given frame conditioned on the map from the previous frame. This smooths the saliency map over time and makes it more appropriate for video sequences [241].

Other studies have explored video sequences by implementing a neural model on a robotic head. For instance, Clark used a robotic head with a single camera to test a model where saccade generation and the gap effect would emerge implicitly. This head calculated a saliency map in real-time as a linear combination of low level features. Each point in this saliency map was then modified by a winner-take-all model with inhibition. The camera was then controlled to always move towards the location of the maximum modified saliency. When run, this head will automatically saccade towards a peripheral stimulus, get bored, and then saccade somewhere else [52].

Another robotic head by Colombo et al. simulated a more complex model and was developed as a demonstration of combining a foveal sensor with an attention model in a real system. This head was created with a log-polar foveal sensor, an attention model and short term memory. The attention model included both bottom-up and top-down components to generate a saliency map. Furthermore, observations were combined over time by mapping them in an ego-centric space based on the pitch and yaw sensors of the head [55].

All of the approaches to saliency estimation described thus far have tried to explicitly model the neural structures in the human brain. Other studies have tried to predict human eye motion by modeling the functional ability of the brain while ignoring how it is physically implemented. These studies model the human eye as a sensor that tries to maximize the usefulness of the imagery captured in an information theoretic sense. Renninger et al. restrict their study to silhouettes and predict saliency based on minimizing the uncertainty of the silhouette's shape [236]. Bruce and Tsotsos propose a bottom-up saliency estimator by maximizing Shannon's self-information of visual features [34]. Zhang et al. combine the bottom-up self-information approach with top-down mutual-information from the visual features biased according to scene statistics from a large set of natural imagery [331]. Butko et al. propose a speed-up to Zhang's model by approximating the filters for extracting features as square [38]. Finally, Tamayo and Traver describe a technique to minimize entropy on log-polar images [273].

There have also been a few data-driven approaches presented that try to predict saliency. Liu et al. calculate a set of feature maps and combine them by learning a conditional random field on object locations [183]. Instead of using extracted features, Kienzle et al. learn a mapping from the raw data in 13x13 pixel image patches to the likelihood of the eye landing on that location [151]. Finally, the most recent approach by Judd et al. combines Itti's features with a horizon detector, a face detector, a person detector and GIST [216] features to train an SVM to predict if each location will be a target for an eye fixation [142].

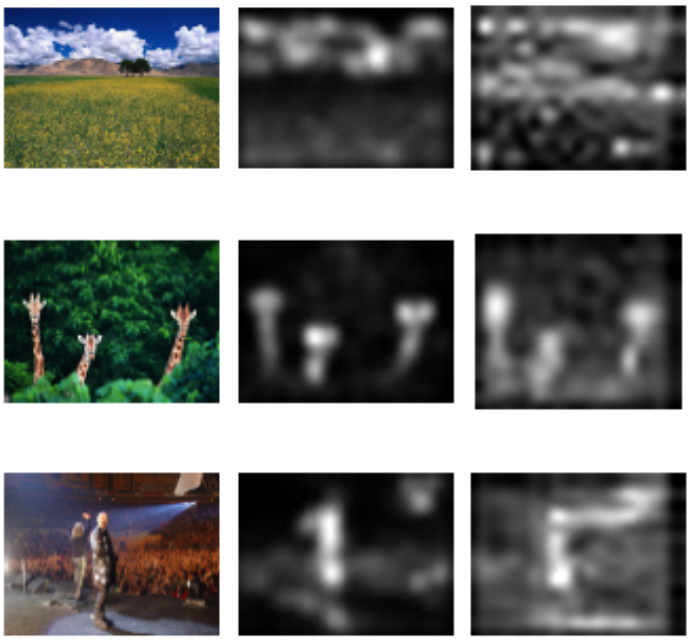


Figure 2.7: Examples of visual utility maps using [121] (center) and [129] (right). Figure from [121]

2.5 Generic Visual Utility Estimators

A generic visual utility estimator is one that can be applied to many different tasks. Saliency is the most obvious example because the human visual system is very adaptable and functions amazingly well for a large range of tasks. However, other models have been developed that are general purpose.

2.5.1 Estimators Called Saliency but Really Aren't

Saliency, defined by being able to predict human eye motions, is a general purpose estimate of visual utility but the literature also includes work where the authors use the term saliency when creating a map of visual utility for their task. In these cases, the authors use saliency because their estimators are general enough that they could be useful to tasks other than the ones they test. However, the estimators are being used for specific tasks like object segmentation (e.g. [2]), background subtraction (e.g. [307]) and tracking (e.g. [144, 145]).

One of the problems with the biological models of saliency is that they can be a costly to compute in a sequential processor. The computational cost is not an issue when trying to understand how the human visual system works, but to use a visual utility map for some application, speed matters. Frintrop et al. try to address this problem using square filters and integral images [92], but Hou and Zhang take a different approach. They introduce a spectral residual approach that is fast to compute and can work well for roughly locating objects different from their background. Their approach identifies anomalies in the frequency domain of the image compared to an average natural image. These anomalies will correspond to objects of interest [121].

Avraham and Lindenbaum use a statistical model to find the areas of interest in a bottom-up way. They use a stochastic model to compute the probability that an area of the image will be of interest assuming that there are a small number of areas of interest in the scene and they are visually distinct

from everything else. Using this assumption, they build a Bayesian tree on image segments and infer correlations between nodes based on the mutual information between the visual feature vectors. Finally, the visual utility is calculated by marginalizing over the N most likely interpretations of the Baysian tree [13].

Valenti et al. base their visual utility estimate on the structure of Isophotes. Isophotes are lines connecting points of equal intensity and are like contour lines on a map. They propose that each isophote is part of an interesting structure and the location of that structure is roughly in the uphill direction from each point on the line. The visual utility map is then created by accumulating all the estimates for the locations of interesting structures and then modulated using the curvature and color distinctiveness of the region [290].

For some problems, like scene summarization, the context around the area of interest is also useful. For these types of problems, Goferman et al. propose a context-aware visual utility technique. This technique reweighs the underlying visual utility metric of the areas around a region of interest by their distance to a location of focus. The modified visual utility map is then segmented to find the final region of interest. This allows a region of interest to grow into those areas around of point of interest that also have reasonably high visual utility. As a result, some context around the point of interest is maintained [98].

Von Ahn et al. use a massive amount of human labeled data to predict the relevant areas for recognizing objects. They collect this data using a web game called Peekaboom where two players are partnered randomly. One partner can see the entire image of an object and tries to get the other player to guess what the object is by slowly revealing portions of the image. A higher score is given if less of the image is revealed. As many players play the game, a large dataset of images with relevant regions is created which can then be used to develop a visual utility metric [300].

Wang et al. also use a large dataset to calculate visual utility. In their case, it is a Google scale image set of random, unlabeled images. To calculate visual utility for a given image, this database is searched for the n nearest neighbors in appearance to the query image. Each of these images is then warped to line up with the query image. Finally, the visual utility is defined as the mean distance to all the neighbours. This approach is akin to anomaly detection across a global knowledge base [305]. An almost identical approach is presented in [195] for patches relative to the image as opposed to the image relative to a corpus.

Hoiem et al. present a different learned approach and apply it to object localization. They assume that objects of interest in the scene can be roughly found by understanding the 3D structure of the scene. To understand the scene, they predict the 3D structure based on a single image using a classifier trained using a large dataset. This classifier labels regions in the scene as ground, sky, or vertical. They then assume that objects will appear on the ground plane of the scene and thus, the visual utility is the probability that a vertical area is supported by the ground. Then, since the scene is undergoing a projection and the rough scene geometry is known, it is possible to estimate the real-world size of various objects. Thus, they use the top-down information about the size of the object being localized in order to prune potential matches [120].

Visual utility maps have also been extended to three dimensions. Schauerte et al. present a framework that uses multiple cameras and microphones to aggregate visual utility estimates in a 3D space. To do this, they use ray casting to back project a saliency value from each camera and gather evidence in each voxel. They then use this model to optimize camera views in a smart environment [249].

Statistical models have also been developed for calculating visual utility in video sequences. Rosenholtz used a simple model for a visual search task where he defined the visual utility as the motion of objects once the average motion of the frame is subtracted [239]. Zhai and Shah also identify regions that move differently than the scene as being of interest except that they find these regions using a RANSAC-like algorithm that clusters SIFT pairs based on their consistency to different homographic transforms [330]. Baluja and Pomerleau train a dynamic neural network to predict the next frame to arrive and then the visual utility becomes the difference between that predicted frame and the actual one [18]. Boiman and Irani use a model that looks for anomalies in the configuration of distinctive spatio-temporal features [29].

Finally, Gao et al. use a model that classifies a region into a stimulus of interest or not in video sequences. Their model calculates the mutual information between features in the center and surround regions of spatio-temporal center-surround structures. The less mutual information present, the higher the visual utility [95].

2.5.2 Objectness

A common task for applying visual utility is to segment objects from the background. In this task, an object is identified by first computing a visual utility measure related to the likelihood that a region in the image belongs to an object. This type of visual utility measure was coined as *objectness* in [4], but has also been labeled imprecisely as *saliency*. Once an objectness measure is calculated, it is used as input into a segmentation or region growing algorithm in order to extract objects from the background.

In one of the early cases for objectness, Rutishauser et al. evaluate whether the bottom-up saliency metric from [128] can be used to find objects. They extract objects by first identifying locations of interest from the saliency map. Then, they grow the region around each point using a region-growing algorithm on the feature map that contributed the most to the high saliency score. Finally, objects are classified using the bag-of-words model on SIFT features from [190]. By keeping track of objects already seen in a database, this approach results in an object discovery technique [245]. Walther and Koch have a very similar approach. Once again, the saliency metric from [128] is used, but then objects are extracted use a biologically plausible neural network to grow the region [302]. Han et al. use use the same saliency measure, while the region growing is done using an MRF model seeded by the objectness score [108]. Ko and Nam take a different approach for the final segmentation. They first over segment the original image and then use the saliency measure from [129] in order to merge together segments that belong to an object [156]. Similarly, Yan et al. use a hierarchical segmentation strategy to more accurately segment objects when the background contains high frequency structures [322].

Looking at the problem from the other direction, Jiang et al. use a measure of background, called backgroundness, that they use to hierarchically carve out the potential objects in the image [139]. Instead of using pure CRFs or MRFs, Yang et al. use manifold learning in order to segment the object. This technique shows a significant improvement in the high dimensional image feature space because distances throughout the space are not consistent [323]. The segmentation can also be done greedily because the problem can be formulated as a coverage problem, which is known to be a submodular maximization problem [140]

Coarse boundaries for objects are found by thresholding a rough estimate of objectness. This is done by Hou and Zhang in [121] by thresholding their spectral saliency approach. Wang and Li then refined the boundaries of the objects by applying a graph propagation step based on the visual similarity and proximity of the pixels. This refinement enables tight object segmentation in some cases [307].

Other techniques define an objectness measure that inherently produces tight boundaries around objects and thus requires a less sophisticated segmentation step. Achanta et al. compute objectness using a combination of band pass filters to create a full resolution visual utility map that can be used extract the objects of interest by adaptive thresholding [3]. Rahtu et al. evaluate the contrast in a large set of center-surround regions at different scales all across the image. For each center-surround region, the contrast is calculated as the chi squared distance between the histograms of the pixels in the center region and those in the surround region. The base features for the histograms are the illumination, color and optical flow. Then, the objectness of each pixel is simply the maximum contrast found in any window whose center includes that pixels. Though many windows are evaluated, they can be evaluated relatively quickly by generating an integral histogram over the image. Finally, the segmentation is refined by defining a pixel-wise CRF on the visual utility map and running graph cuts [232].

Cheng et al. also define the objectness as a distance between colors in LAB space, except that they define it globally across the entire image instead of just within a specific window. Thus colors that are

far away from all the other colors in the image will have a higher objectness [50]. Shi et al. expand this concept and use the global contrast along with an image prior and a region filtering technique to get accurate boundaries quickly [256]. Mai et al. combine many features for a given image in order to generate the saliency map. The features with highest importance are determined by mining a large database of images to find those that are similar [191].

Thus far, most of the objectness measures have been built upon an intuition of what makes an object an object. Alexe et al. took a different approach and learned an objectness metric for a given bounding box from data. They trained an objectness classifier using 50 images from different object classes in the PASCAL VOC 2007 dataset. Their classifier is a Naive Bayes classifier that combines a multiscale version of the spectral saliency metric from [121], the color histogram distance in the center and surround regions, the edge density and the number of superpixels that straddle the region boundary [4]. Rahtu et al. extend this work in two ways. First, Alexe et al. identify candidate bounding boxes for objects by sampling uniformly spatially and in scale, while Rahtu et al. learn a distribution of image boxes found in a frame from the PASCAL data and then sample from that distribution to get candidate boxes. The resulting boxes are more densely sampled in locations where an object tends to appear in a still frame. The second improvement is to propose a different feature set. They use Alexe's superpixel metric along with the distribution of oriented gradients near the edge of the window and a measure of window symmetry. The resulting classifier is faster to compute and more accurately matches the bounding boxes for the objects [233].

An unsupervised approach for modeling objectness was presented by Siva et al. They hypothesize that objects will be composed of patches that are relatively rare. Therefore, the objectness around each pixel is inversely proportional to the probability of the n x n region around that pixel being found in the corpus. Bounding boxes are then sampled from this pixel-by-pixels map [262].

Objectness can also be defined for superpixel regions without having a valid per-pixel definition. Endres and Hoiem take this approach and learn an objectness using statistics about the boundary of the region combined with a comparison of color and texture histograms with both local and global areas. They then use this definition to rank a set of candidate segmentations that are generated using a hierarchical segementation scheme [77]. Defining objectness for a region, can be useful for segmentation or object classification tasks, however, it does require an initial segmentation to be calculated, which can be computationally expensive.

2.6 Applications of Saliency and Other General Visual Utility Estimators

Saliency and other visual utility estimators have been used in a variety of different applications. In data processing tasks, a couple of studies have used saliency in image compression to change the resolution of areas in a frame so that areas with low saliency are encoded with less resolution and thus take less space [107, 125]. In another study, Longhurst et al. showed how a saliency map could be used in the visual world by speeding up the rendering time of complex graphics [186]. In a different task, Lienhart et al. use a visual utility metric biased for text in order to extract and read text in scenes. These extracted words are then be used in traditional information retrieval frameworks to add richness and data to search [179]. Another study, by Van De Weijer et al. looks at the stability of using a color based visual utility in order to better select points of interest for feature detectors like SIFT [292]. Shahbaz Khan et al. then extend this concept with a bag-of-words classifier in order to perform object classification on the PASCAL VOC 2007 dataset [255].

Visual utility has also been incorporated on mobile robotic platforms. A collection of experiments have used bottom-up metrics of visual utility to identify landmarks for navigation and mapping. These robots have operated indoors [91, 199, 284] and outdoors along walkways [259]. In contrast to these works, this thesis focuses on robotic systems that operate in unstructured environments and use their cameras for

other tasks than navigation.

Social robots have also used visual utility models to recreate human eye movements and thus interact more naturally with people and learn about their environment in a biologically plausible way. Butko et al. use a spectral residual model to allow the robot to quickly home in on human faces [38]. Ruesch et al. combine visual and auditory saliency metrics to direct the motion of a robotic head and interact with its environment [242]. Finally, Orabona et al. use saliency on a robot to help direct its foveal eyes to learn about different objects over time [217].

Finally, a few studies have used visual utility metrics to improve mid-level vision tasks which are then used by a higher level task. For example, Chen et al. use the saliency estimator from [183], which has tight spatial boundaries, in order to segment objects from their background automatically across a very large dataset. They then use those objects to compose images from a user's sketch of the scene [48]. In another example of higher level tasks, Schauerte et al. develop a system that can identify objects that a person points to. They use saliency to help locate which object the person is pointing to and then apply a higher level object classifier [250].

Although multiple groups have shown the benefits of using visual utility in complex tasks, there has been little work relating the accuracy of different visual utility measures to resulting task performance. Most research either compares the accuracies of different measures or uses one measure and determines the impact on the given task. Furthermore, to our knowledge, nobody has developed guidelines for when a task specific visual utility metric is worth developing instead of using a general purpose bottom-up approach.

In the following sections, we examine how visual utility has been applied in active and reactive vision systems.

2.6.1 Active Vision

A visual utility process can also be found in active vision systems. Active vision is the process of actively changing the parameters of a sensor in order to reduce some uncertainty in a model over time [16]. In a camera, these parameters include orientation, position, focus and zoom. While this definition of active vision encompasses many tasks, most active vision work has been in 3D reconstruction, mapping and exploration because in these tasks, the uncertainty in the model is well defined by the portions of the world seen and not seen. Solutions to these problems invariably require a complex map of the world along with an observation model of the sensor in order to predict future measurements and a planner to move the sensor to the new location.

In the following sections we will outline the prior work in 3D reconstruction, robotic exploration, simultaneous localization and mapping (SLAM) and object classification. For more detailed reviews see [73, 74, 269, 274].

2.6.1.1 3D Reconstruction

Active vision has been used to efficiently reconstruct 3D objects. Early systems use range sensors on the end of robotic arms and try to reconstruct and recognize simple objects from a known set. For example, Hutchinson and Kak use a database of known objects to generate a set of hypotheses about a scene. Their range scanner takes successive scans of a scene until the objects are uniquely recognized. After each scan, they update the valid hypotheses and then plan the arm's operation to capture the next measurement that will minimize the ambiguity in the remaining hypotheses [123]. Califano et al. also use hypotheses to recognize objects from a range scan. However, instead of moving the physical sensor, they selectively process different parts of the scan in each case starting with those areas that are not well supported by a feature or object hypothesis [40].

Cameras have also been used for 3D reconstruction in active vision systems using structure from

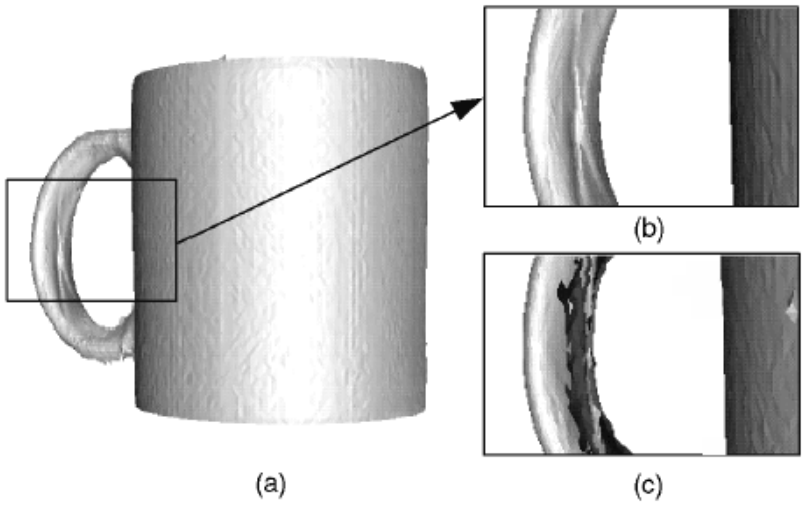


Figure 2.8: Example reconstruction of a coffee cup with (b) and without (c) a next-best-view algorithm. Figure from [228]

motion algorithms [208] or a stereo head that acts as a range sensor. Aloimonos et al. prove that the 3D reconstruction of a scene using structure from motion is unstable if the camera is passive, but tractable with active control [6]. Shmuel and Werman use this result to define motions that will allow a camera to recover depth measurements in a scene using multiple pictures [257]. Finally, Abbot and Ahuja describe how to merge measurements from a stereo pair with different vergence and focus parameters in order to reconstruct 3D information [1].

The next-best-view problem is one of the most studied formulations for solving the active control for 3D reconstruction. In this formulation, the problem is simplified to a greedy approach where given the current reconstruction of the scene and a model of the sensor, what is the next measurement that should be taken in order to best refine the reconstruction [57]. Maver and Bajcsy present an algorithm for range sensors that identifies the shadows in the range measurements and then plans motions in order to fill in those shadows [198]. Banta et al. use a similar approach but instead of identifying shadows, they look for areas of high curvature that will correspond to the edges of objects. They then plan a new measurements that maximizes the amount of area beyond those edges that will be seen [20]. Finally Kutulakos et al. combine a laser scanner algorithm with a camera. The camera is used to identify occlusion boundaries and direct the camera towards them [166]. While the laser scanner is moved towards the surface normal at the horizon of the images in order to create smooth reconstructions [165].

Instead of looking for occlusion boundaries and object boundaries, some approaches try to directly reduce the uncertainty of the reconstruction. Shmuel and Werman use a gauss-markov model combined with a kalman filter in order to define the scene and the uncertainty about it. They then determine analytically which camera motions will not give new information and run an optimization on the other possible motions in order to reduce the model's uncertainty after taking next picture [257]. Whaite and Ferrie assume that the world to consist of a series of superellipsoids. The uncertainty in the scene is then captured by the covariance matrix in the parameters of the superellipsoids [310]. They then plan trajectories for a sensor on the end of a robot arm that will minimize the determinant of the covariance matrices and thus the uncertainty in the model [311].

Planning motions to only get to the next-best-view is not the optimal way to create a reconstruction because it does not take into account the cost of the motions. Therefore, Marchand and Chaumette combine the volume of unexplored space expected to be seen with a metric for the ease of getting to that location with the actuator [194]. Pito adds another metric and includes the amount of overlap with the

current model in order to ensure that enough overlap is captured so that measurements can be registered easily. Pito also projects camera orientations into a new space called positional surface space which makes the search for the next-best-view more efficient [228].

3D reconstruction of a scene is not always the final goal. In some robotic systems, the end goal is to be able to manipulate the environment, which is easier if the locations of objects are known. Ballard studied such systems and showed that when active gaze control is used, it becomes computationally more efficient to determine objects' locations relative to each other and the robot because additional constraints can be imposed on those calculations [17]. In the same spirit of Ballard's result, this thesis explores techniques to solve a simpler vision problem in order to improve performance of a more complex task.

2.6.1.2 Mapmaking and Environment Exploration

Along with reconstructing 3D objects, active vision has also been used for exploring environments and creating maps using mobile robots. In these settings, the model is the resulting map and the uncertainty in that model is a combination of the areas seen/unseen and the uncertainty of the localization of objects. In an early study, Kleinburgh presents a 2D search algorithm for a mobile robots that travels the minimum distance for exploring rectilinear polygons. The algorithm identifies cave mouths, which are corners in the walls where the space opens up beyond them. They then keep these cave mouths on either side of the robot during travel [155].

Kruse et al. extend mapping to a bounded 3D occupancy grid where each box is either filled, empty or unknown. They develop a planning algorithm that maximizes the expected information at the next location while minimizing the cost of getting there [162]. This paradigm has also been used for 2D environments like indoor corridor navigation (e.g. [76, 320]).

Unfortunately, planning using an occupancy grid can fail if the localization of the robot drifts because the entire map will be shifted relative to the robot. To help solve this, Thrun identifies landmarks using a neural network and then plans movements in order to minimize the localization error when navigating by the landmarks [278]. When navigating by landmarks, it is also import to be able to measure the location of those landmarks very accurately. Murray et al. present a control strategy for robots that have an actuating head that fixates on landmarks as the robot moves by in order to improve the localization of the landmark [206]. This work uses the results from Fermuller and Aloimonos where they proved that this kind of dynamic fixation enables more effective navigation [85].

In the mobile robot exploration framework, there is uncertainty in the map that derives from inaccurate sensor, inaccurate motion models and changing environments. The first two sources of error have been modeled using techniques like Kalman filtering (e.g. [59]), which assumes gaussian error distributions, and particle filtering ([240, 279]), which samples from an arbitrary error distribution.

Though these techniques provide a method for quantifying the uncertainty in the world, there still remains the question of what is the next optimal move. Cassandra et al. model this decision process as a partially observable Markov decision process (POMDP). POMDPs provide a grounded method for making decisions in the face of uncertainty, however, they are computationally expensive to solve. Therefore, Cassandra et al. provide some heuristics to make the computation tractable when navigating [46]. Smith provides a more thorough analysis of POMDPs and applies them to Science Autonomy, which enables an exploration vehicle to autonomously search a terrain to make measurements that will most likely inform a scientific hypothesis. In Smith's case, he builds a map of life of an arid desert by combining two types of measurements: camera images that are fast to capture but very inaccurate for finding organic matter, and a fluorescence imager that is slow to use but can accurately detect the presence of organic matter. Therefore, the true measurements across a wide area of the map are difficult to acquire and so Smith uses POMDPs in order to plan operations that will build the map efficiently [264]. Unfortunately, building a plan can take a long time (20min in Smith's case) so follow-on work by Thompson et al. examines some faster, less accurate techniques and builds geologic maps. For example, in one experiment, they use a rock classifier on the camera imagery in order to determine which rocks should be targeted for more in-depth,

time consuming spectrometer measurements [275]. In another experiment, they use overhead satellite imagery to create a prior estimate the locations of different regions and then plan operations to sample efficiently sample in that space [277]. Inspired by this work in Science Autonomy, this thesis examines an even simpler framework that can be used in realtime for any visual task to more efficiently sample the world and thus improve the quality of the data captured.

The next-best-view paradigm has also been used in mapmaking. Similar to the work in 3D reconstruction where the next-best-view is determined by finding occlusion boundaries, Yamauchi et al. plan operations to maximize the frontier space that the robot will encounter as it explores planar indoor environments [320]. Gonzlez-Baos et al. use a slightly different approach and use a metric that includes the expected number of cells that will becomes known and the distance to move to the location while ensuring that there will be enough overlap in the measurements in order to register the new data [101]. Makarenko et al. have a similar approach but also include the expected localization quality once the robot arrives at the location to take the next measurement [192]. Burgard et al. use the next-best-view paradigm for teams of robots. In their solution, each robot calculates the next-best-view however, expected utility of a location will be inhibited if it will be visible by another robot when it gets to its next goal. As a result, the robots tend to spread out and cover the area effectively [36]. Finally, large occupancy grids are not always used. Kakusho et al. use a more compact representation and represent the space as a skeleton graph where each node is a small grid. The next-best-view is determined by searching the skeleton graph to find close nodes whose grids are unknown. Then, as the robot moves the grid is filled in and new nodes are added to the graph as necessary [146].

Due to computational limitations, most of the robotic mapping work that we have described thus far has simplified the planning problem to 2D indoor environments like corridors and classrooms. However, this thesis is most interested in tasks for unstructured real world environments. Robots have explored and built maps in these environments too, but the techniques are more complicated. For example, Sujan and Dubowsky also apply a next-best-view paradigm to teams of robots in unstructured environments. However, to handle the more complicated environment and poorer motion model, they use a gaussian probability distribution to fill out the occupancy grid instead of assigning the entire measurement to a single cell [268]. Nabbe takes another approach in his thesis. He noticed that mobile robots are limited by the range of their sensors for detecting obstacles and that a planning algorithm cannot identify an efficient path if it has not seen the terrain. Using this insight, he introduces a forward simulation technique that predicts the terrain beyond the sensor's horizon and then plans operations in order to make the best observations for navigation en route to the goal. The resulting system will show behaviour like driving to the top of a rise to get a better view [207].

2.6.1.3 SLAM

In another active vision paradigm, when mobile robots explore a scene, there will be uncertainty in the map and in the robot's location. Solving for these uncertainties together is called Simultaneous Localization and Mapping (SLAM). SLAM is generally solved by incrementally building a map as the robot moves around the scene assuming that the localization is correct. Of course the localization will drift over time. Then, when a known location is identified again, the localization solution is adjusted until the maps align, a process known as loop closure. SLAM has been demonstrated in 2D domains (e.g. [104]), 3D domains (e.g. [82]), with multiple robots (e.g. [90]) and even using sensors that are controlled by an external agent (e.g. [64]). When using SLAM to explore an environment, there will be a trade-off between expanding the horizon and returning to known locations to close loops. On one hand, closing loops often minimizes the drift in the localization, while on the other hand, it takes time and requires capturing more redundant data. Vildal-Calleja et al. address this problem by maximizing the expected mutual information gain between the measurements and posterior state of the map and robot. Therefore, it is possible to balance the expected information gain from exploring new areas with the information gained by improving the localization [296]. Tovar et al. take a similar approach, but they model the geometry of the scene explicitly and then use this known geometry to predict what will be visible from the next location [283]. Fox et al.

has a similar approach for multiple robots exploring a scene where the maps between robots must also be merged [90]. Finally Sim and Roy do not use a greedy approach for exploring the scene. Instead, they us an a-optimal formulation given an initial estimate of the scene [260].

2.6.1.4 Active Vision for Classification

Active vision techniques can also be used to improve object classification in mobile robots. Dima noticed that many robots capture large amounts of monotonous, similar imagery with the occasional surprising picture. If this imagery is used to train a classifier, then this unbalanced prior will break many machine learning techniques. Even if the technique is resistant to biased labeling, many excess images will be labeled for the training. In order to help solve this problem, Dima uses active learning techniques on the imagery in order to identify the subset of images that if labeled will enable the classifier to perform effectively [70].

We have also seen excess imagery negatively impact the performance of our marine species classifier (appendix A). In an experiment, a classifier was built using labeled fish that were tightly segmented. This classifier uses a nearest-neighbour image search technique and will return up to five scored matches. We then queried the classifier with instances of different sized bounding boxes around known fish and measured the accuracy of the correct result being found in the first N returns. The results are shown in fig. 2.9. As the bounding box increases, we see the accuracy drop significantly, thus demonstrating that the excess data used in the query is adding noise and confusing the classifier.

In this thesis, we explore image filtering using visual utility that is designed to minimize the excess imagery processed and passed to the high-level algorithm. Using this technique, we show that it is possible to reduce the noise in the signal and thus be able to improve some tasks like object classification. Unlike Dima's work, we are not using an active vision technique with a feedback loop.

2.6.2 Reactive Vision

Reactive vision is often called active vision but it does not fit Bajcsy's definition [16] that active vision is to purposefully control a sensor in order to reduce uncertainty in a model. In reactive vision systems the sensor is actively controlled but the controller does not keep any state information and thus does not have a model to improve. Instead, reactive vision systems react to changes in the environment and direct the sensor to events of interest. The framework we explore in this thesis is not reactive vision because it does not necessarily control a sensor.

In this section we will detail two main fields that have used reactive vision. First, we will consider structured environments like automated lecture hall and conference rooms. Next, we will examine surveillance systems.

2.6.2.1 Structured Environments

Some reactive vision systems rely on the structure of the environment in order to simplify their task, while in this thesis, we are interested in unstructured environments where structural assumptions cannot be made. One structured setting is the conference room. These conference rooms can be used for teleconferencing through the internet in order to save time and money on travel costs. Many commercial teleconferencing systems either use a wide angle field of view to see all the participants or have a pan-tilt-zoom (PTZ) camera that is controlled by one of the parties [88]. However, there has been interest in creating a system that is more interactive and will show whoever is speaking at a given time. In an early study, Wang and Chu describe how to use an array of four microphones to triangulate the speaker and then use this signal to direct a camera to that person [304]. Liu et al. take a different approach and use a panoramic camera

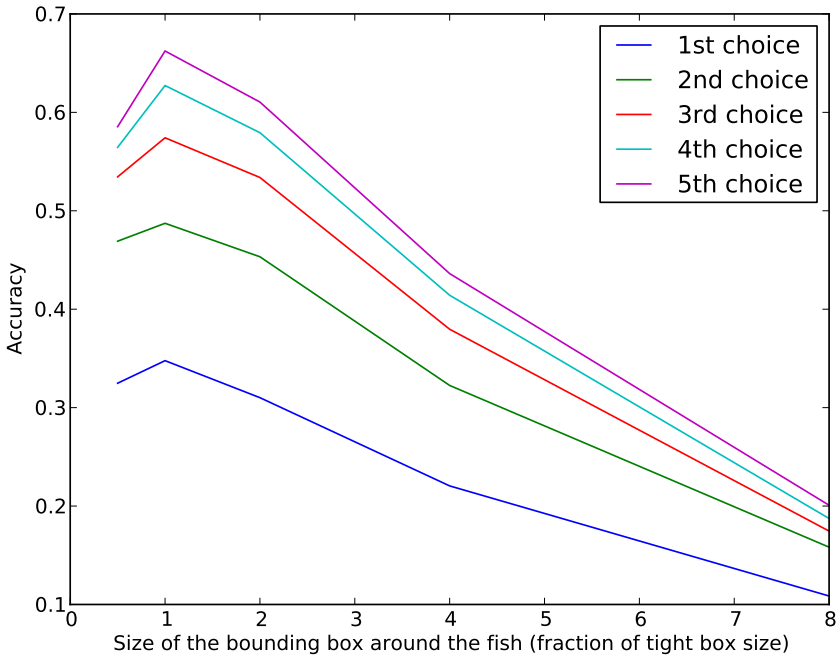


Figure 2.9: In the fish classification task in appendix A, the relationship between the size of the query bounding box and the fraction of the time the correct answer is in the 1st-5th results. As the bounding box increases beyond the tight box at 1, the accuracy drops, showing that the extra data captured is adding noise to the classification.

combined with a PTZ unit. They then allow many remote viewers to define what areas they would prefer to see and use an automated algorithm to optimize the viewing location for all the viewers [181].

Another structured setting is the lecture hall. A few systems have been created to automatically record lectures efficiently so that it is more interesting to the viewer. Rui et al. use multiple cameras, some of which point at the audience and some of which point at the speaker. They then use microphones to triangulate the location of the speaker whether in the audience or at the podium and finally direct a camera to capture the speaker [243]. Lampi et al. only use visual information. They combine a wide field of view camera with a PTZ unit, both pointing at the front of the room. They then use motion and face detection in order to identify the location of the lecturer and direct the PTZ unit to frame and follow the speaker effectively [168]. Finally, Liu et al. describe a system that combines multiple PTZ cameras with wide angle feeds. Their system is designed for viewing the lecture live on the web by multiple participants while minimizing the amount of video data that needs to be transmitted. For each viewer, the system composes a custom feed by merging into a single frame, live high resolution data from the PTZ units, cached high resolution data if the area has not changed since the last PTZ unit viewed it and low resolution data from the wide angle feeds [182].

Reactive vision with structured environments can also be found in attempts to automatically film scenes for human viewing. In an early example, Pinhanez and Bobick automatically film a cooking show. They create a 3D models of the relevant objects in the set like the chef, the table, and pots and then find those objects in the camera shots to identify the current state of the environment. Then a director specifies high-level camera motions and the automated system frames the best shot [227]. In a more recent example, Wu demonstrates a system to automatically film hockey games by tracking the players using multiple cameras. Frankly, I'm including this example because I love hockey and it's tangentially related

[317].

2.6.2.2 Surveillance

Surveillance systems are another example where reactive vision is useful. When using a surveillance system, one wants to simultaneously cover a large area to make sure that a target is not missed and acquire high enough resolution information about each target in order to be able to get a positive identity. In fact, a surveillance system can be described as a specialization of our general purpose control framework where the visual utility is defined by a person detector or similar metric, potentially augmented with a trajectory prediction. Batista et al. demonstrate a system with a binocular head that can move and has a small field of view and a static camera with a wide field of view. The static camera is used for tracking and predicting where people will move and then the binocular head is directed to collect the high resolution information [22]. Costello et al. use a similar approach but with multiple PTZ cameras while experimenting with different scheduling strategies like First Come, First Serve [58]. Comaniciu et al. use a more explicit biological approach and incorporate face detection to identify the targets for a log-polar foveal camera with saccadic motion [56]. Goodridge and Kay also incorporate audio cues to locate the targets [102]. Finally, Del Bimbo and Pernici predict the trajectories of the targets and then formulate the scheduling problem for the high resolution cameras as a dynamic vehicle routing problem with deadlines [67].

In surveillance systems with multiple PTZ cameras, there is also the problem of handoff where a target moves outside the range of one camera but still needs to be tracked. In that case, another camera must be scheduled to pick up the target when the target becomes visible to it. Erdem and Sclaroff solve the calibration for this problem dynamically by keeping track of instances when the same object is visible in more than one camera. Over time, many instances are collected and the transform between cameras is refined. Once the calibration is acquired, the scheduling is done by adding the target to the camera's local priority queue [78]. Instead of using a local priority queue for each camera, Qureshi and Terzopoulos plan simultaneously across all of the cameras using a greedy best-first search to find a solution quickly [231].

2.7 Effectiveness of Previous Visual Utility Estimators

Now that we have provided an overview of many different areas that use the concept of visual utility, let us dig more deeply into some works and examine the effectiveness of different visual utility estimators. Remember that a visual utility based system has two potential benefits: to speedup processing and to improve performance by reducing false positives and thus noise. As we show in section section 3.4.1, an effective visual utility estimator must:

1. Be fast to compute relative to the computational cost of the high-level algorithm.
2. Have a high recall for those elements that are used by the high-level algorithm.
3. Minimize the number of examples passed to the high-level algorithm.

However, this theoretical underpinning does not specify how to build or find such an estimator. So, we will look at the prior work to find examples of effective and ineffective applications of visual utility and from that identify some common characteristics to help a designer build or choose an appropriate visual utility estimator. In particular, we will describe many examples, but most of the effective visual utility estimators are those that are tailored for the specific high-level algorithm and task at hand or those that operate in a scenario where the high-level algorithm is extremely expensive.

2.7.1 Generic Visual Utility Measures

Let us start by examining some instances where it is unclear if the visual utility based approach is effective. In particular, we will focus on the generic visual utility estimators like saliency and objectness. These bottom-up measures are based on low level cues and claim to provide a speed benefit for any task. As a reminder, in this discussion, we are using the term *saliency* to strictly mean those measures that attempt to predict the motion of a human eye when looking at a scene. In the literature, the term *saliency* is also used for generic visual utility measures that could be applied to other tasks than those tested by the authors. Objectness is a common subset of these measures that predicts the likelihood that a region in the image is part of an object. It is thus tailored for a large class of object segmentation and detection tasks.

Saliency was originally proposed by Koch and Ullman [157] and codified by Itti et al. [129] as a model of the human visual system. This work was conducted primarily in order to help understand and model the visual processing centers in the human brain. Itti et al. even make an argument for the computational advantages of this model, but they view the computation with respect to the human visual system, which has a massively parallel engine in the V1 that can almost instantly calculate simple features at every location in the field of view [129]. In contrast, computer vision systems are primarily serial machines and thus, the calculation of those simple features, which become the visual utility estimator, takes more time. This is equivalent to the estimator cost term (D_H) from eq. (3.20) in our theoretical formulation (chapter 3) becoming significant.

Even still, other studies have claimed the potential of faster processing in a computer vision system if a generic visual utility measure like bottom-up saliency is used [4, 5, 13, 38, 50, 55, 95, 121, 127, 142, 226, 233, 249, 273, 330]. However, in all of these studies, the experimental methodology is incomplete. None of these studies actually measure the resulting total system speed when the visual utility measure is used to filter the data for some task. Instead, they examine the resulting output of the visual utility measure, either in the form of a saliency map [50, 226], list of attention targets [13, 127, 142], or as a feed that would be used as input into a higher level process [4, 5, 38, 55, 121, 233, 249, 273, 330]. This output is then compared directly to previous studies or a random approach. However, this analysis ignores the fact that computing the visual utility metric takes time. In other words, they are ignoring the D_H term when it cannot be ignored. Since D_H is often significant, it is not valid to compare the potential impact of two visual utility measures based on the same sized outputs. Of course there could be a benefit, but the analysis conducted by the authors cannot show it. In the experimental section of this thesis (chapter 7), we perform a full analysis for pedestrian detection and show that actually, the generic visual utility metrics do not perform any better than a random sampling approach.

This flawed analysis is particularly prevalent in two objectness papers by Alexe et al. [4, 5]. In these papers, the authors specify a generic object detection algorithm that is equivalent to a visual utility filter. They then explain how to apply said algorithm to try and speed up the HOG detector from [60], the deformable parts detector from [83] and the bag-of-words detector from [167]. In their analysis, they then test their technique on the PASCAL VOC 2007 dataset and examine the number of windows processed by the high-level algorithm and compare that with the loss in mean average precision (mAP). They present a loss of between 2-4% in mAP, while examining between 15x-60x less windows and therefore claim a "massive speedup". However, this speedup assumes that computing objectness is free relative to the computational cost of the high-level algorithm. This is an invalid assumption. On a 2.5 Ghz. Core 2 Duo with 4 Gb of RAM, on 640x480 pixel images, computing the objectness takes 4.45s, while running the high-level algorithm on the entire image takes 0.63s, 1.36s and 0.05s for the HOG detector, the deformable parts model and the bag-of-words respectively. Full timing for many different approaches are available in table 7.1.

Now that we've seen some cases where generic visual utility estimators are not significantly helpful, let us examine some examples where they can improve the performance of high-level algorithms. In particular, they can be very valuable in cases where the high-level algorithm requires stable detections but that do not necessarily correspond to full objects as humans would define them. For example, saliency measures have been used to identify *proto-objects*. Proto-objects are visually distinctive regions in the image that

could potentially be combined into a real object. In other words, they are mid level visual entities between the low level filter responses and the high-level object. In a sequence of papers from Koch and Perona's group, they show the use of proto-objects in an object discovery scenario. In this scenario, proto-objects are extracted from a sequence of images and put into a dictionary built using a SIFT bag-of-words model. Using this framework, they show that previously seen objects can be recognized more easily if a saliency measure is used to extract the proto-object compared to simply selecting a random patch of the image [245, 302, 303]. Saliency, a generic visual utility estimator, is effective in this scenario because it limits the data in the object models to those that are visually distinctive. As a result, the model for a given object has a higher signal to noise ratio, which significantly reduces the false positives (the C_F term in eq. (3.20) from our theoretical formulation in chapter 3).

Similar to object discovery, proto-objects that are extracted using saliency have also been used to identify landmarks for navigation. Siagian and Itti combine the landmarks with a rough scene classifier in order to perform localization on a robot [259]. Frintrop et al. go a step further, and use the landmarks in a visual SLAM procedure and show that the regions are stable enough for long term loop closures [91]. The fact that the landmarks are stable enough to enable SLAM shows that the generic visual utility measure sufficiently filtered the set of candidate regions to be valuable for this task. Furthermore, the runtime of SLAM is known to be dominated by the landmark matching procedure, not feature extraction [71]. This in turn is dependent on the size of the set of previously seen landmarks that must be searched through. Therefore, using saliency will also speed up SLAM compared to randomly sampling for features because it will generate less landmarks.

Contrary to much previous work, generic visual utility measures do not clearly improve the speed of high-level tasks. However, some generic measures like saliency can be valuable for tasks where a large quantity of data must be reduced to a smaller set of visually stable regions, like landmarks for SLAM. In these scenarios, the high-level task performs better because the resulting regions are by design, visually distinctive and thus can be reliably re-extracted.

2.7.2 Task Specific Visual Utility Measures

We have examined some cases where generic visual utility measures have been used to try and improve the performance of a high-level task. The results are mixed. We will now examine some cases where task specific measures were used effectively. A task specific measure is one that is engineered to more accurately identify regions that will be used by the high-level algorithm. This engineering could be in the form of learning, or deliberate design. In the saliency literature, this would be referred to as a top-down saliency metric. As these metrics have been tuned for the task at hand, they can often be computed more quickly than a generic measure and thus reduce the computational cost for a given increase in the number of false positives.

In our first example of a task specific visual utility measure, let us examine the object recognition system of Shahbaz Khan et al. [255]. In this system, the high-level algorithm is a bag-of-words object detector, where the words are greyscale SIFT features. The visual utility measure is built using a Bayesian color model that is calculated for each object category from some training data. The SIFT features are then sampled more densely in regions of the image that have a high color score. The resulting word histogram is used for classification. The authors show improved classification performance on the PASCAL VOC 2007 dataset compared to two bag-of-words baselines, neither of which has a visual utility step. The first uses the same greyscale SIFT features, while the second uses WSIFT features [291], which are calculated using the opponent color channels and thus incorporate color information. This improved performance shows that the category specific visual utility measures are able to improve the high-level algorithm.

In our next example, let us look at Dima's work on active scene classification in [70]. In his thesis, Dima recognizes that in continuous obstacle identification problems where a video feed is used to identify obstacles for a mobile robot, most of the video feed does not contain objects of concern. In this scenario, if one were to sample randomly from the video sequence in order to label the data, most of the labels would be

for scenes that do not contain obstacles. This has two issues. First, it can bias any resulting classifier built on this data and second, it is very time consuming to label all that data. To address these two problems, Dima uses an active approach to identify which examples need to be labeled. In particular, he iteratively builds his classifier by identifying, at each step, a set of examples that would be most discriminative for the classifier if they were to be labeled. We can treat this like a visual utility process where the high-level task is to build a classifier that can identify obstacles. If we consider the cost of labeling as a part of this process, then each additional example that must be labeled becomes very expensive (the high-level algorithm cost (D_{R_t}) from eq. (3.20) is large). Then, we can identify the visual utility measure to be the discrimination metric that is recomputed during each iteration based on the current state of the classifier. It is task specific because its form is dependent on the algorithm used by the classifier. Dima shows in his experiments that using these techniques, less images need to be labeled, thus speeding up the overall processing for training the classifier and the classifier performs more accurately because it is less biased.

Our final example of specific visual utility measures are those that directly estimate the high-level algorithm, but are able to perform that estimation much more quickly. The first class of these examples are coarse-to-fine detectors. They were originally proposed by Burt in 1988 in order to execute computer vision algorithms in a reasonable amount of time [37]. In these detectors, the visual utility estimator is simply a lower resolution version of the detector that is calculated on a lower resolution version of the image. This is significantly faster than the full resolution detector because in a naive implementation, a sliding window detector has a computational complexity of $O(mn^2)$ where $O(m)$ is the computational complexity of the detector and n is the size of the image. Furthermore, the low resolution detector will have a high accuracy because it is directly calculating an approximation of the high-level detector. Also, it is trivial to define a full hierarchy of resolutions in order to have multiple visual utility steps.

When Burt proposed the coarse-to-fine approach in 1988, computer vision was in its infancy and computers were not powerful enough to perform significant experiments. However, more recently, others have used this approach on more modern detectors and on large experiments. In the first case, Zhang et al. showed that a four stage HOG detector can operate 5x faster than the full resolution detector on the pedestrian detection task from [60] and on the car and motorbike categories of the PASCAL VOC 2006 challenge. Furthermore, the accuracy of the detector can actually be increased by incorporating the HOG responses at all scales of the hierarchy in the final evaluation [332]. Pedersoli et al. extend this approach using a non-maximal suppression at each level in the hierarchy and then performing a finer local optimization. They report a 12x speedup on average for the PASCAL VOC2007 dataset [222]. Then then extend their approach to the deformable parts detector of [83] and show a 10x speedup with no loss of precision compared to the original implementation [223].

Felzenszwalb et al. present another approach to speed up their parts-based detector using an approximation of the final detector as the visual utility estimator. In the deformable parts model, each object is composed of a root filter, defined at a low resolution and a group of smaller parts that are defined at a higher resolution and a location relative to the root filter. The deformable parts model allows the parts to move relative to the root filter, incurring a small cost. This leads to a detector that is more robust to changes in the object and viewpoint. To speed this up, Felzenszwalb et al. use a star cascade composed of multiple visual utility filters, one for each part in sequence. At each part, the visual utility estimator is simply the score of placing the part at a given location, combined with score of all the previously placed parts. If this total lies below a threshold, the candidate is thrown out. The threshold is chosen using a validation dataset to ensure that all positive examples will pass the filter. In other words, the threshold is set to throw out all examples that cannot accrue enough further evidence to pass the detector. Once all the parts are placed, the final score is compared to the same overall threshold used in the original algorithm. The resulting detector performs 20x faster than the original algorithm with no decrease in detection accuracy for the 2009 PASCAL VOC dataset [84].

Recently, Levi et al. have combined the coarse-to-fine approach with a fast search for parts using a restricted KD-Tree, which they call KD-Ferns. This enables hundreds of parts to be used, which increases task performance, while still able to search a scene very quickly. They report processing times of nearly 10 frames per second on 640x480 images [174].

2.7.3 Very Expensive Tasks

We have just shown many examples where task specific visual utility measures prove efficient enough to improve the overall performance of a high-level task. However, there are some tasks where the visual utility measure does not need to be as efficient: those that are already extremely expensive to compute. This will be explored in more detail in chapter 3 but intuitively, there can be a computational gain as long as the computational cost of the visual utility measure is significantly less than the cost of the high-level algorithm. One way to improve this ratio is to have a very efficient visual utility metric, which is easier to do if it is designed for the task at hand. Another approach would be to apply a visual utility framework to problems that already have a very large computational cost. We will now examine some cases where visual utility was used successfully in these high cost environments.

The first case of expensive tasks are those that require training. If we consider the cost and time of labeling images to be a part of the training, then using a visual utility estimator to limit the number of examples that need to be labeled, will improve the speed of the process. We have already discussed one such case that was presented by Dima in the context of learning a obstacle classifier for a mobile robot [70].

Another class of examples are active vision approaches where the cost of acquiring the next measurement is large. For example, this happens when a robot must move around a scene in order to take the next measurement. Though many of the active vision approaches mentioned in section 2.6.1 fall into this category, we'll look at more detail into some science autonomy tasks. In science autonomy, the goal is to maximize the amount of scientifically useful data returned from an exploration rover that could be in a treacherous environment with poor communication (like on another planet). If the rover is able to reason about the science it is helping with, and terrain it encounters, then it can be much more effective. Many of these experiments were performed as a part of the Life in the Atacama project, which operated an autonomous analog planetary rover in the Atacama desert in Chile in order to develop techniques for exploring Mars for life [308, 309]. In one experiment, Smith et al. demonstrate a procedure improve the use of the on-board fluorescent imager. This imager sprays dyes that will fluoresce in the presence of chlorophyll, amino acids and other signs of life. A full measurement takes 25 minutes and there is a limited quantity of dye on-board. In order to minimize the use of the dye, the rover first performs a subset of the sampling procedure at regular intervals where only chlorophyll is detected. This limited procedure only takes approximately 5 mintues to perform. If chlorophyll is detected, then, the full procedure is completed. Smith et al. show that this approach finds significantly more positive life detections for the amount of dye used in comparison to performing the full sampling at every location. Unfortunately, there is no analysis on the rate of detections missed in the process [263]. This procedure is exactly like a visual utility procedure where the visual utility estimator is the first chlorophyll-only detection and the high-level algorithm is the ability to map life. Even though the visual utility estimator is expensive (˜5 min and some dye used), the cost of the high-level algorithm is significantly higher (˜25 min and more dye used), so the procedure proves effective.

A second relevant experiment from the Life in the Atacama project is presented by Thompson as the corridor exploration problem [276]. In this experiment, the rover must reach a certain distance in a time window, but it has the flexibility in its schedule to take an indirect route in order to acquire measurements in different locations. These measurements use an infrared spectrometer in order to determine the mineralogical composition of a location. Multiple measurements are fused with an orbital image using a gaussian point process in order to generate a map of the region delineating different geologic regions. The process of building this map can be viewed as the high-level algorithm, where the performance is measured by the map reconstruction accuracy. Thompson presents a large-scale planner that iteratively re-plans along the path in order to maximize the joint entropy of the expected measurements conditioned on the previous measurements and the orbital map. This planner is equivalent to a visual utility estimator because it is filtering the set of all possible future measurements into the ones that it estimates will maximize the fidelity of the resulting map. The planner requires between 5-30s to execute, which is significantly faster than the motion of the rover at the scale being planned on and thus the visual utility approach has

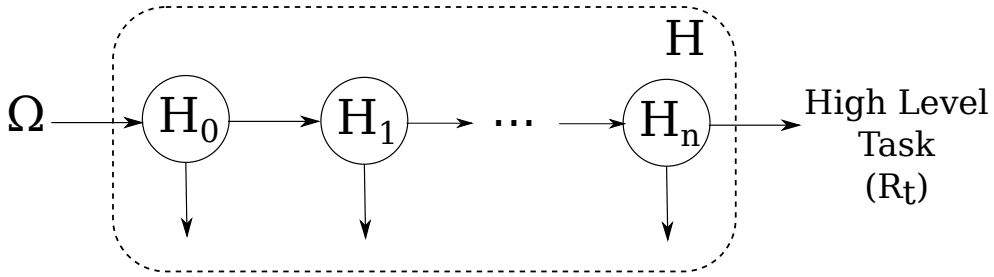


Figure 2.10: A generic visual utility cascade composed of n filters. It filters the set of candidate regions Ω into the set H, while set R_t are those regions that are actually used by the high-level task t.

the potential to improve performance. In experiments, Thompson shows that indeed, using the planner produces significantly better maps than two naive approaches: simply driving to the goal, or performing a regular zig-zag pattern.

2.7.4 Summary of Effective Visual Utility Estimators

This section described in detail a number of different visual utility estimators from the literature. From this analysis, there is little evidence that generic visual utility estimators are effective at improving high-level task speed or performance unless that task only requires stable detections in an image, not higher level image understanding. However, when task specific knowledge is incorporated into the visual utility estimator, there is conclusive evidence of improved performance on some high-level tasks. Finally, using a visual utility framework on tasks that are very expensive does lead to task improvement simply because there is a larger computational budget to calculate a more complicated visual utility estimator.

2.8 Selective Computation For Object Detection

Instead of filtering for a high-level task, there are some proposed object detection techniques that selectively process portions of the image in stages. The most famous of these is the cascade, presented by Viola and Jones in their seminal paper [298] and then expanded upon by others. In this thesis, we show how to build a cascade of any type of visual utility filters that explicitly accounts for computational cost.

2.8.1 Cascades

Cascades reduce the computational cost of object detectors by quickly throwing out portions of the image that are unlikely to contain an object of interest. A cascade is composed of a series of filters. At each stage, only those regions that pass the filter move on (fig. 2.10). This approach has been shown to be quite effective in practice, being used to create real time systems for face detection [298], pedestrian detection [334] and others.

Some previous studies have examined the performance bounds for cascades, mostly as an effort to improve upon the Viola-Jones detector. For object detection, cascades inherently operate in an asymmetric regime of significantly more negative cases than positives cases. The original Viola-Jones cascade handles this by enforcing a maximum miss rate after the AdaBoost learning is complete [298]. Brubaker et al. handle this using an ad-hoc method that re-adjusts the filter thresholds in a second pass to optimize the joint performance [33]. Other approaches reformulate the AdaBoost learning process to incorporate this asymmetry directly [197, 297, 316].

Jun and Jones examine cascades for energy constrained scenarios. They are thinking specifically of physical sensors, which can be cascaded for making a measurement of the environment, but the analysis would work for processing time as well. They assume that for every sensor in the cascade most of the regions will be removed. This is a valid assumption early in the cascade, but later, the assumption breaks. With this assumption, they are able to express the problem as a continuous, convex optimization that is similar to our submodular approach except that our approach stays in set theory and thus does not require their assumption[143].

Sochman and Matas extend the Viola-Jones detector using Wald's theoretical results showing that the optimal strategy in terms of average time to a decision occurs in the form of a sequential probability ratio test [301]. At each stage, an example can either be positive, negative, or more information is needed. Sochman and Matas extend this theory for the object detection case and show how to set the decision boundaries [266]. Bourdev and Brandt have a similar formulation that builds a soft cascade that thresholds on the sum of the responses so far instead of just the response of the current filter [31].

Some theoretical bounds have also been found by identifying more generic scenarios. For instance, Anthony identifies the error bounds for threshold decision lists, of which cascades are a special case [9].

A boosted cascade inspired by Viola-Jones has also been used more recently for multi-class object detection. In their regionlet paper, Wang et al. start by using an objectness estimator to generate candidate object boxes and then build a boosted cascade simple features to detect parts. The resulting cascade can run at 5 frames/second on a single 2.1 Ghz core, but, just like Viola-Jones, the computational time is achieved as a by-product of the boosting process and not optimized for explicitly [306]

The original Viola-Jones detector sped up processing as a side effect of finding of finding an accurate cascade. Chen and Yuille took a slightly different approach and directly minimize the processing time subject to a 0 miss rate and limited false positive rate. Then, binary search is used to find a tight bound on the false positive rate [49]. This is a very similar formulation that we present in this thesis for our generic visual utility cascade, except that we also present the submodularity analysis and are not limited to the case of a 0 miss rate.

An extension of cascades is presented in [272]. In their formulation, the stopping criteria is based on a confidence threshold built based on the responses to classifiers up to that point. This is significant when each stage of the cascade is composed of a tree-based classifier. The resulting cascade evaluates between 2 and 10x less regions [272].

Li and Zhang show that more complex components can be used in cascades to achieve slightly better result. They build cascades using SURF features instead of Haar features, and logistic regression instead of decision stumps in order to reduce training time. This allows many more images to be trained on, thus improving task performance [177].

In this thesis, we also examine the Viola-Jones ([298]) and Star ([84]) cascades with respect to our generic visual utility formulation as a submodular optimization. We show that these two algorithms are special cases of a greedy submodular optimization that is near optimal, but achieves a speedup as a by-product. We then extend this insight to develop the SCATAT algorithm, which explicitly trades-off computational cost and task accuracy.

2.8.2 Submodular Approaches to Speed Up Object Detection

Though submodularity theory has not previously been used to analyze cascades, it has been used for other approaches in object detection. An et al. show how submodularity can be used to speed up a branch and bound object detection search if it is not necessary to guarantee that the best location in the image will be found [8]. Grubb and Bagnell use submodularity theory to create an anytime object detector that focuses its processing on promising regions of the image, but does not throw out regions like a cascade does[105]. In other words, their SpeedBoost algorithm is an example of a soft attention model, whereas cascade and

visual utility are hard attention models. Their formulation is especially powerful because they can adjust the processing on the fly to guarantee the processing time does not exceed a limit on every image. In contrast, a cascade can only be built to limit the average processing time on an image.

2.8.3 Other Approaches To Fast Object Detection

Most work on speeding up object detection is focused on improving the processing time of finding a single type of object in a scene. However, when trying to understand a scene, we would like to be able to find and label many different objects in the scene. The number of potential objects could be very large and yet all of the approaches are still roughly linear in the number of object classes. A different approach, presented by Dean et al. uses hashing on object parts and binary dot products instead of convolutions in order to detect objects such that the runtime is relatively independent of the number of object classes. Using this technique, they build an object detector for 100k+ object types that fits in memory in a single machine and can process an image in a few seconds [66]. In this thesis we examine in Section 7.3.2 how cascades and other visual utility processes scale to the multiple object scenario.

2.9 Submodularity

A *submodular* function is defined over a set $A \subseteq V$ and has the property of diminishing returns, where adding a new element to A in the future can never have a larger effect than adding it now. Formally, let F be a function defined on $2^V \rightarrow \mathbb{R}$. Then, the function F is submodular if:

$$F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B) \forall A \subseteq B \subseteq V, x \notin B, \tag{2.1}$$

Similarly, a *supermodular* function F is where:

$$F(A \cup \{x\}) - F(A) \leq F(B \cup \{x\}) - F(B) \forall A \subseteq B \subseteq V, x \notin B. \tag{2.2}$$

From these two definitions, we can see that if $F(A)$ is submodular, then $-F(A)$ is supermodular. Furthermore, non-negative linear combinations of submodular functions are also submodular.

Submodular functions are interesting because they are the discrete analog to convex/concave function. In other words, they can be optimized relatively efficiently. However, there are different approaches for maximizing and minimizing submodular functions.

2.9.1 Submodular Minimization

Submodular functions can be minimized exactly in polynomial time. In order to do this, Lovasz recognized that every submodular function has an associated convex function [189]. This convex function can then be minimized using the ellipsoid method in polynomial time [106]. However, the ellipsoid algorithm is very slow, so work has continued to speed up the process. The most efficient technique guaranteed to have the correct solution runs in $O(n^6 \log n)$ [132]. In practice, the minimum norm algorithm by Fujishige et al is more useful. The guaranteed runtime is finite and unknown, but in the majority of the situations, it will find the solution order of magnitude faster than Iwata and Orlin's approach [93].

For a more complete review of submodular minimization, see [131]. In this thesis, we require that the submodular optimization runs extremely quickly because the number of candidate filters can be extremely large. Therefore, submodular minimization is too slow and we focus on submodular maximization which does not guarantee optimality, but does present an optimality bound.

2.9.2 Submodular Maximization

The typical problem for maximizing a submodular function is to find the maximum value of $F(A)$ subject to a cardinality constraint. Specifically, it solves:

$$A* = \arg\max_{|A|<k} F(A). \tag{2.3}$$

This problem is known to be NP-Hard, however, the greedy algorithm is known to find a near optimal solution [213]. In particular, for non-decreasing, positive submodular functions, the greedy optimization gives a result that is near optimal because it has a worst case lower bound relative to the optimal solution:

$$F(A_{greedy}) \geq (1 - 1/e) \max_{A \subseteq V} F(A). \tag{2.4}$$

We can also identify "how close" a function is to submodular using the submodularity ratio introduced in [62]. A function F has a submodularity ratio $\gamma_{U,k}$ if, for the set U and parameter $k \geq 1$:

$$\sum_{x \in S} [F(L \cup \{x\}) - F(L)] \geq \gamma_{U,k}[F(L \cup S) - F(L)], \forall L \subseteq U, S : |S| \leq k, S \cap L = \emptyset \tag{2.5}$$

Now, using the minimum submodularity ratio γ over all possible U and k, [62] showed that maximizing F using the greedy algorithm is bounded by:

$$F(A_{greedy}) \geq (1 - e^{-\gamma}) \max_{A \subseteq V} F(A). \tag{2.6}$$

Submodular maximization has been used to create efficient and effective algorithms for areas including image segmentation[138], sensor placement[159] and dictionary selection[62]. For a more complete review of submodular maximization, see [158].

2.10 Summary

In this chapter, we have discussed the biological and computational inspirations for visual utility, namely that most animals' vision systems are equivalent to efficiently filtering the complex visual world in order to enable higher level visual processing on our limited wet computers. We have explored attempts to recreate the biological system either through foveal cameras, or saliency models. Then, we have explored non-biological visual utility models and examine their reported effectiveness for computer vision tasks, most commonly object detection and mapmaking. The full list of applications is summarized in table 2.1. In this analysis, we have determined that visual utility approaches tend to be more useful when:

1. The visual utility estimator is specific to the task at hand.
2. The high-level task/algorithm is complex and thus computationally expensive.

We have also identified a few holes in the literature that we will examine throughout this thesis. In particular, there is little information about the full processing time and thus computational costs, of the whole visual utility process. Many cases ignore the processing time of the visual utility estimator and assume that if less regions are passed to the higher level algorithm, the overall process will speed up (e.g. [5]). Other cases mention that the overall processing time is fast enough to be useful for the task of interest, but dont examine the tradeoffs required to get to that processing time (e.g. [38]). Finally, many

	Unstructured Environments	Biologically Inspired	Large Computation Time	Most Data Not Useful
Object Detection	x	x		x
Object Discovery	x	x	x	x
Saliency	x	x		
Social Robots		x		x
3D Reconstruction			x	
Mapping and Exploration	x	x	x	
SLAM/Localization	x		x	
Surveillance				x
Image Compression	x	x		x
Image Search	x			x
Image Composition	x	x	x	x

Table 2.1: Applications where visual utility approaches have been applied.

works dont consider processing time at all because they are not interested in processing time and are using visual utility for another purpose (e.g. saliency work to understand the brain). Our experimental case studies (chapter 7) in this thesis are specifically targeting at cases where the high-level algorithm is already reasonably fast, but not real-time. This allows us to examine visual utility under the most challenging scenarios to see if it can be used to achieve that last step in speeding an algorithm.

Finally, we provided some background on submodularity, which is used in this thesis because the process of building a cascade of visual utility filters is in fact a submodular process. From this insight, we are able to show that by limiting the set of candidate filters, popular cascade building algorithms are in fact near optimal. We are then able to expand on that insight in order to develop the SCATAT algorithm (chapter 5), which allows cascades to be built by explicitly trading off computational cost and task accuracy.

Chapter 3

Theory

When life gives you lemons, make a daiquiri.

Chris Rose

In this section, we will explore the theory behind visual utility in order to understand in what circumstances it is most useful and how to use it most effectively. We will start by defining visual utility. Then, we will use Bayes Risk in order to determine how to use visual utility in a single frame. Finally, we will extend our analysis to situations where visual utility is estimated over time.

3.1 Defining Visual Utility

Visual utility is a detection process where the goal is to detect those locations in the image that will be useful to the higher level algorithm. If we consider a single image I, then in the most general case, the set of all possible regions to be evaluated can be defined by

$$\Omega = \{I(x, y, \Upsilon)\}, \tag{3.1}$$

where x and y are the coordinates of the region and Υ is a set of dimensions that describe the shape and scale. Note that for many problems, the number of dimensions can be reduced. For instance, if the high-level task involves finding an object with a known shape, then the set reduces to

$$\Omega = \{I(x, y, s)\}, \tag{3.2}$$

where s is the scale of the object. Finally, some high-level algorithms only require points to be found. For instance, the process of calculating a SIFT feature[190] automatically determines the natural scale of the feature. Therefore, we must only find the point of interest and the possible set of regions reduces yet again to

$$\Omega = \{I(x, y)\}. \tag{3.3}$$

We can now define a new set $R_t \subseteq \Omega$ comprised of all those locations in Ω that are (R)eally used by the high-level task t. Then the probability of a region being useful can be defined as:

$$P(R_t) = P(r \in R_t | r \in \Omega). \tag{3.4}$$

Note that the set R_t varies depending on the high-level task and algorithm used and it cannot be found a priori. However, it can be *estimated* a priori using a visual utility estimator that finds a hypothesis set $H \subseteq \Omega$. The probability that a region is found by this estimator can be defined as:

$$P(H) = P(h \in H | h \in \Omega). \tag{3.5}$$

Different estimators will have different performance, and in the following discussion, we will examine the characteristics of effective estimators.

3.2 Visual Utility for a Single Frame

In this section, we will look at visual utility in a single frame and determine how to chain together multiple visual utility estimators in order to create a more effective overall system. To do this, we use the insights from the theory of signal detection and develop an aggregate visual utility estimator that minimizes the Bayes risk in the binary hypothesis testing scenario. In this case, our two hypotheses are whether the region $h \in R_t$ or not. Therefore, the Bayes risk for the visual utility estimator H is defined as

$$BR(H) = C_F P(\neg R_t H) + C_M P(R_t \neg H), \tag{3.6}$$

where C_F is the cost of a false positive and C_M is the cost of a miss. Since we are also interested in using visual utility to speed up complex visual algorithms, we add an additional cost D equal to the computational cost of the entire process. This cost should be proportional to the execution time of the algorithm. The resulting visual utility risk becomes:

$$\nu(H) = D + C_F P(\neg R_t H) + C_M P(R_t \neg H). \tag{3.7}$$

3.2.1 Setting Relative Weights

In the visual utility risk equation, there are three weights that need to be set before using the SCATAT algorithm (chapter 5) to minimize the risk. There is the cost of a false positive (C_F), the cost of a miss (C_M) and the computational cost (D terms). Note that the magnitude of the risk is not important and so there are actually only two degrees of freedom. Even still, these parameters represent the engineering trade off that occurs when designing a system. For instance, when performing an offline analysis with significant computational resources, the computational cost would be very low and perhaps 0 while in a real-time system, computation is more important and the computational cost will be higher. In another example, take our application of pedestrian detection for safety where the system must identify a dangerous situation and alert an operator. The cost of a false positive would be important because too many false alarms can make the system unusable. However, the cost of a miss would be significantly higher because of the possibility of injury or death. Therefore, we will examine how we might want to set these two degrees of freedom.

We can think about the first degree of freedom as being the relative cost of false positives vs. misses. This should be set according to the task at hand. For instance, in a safety critical scenario, a miss will be more expensive than a false positive because it could result in an injury. However, without knowing a special task, it is reasonable to assume that each of the errors is equivalent. In other words, the task is being evaluated based on accuracy.

No matter the relative cost of false positives vs. misses, a system designer must think about these costs relative to the output of the entire system. However, the SCATAT algorithm is training based on errors at the interface between the visual utility filter and the high-level algorithm. For instance, if the

high-level algorithm is very accurate, then the cost of a false positive in the visual utility filter will be low. Furthermore, it is training on those examples that are both useful to the high-level algorithm and are relevant to the task at hand. In the object detection case, this means that the visual utility filter is trained only on those regions that are identified by a higher level object detector and actually contain the object of interest. Therefore, we must adjust the system level desired costs into those for training the visual utility cascade. This can be done by examining the general case where we are trying to train the visual utility filter H, while there is a high-level algorithm R and G is the ground truth. Therefore, the system level cost is:

$$\bar{C_F}P(\neg GHR) + \bar{C_M}P(G\neg(HR)), \tag{3.8}$$

while the costs we are training on are:

$$C_F P(H\neg(RG)) + C_M P(\neg HRG). \tag{3.9}$$

First, we will examine the cost of misses. The system level cost is:

$$\begin{aligned} \bar{C_M}P(G\neg(HR)) =& \bar{C_M}[P(G) - P(RG) + P(\neg HRG)] \\ =& C_M P(\neg HRG) + \bar{C_M}[P(G) - P(RG)]. \end{aligned} \tag{3.10}$$

Now, we can remove the second term because it is a constant offset relative to the filter being trained (H):

$$\bar{C_M}P(G\neg(HR)) = C_M P(\neg HRG). \tag{3.11}$$

Therefore, C_M stays the same. Next, we will examine the cost of the false positives:

$$\begin{aligned} C_F P(H\neg(RG)) =& \bar{C_F}P(\neg GHR) \\ C_F =& \bar{C_F}\frac{P(\neg GHR)}{P(H\neg(GR))}, \end{aligned} \tag{3.12}$$

which, if we approximate that $P(H)$ is independent of $P(\neg(GR))$ and $P(R\neg G)$, becomes,

$$C_F = \bar{C_F}\frac{P(\neg GR)}{P(\neg(GR))}. \tag{3.13}$$

where $P(\neg GR)$ is the false positive rate of the high-level detector and $P(\neg(GR))$ is the probability of a region not being useful. Therefore, when training the visual utility filter, the cost of false positives at the system level must be multiplied by $P(\neg GR)/(\neg(GR))$.

Next, we must address the cost of computation compared to task accuracy. It is more practical to have a formulation as a constrained optimization, where the goal is to minimize the Bayes risk subject to a computational constraint. We can transform our weighted optimization into this constrained optimization by noting that the training approximately smooth; as we increase the cost of the computation relative to task accuracy, the resulting cascade will be faster, but the recall will fall. Therefore, we can set a desired processing speed and perform a robust binary search on the relative weight of the computation cost until we home in on a cascade that operates and approximately the desired speed. This search is still fast because the training of a given cascade is fast due to the problem's submodular structure.

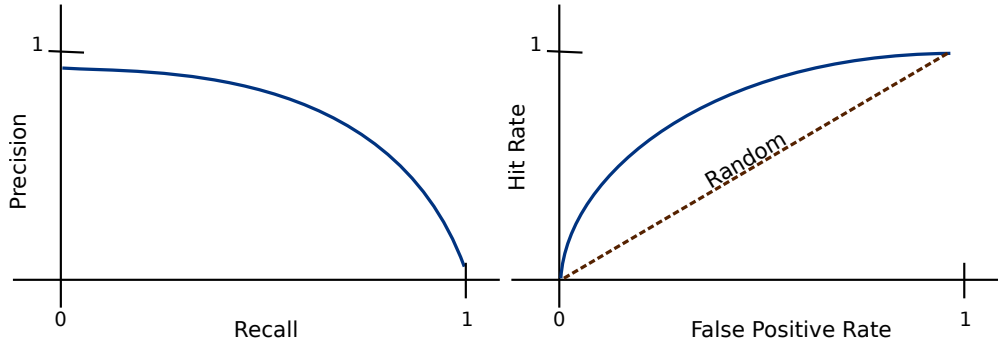


Figure 3.1: Typical Precision-Recall and ROC curves.

3.3 Metrics in Detection Tasks

Now that we have seen the risk metric that we will use throughout the analysis for visual utility, we should pause for a second and discuss other common metrics in detection tasks and why they are used. At times, we will be referring to these other metrics when it makes sense in the context of our discussion.

3.3.1 Precision and Recall

Precision and recall are often used in an information retrieval framework where a set of objects are returned. In this case, precision is the fraction of those object that are actually relevant and recall is the fraction of relevant objects in the world that were returned. Precision and recall are defined using the number of True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN):

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{P(HR_t)}{P(HR_t)} \tag{3.14}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{P(HR_t)}{P(R_t)}. \tag{3.15}$$

Precision and recall represent a tradeoff that is encapsulated in a Precision-Recall (PR) curve (fig. 3.1(a)). Given an ordered set of returns, one can move a threshold for how many of the top ranked objects should be returned. Each different threshold represents a point on the precision recall curve. For different operating situations, there could be a desire to be at a different point on this curve. Note that when minimizing the visual utility risk, increasing the ratio C_M/C_F will move the operating point to the right on the PR curve.

Changing the classifier and/or features involved in the classification can change the overall shape of the PR curve. So, to compare one type of classifier versus another, the average precision is commonly used. Average precision is the area under the PR curve, or formally, given the precision as a function of recall $p(r)$:

$$\text{Average Precision} = \int_0^1 p(r)\mathrm{d}r. \tag{3.16}$$

For a more detailed discussion about precision and recall, see [333].

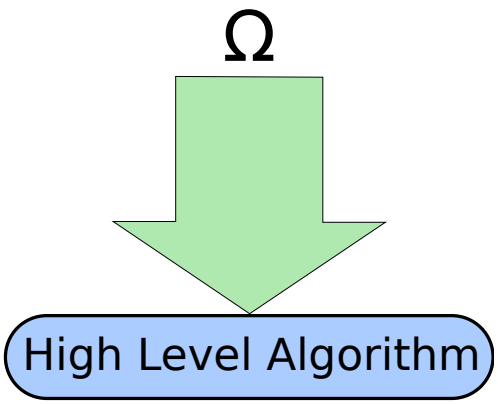


Figure 3.2: The base case with no visual utility filtering where the entire set Ω is processed by the high-level algorithm

3.3.2 ROC Curve

The receiver operating characteristic (ROC) curve is another way to look at detection performance (fig. 3.1(b)). It is a plot of the hit rate vs. the false positive rate where:

$$\text{Hit Rate} = \frac{TP}{TP + FN} = \frac{P(HR_t)}{P(R_t)} \tag{3.17}$$

$$\text{False Positive Rate} = \frac{FP}{FP + TN} = \frac{P(H\neg R_t)}{P(\neg R_t)}. \tag{3.18}$$

The ROC curve is very useful in an engineering situation where there is a design requirement that the detector must be below a given false positive rate. In this situation, one must simply identify the hit rate at the point to compare detectors. This corresponds to a Neyman-Pearson test, which will be discussed more in section section 3.4.2.

There are a few useful properties of the ROC curve. First, a random detector operates on the diagonal. Second, the ROC curve is concave, so trying to optimize along the curve will be an efficient operation that is guaranteed to get the globally optimal solution. Finally, in our situation of minimizing the Bayes risk, increasing the C_M/C_F ratio corresponds to moving left along the curve.

Classifiers can be compared using the area under the ROC curve, however, it has been shown recently that this metric is very noisy to compute [109, 184], and does not weight the types of errors equally for different classifiers [110]. Therefore, we will not use this metric in this thesis and will instead focus on precision-recall curves when comparing models.

3.4 Building a Visual Utility Filter

A visual utility filter is a process that uses visual utility in order to reduce the subset of image locations that are processed by the high-level algorithm. Using the visual utility risk metric from section 3.1, we can now determine if a filter built on a given visual utility estimator will be valuable. To start, let us consider the base case of no filter shown in fig. 3.2.

In this case, all the data is processed by the higher level algorithm and the visual utility risk becomes:

$$\nu(R_t) = D_{R_t} + C_F P(\neg R_t). \tag{3.19}$$

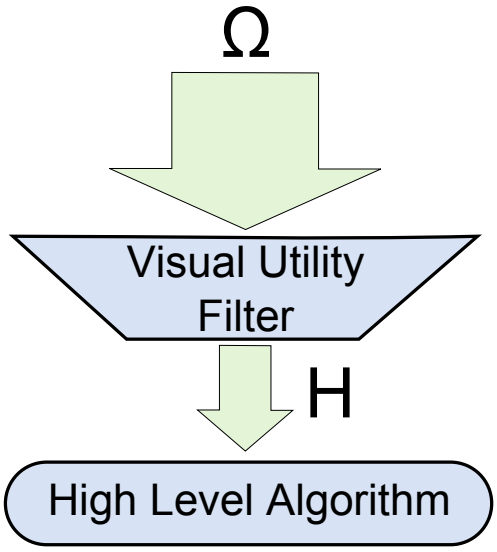


Figure 3.3: A visual utility filter pre-filters the input data into a higher level algorithm. This can potentially reduce computation time and increase the signal to noise ratio.

Then, we can expand eq. (3.7), the risk from a single visual utility filter H, to account for an arbitrary function for the computational cost based on the number of regions processed by the high-level algorithm. In particular, eq. (3.7) becomes:

$$\begin{aligned}\nu(H) =& D + C_F P(\neg R_t H) + C_M P(R_t \neg H) \\ =& D_H + D_{R_t}\tau(P(H)) + C_F P(\neg R_t H) + C_M P(R_t \neg H),\end{aligned} \tag{3.20}$$

where τ is a monotonically increasing function that specifies the computation time of processing a fraction of the candidate regions using the high-level algorithm. For instance, in a common case where each region is processed independently and no optimization is performed, τ would be a linear function of the form $\tau(P(H)) := kP(H) + b$. However, with optimizations, this function could easily be sub-linear.

Now, combing eq. (3.19) and eq. (3.20), we can say that adding a visual utility filter (fig. 3.3) is worthwhile for a given task if:

$$\begin{aligned}\nu(H) - \nu(R_t) &< 0 \\ D_H + D_{R_t}\tau(P(H)) + C_F P(\neg R_t H) + C_M P(R_t \neg H) - D_r\tau(1) - C_F P(\neg R_t) &< 0 \\ D_H - D_{R_t}[\tau(1) - \tau(P(H))] + C_M P(R_t \neg H) - C_F[P(\neg R_T) - P(\neg R_t H)] &< 0.\end{aligned} \tag{3.21}$$

Let us look at each of these components in turn. First, the computational cost:

$$D_H - D_{R_t}[\tau(1) - \tau(P(H))]. \tag{3.22}$$

First, τ is monotonically increasing and $P(H) \leq 1$. Therefore, $D_{R_t}[\tau(1) - \tau(P(H))]$, which represents the computation saved by avoiding computing the high-level algorithm on the filtered regions, is positive and the overall computational cost will decrease if:

$$D_H < D_{R_t}[\tau(1) - \tau(P(H))]. \tag{3.23}$$

We can meet this criteria in a few ways. First, if we could use a visual utility filter that is simple relative to the high-level algorithm. In other words, $D_H \ll D_{R_t}$. Another approach is to use a selective visual utility filter that passes few regions to the high-level algorithm. This type of filter would have a low $P(H)$ and thus, $[\tau(1) - \tau(P(H))]$ would be large.

However, using a selective visual utility estimator has a negative impact on the next term, which penalizes any example that was incorrectly filtered that would have been useful to the high-level algorithm.

$$C_M P(R_t \neg H) \tag{3.24}$$

Finally, the last term accrues a benefit for each of the examples that the high-level algorithm does not use that were filtered out by the visual utility filter.

$$- C_F[P(\neg R_t) - P(\neg R_t H)] \tag{3.25}$$

Putting these three components together, we see that using a visual utility filter generally involves a trade off between lower computation and lower false positives on one side and lower miss rate on the other.

3.4.1 Choosing a Visual Utility Filter

Looking at these results, we can describe the features of a visual utility estimator that will perform well by either decreasing the computational cost and/or increasing the precision of the overall system with minimal loss to recall. Let us start by examining one of the simplest visual utility estimators: one that samples candidate regions to process more sparsely than a given baseline B. Let's call this type of estimator C and let $P(H_C) = \gamma P(H_B)$ where γ is a sampling factor. In this scenario, different performance characteristics are achieved by changing how sparsely we sample. In this case, the computational cost of the visual utility filter is $D_C = 0$ because the sampling locations are fixed. Then, the change computational cost is:

$$\begin{aligned} & D_C - D_{R_t}(\tau(1) - \tau(P(H_C))) \\ = & - D_{R_t}(1 - \tau(\gamma P(H_B))). \end{aligned} \tag{3.26}$$

Which shows that the computational cost will always decrease and the size of that decrease is related to the number of examples that the high-level algorithm no longer needs to process. Next, look at the precision, which is equal to:

$$\text{Precision}_C = \frac{P(H_C R_t)}{P(H_C)}. \tag{3.27}$$

Now, because the sampling does not consider the content of the regions, we can say that H and R_t are independent. Therefore, the precision becomes:

$$\begin{aligned} \text{Precision}_C &= \frac{P(H_C)P(R_t)}{P(H_C)} \\ &= P(R) \\ &= \text{Precision}_B. \end{aligned} \tag{3.28}$$

Therefore, the precision will stay constant as the sampling becomes more sparse.

Next, let us look at the recall. If we consider the recall of the number of regions that can be used by the high-level algorithm, the recall is defined by:

$$\begin{aligned} \text{Recall}_C &= \frac{P(H_C R_t)}{P(R_t)} \\ &= \frac{P(H_C)P(R_t)}{P(R_t)} \\ &= P(H_C) \\ &= \gamma P(H_B). \end{aligned} \tag{3.29}$$

Therefore, the recall will drop at the same rate as the sampling is reduced.

However, there can often be another measure of recall that is relevant to many different tasks. In many situations, there will be a number of candidate regions that overlap the true object we are looking for and can be identified by the higher level algorithm. In this scenario, we can find the object of interest so long as at least one of the candidate regions are passed through the visual utility filter. Passing extra regions does not provide any benefit, but the particular region passed does not matter. So, in this scenario, let us define the fraction of target objects found to be the target recall. Now, for each of the objects $i \in O$, let $\bar{k}_i$ represent the number of candidate regions that would identify the object in the baseline scenario. Then, in the sampled scenario, the expected number of regions that are still processed is:

$$E_\gamma[k_i] = \gamma \bar{k}_i. \tag{3.30}$$

Now, if the sub-sampling is done uniformly, we know that if $E_\gamma[k_i] > 1$, the object will still be found. Therefore, the target recall becomes:

$$\text{Target Recall}_C = \frac{\sum_{i \in O} \min(1, \gamma \bar{k}_i)}{|O|}. \tag{3.31}$$

So, when selecting a visual utility filter, we are looking for one that:

1. Is fast to compute relative to the computational cost of the high-level algorithm.
2. Has a high recall for those elements that are used by the high-level algorithm.
3. Minimizes the number of examples passed to the high-level algorithm.

Furthermore, if the goal is to reach a given computational budget and the naive resampling approach reaches that budget at γ. Then, for the visual utility filter to be worthwhile, it must, when running at the same speed, have better or equal precision and, depending on the problem, either have a recall $\geq \gamma$ or a target recall $\geq \sum_{i \in O} \min(1, \gamma \bar{k}_i)/|O|$. Note that process using the content-based visual utility estimator must necessarily pass fewer candidate regions to the high-level algorithm than the naive approach. This can be seen by looking at eq. (3.26) and recognizing that in the naive approach, $D_{naive} = 0$, while for the visual utility estimator, $D_{content} \geq 0$. Therefore, to have the same overall computational cost, $P(H_{naive}) \geq P(H_{content})$.

3.4.2 Cascades (AND operation)

Now that we can measure the effectiveness of a visual utility filter, let us look at how we could construct such a filter using simple features. To do this, we will examine combining simple filters using the boolean AND operation.

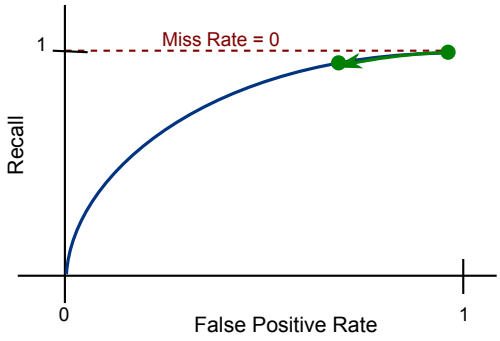


Figure 3.4: When training a cascade, a Neyman-Pearson test is used where the miss rate is fixed to a very small number β. However, the ROC curve around that point is typically very steep, so a small decrease in miss rate could result in a very large reduction of false positives, which would generate a better overall solution.

Cascades have been used by object detectors in order to speed up processing (e.g. [84, 233, 299, 334]). In a cascade, each candidate region of an image is passed through a sequence of weak classifiers. At any step, if the classifier fails the region, processing stops and the detection is considered failed. Only if the region passes successfully through all of the classifiers is it flagged as containing the desired object. This fail-early approach results in significantly less processing than evaluating each classifier for all candidate regions. Also, this process is equivalent to the boolean AND operation.

The typical method to create this cascades is to use a boosting technique from some training data. This technique is described in [299] and is based on the Neyman-Pearson test. A Neyman-Pearson test is a binary hypothesis test that instead of using costs, explicitly trades off the miss rate (P_M) and the false positive rate (P_F) by minimizing one while constraining the other. Formally, a Neyman-Pearson test δ_{NP} is one that solves one of the following two equations [175]:

$$\delta_{NP} = \arg\min_{\delta} P_M(\delta) | P_F(\delta) \leq \alpha, \tag{3.32}$$

$$\delta_{NP} = \arg\min_{\delta} P_F(\delta) | P_M(\delta) \leq \beta. \tag{3.33}$$

Then, choosing a particular α or β is equivalent to choosing an operating point on an ROC curve.

When training a cascade, we focus on eq. (3.33). For each level in the cascade, a very small β is chosen so that most positive examples in the training data are passed down the cascade and will be eventually evaluated. This implies that performance will not suffer significantly, but a speedup is still likely because there are usually many more negative regions in an image than positive ones. In the extreme case of [299], β is set to 0. However, this is a very conservative choice. In an ROC curve, the slope around $P_M = 0$ is typically very steep (fig. 3.4), which means that allowing a small number of misses could significantly reduce the number of false positives. Therefore, by being slightly less conservative, we could create a system with lower visual utility risk and thus better overall performance. This approach has been taken in some subsequent cascade classifiers [178, 334], however, β is manually set to some small constant number. Instead, we will show how to set the thresholds of the cascade levels in order to optimize visual utility risk directly.

In order to build a cascade to optimize visual utility risk, let us consider the step of adding the n+1 filter. This corresponds to the situation depicted in fig. 3.5 where the aggregate of the filters 1 to n are depicted as a single filter H_1 and we would like to determine what filter to insert as H_2 (if any). To do this we must first define the cost of using all n+1 filters, which is given by:

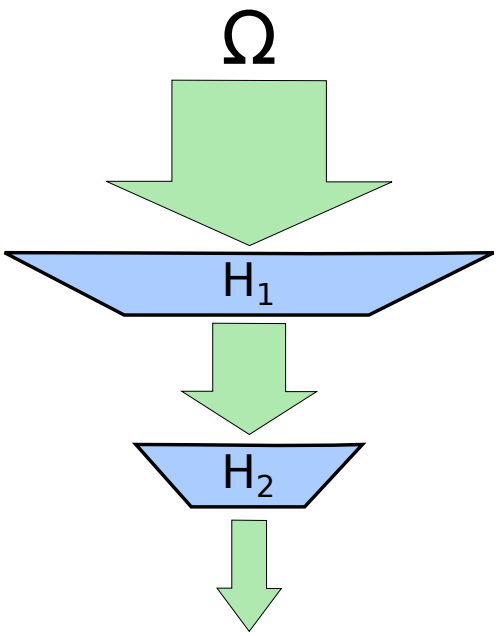


Figure 3.5: Combining two visual utility filters in an AND operation. After each filter, the number of regions to consider decreases, producing a potential speedup.

$$\begin{aligned}\nu(H_1H_2) &= D_1 + P(H_1)D_2 + \tau(P(H_1H_2))D_{R_t} + C_F P(\neg R_t H_1 H_2) + C_M P(R_t \neg(H_1 H_2)) \\ &= D_1 + P(H_1)D_2 + \tau(P(H_1H_2))D_{R_t} + C_F P(\neg R_t H_1 H_2) + \\ &\quad C_M[P(R_t \neg H_1) + P(R_t \neg H_2) - P(R_t \neg H_1 \neg H_2)]. \end{aligned} \tag{3.34}$$

Then, subtracting the risk from using n filters (eq. (3.7) with $H = H_1$), we can say that the filter to use for H_2 is the one that minimizes:

$$\begin{aligned}&\nu(H_1H_2) - \nu(H_1) \\ =&D_2P(H_1) + D_{R_t}\tau(P(H_1H_2)) - D_{R_t}\tau(P(H_1)) \\ &+ C_F[P(\neg R_t H_1 H_2) - P(\neg R_t H_1)] \\ &+ C_M[P(R_t \neg H_2) - P(R_t \neg H_1 \neg H_2)] \\ =&D_2P(H_1) - D_{R_t}[\tau(P(H_1)) - \tau(P(H_1H_2))] \\ &- C_F P(\neg R_t)[P(H_1|\neg R_t) - P(H_1H_2|\neg R_t)] \\ &+ C_M P(R_t)[P(\neg H_2|R_t) - P(\neg H_1 \neg H_2|R_t)]. \end{aligned} \tag{3.35}$$

If this term is less than 0, then it will be beneficial to insert filter H_2. Let us now examine at each of these terms individually, starting with the computational cost:

$$D_2P(H_1) - D_{R_t}[\tau(P(H_1)) - \tau(P(H_1H_2))]. \tag{3.36}$$

In this equation, $P(H_1)$ is the number of examples passed through H_1, while $\tau(P(H_1)) - \tau(P(H_1H_2))$ is related to the number of examples that were filtered out by H_2. We know that $P(H_1) \geq P(H_1) - P(H_1H_2)$ and therefore, to have any computational gain, $D_2 < D_{R_t}$. Furthermore, in order to reduce computation,

we want to choose an H_2 that filters out as many examples as possible. Now, let us look at the false positive term:

$$- C_F P(\neg R_t)[P(H_1|\neg R_t) - P(H_1 H_2|\neg R_t)]. \tag{3.37}$$

This term says that we will accrue a benefit for every negative example that was incorrectly let through H_1, but is filtered by H_2. Therefore, to calculate this term, we only need to examine the negative training cases that get through H_1. Also, this term is proportional to the $P(\neg R_t)$, which will be high relative to $P(R_t)$ in most image applications because there is often only a small portion of the image/scale space that contains an area of interest. Finally, let us look at the miss term:

$$C_M P(R_t)[P(\neg H_2|R_t) - P(\neg H_1 \neg H_2|R_t) \tag{3.38}$$

This term says that we will be penalized by all positive training examples that are incorrectly filtered out by H_2 and were passed by H_1. Taken together, we can notice three things about these three terms. First, only those training examples that are passed by H_1 are needed to calculate this term and thus we can ignore all those examples already filtered out when trying to find the best weak classifier. Second, when searching for the next classifier to add, we could search through many classes of features, like motion, shape, texture and so on. Within each of these classes, parameters can be adjusted to change the resulting $P(H_2)$ terms. Finally, as more weak classifiers are added the impact of any additional classifier begins to decrease. This property is known as submodularity and we will next examine how it applies to this formulation.

3.4.3 Submodularity in Cascades

A submodular function is defined over a set $A \subseteq V$ and has the property of diminishing returns, where adding a new element to A in the future can never have a larger effect than adding it now. Formally, let F be a function defined on $2^V \rightarrow \mathbb{R}$. Then, the function F is submodular if:

$$F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B) \forall A \subseteq B \subseteq V, x \notin B, \tag{3.39}$$

or equivalently

$$F(A) + F(B) \geq F(A \cup B) + F(A \cap B) \forall A \subseteq V, B \subseteq V. \tag{3.40}$$

Submodular functions are interesting because they are analogous to convex functions, but defined for discrete sets. Similar to convex functions, submodular functions can be maximized effectively using the greedy algorithm and thus can be optimized quickly. In particular, even though maximizing a submodular problem is NP-Hard, [213] showed that for non-decreasing, positive submodular functions, the greedy optimization gives a result that is near optimal because it has a worst case lower bound relative to the optimal solution:

$$F(A_{greedy}) \geq (1 - 1/e) \max_{A \subseteq V} F(A). \tag{3.41}$$

We can also identify "how close" a function is to submodular using the submodularity ratio introduced in [62]. A function F has a submodularity ratio $\gamma_{U,k}$ if, for the set U and parameter $k \geq 1$:

$$\sum_{x \in S} [F(L \cup \{x\}) - F(L)] \geq \gamma_{U,k}[F(L \cup S) - F(L)], \forall L \subseteq U, S : |S| \leq k, S \cap L = \emptyset \tag{3.42}$$

Now, using the minimum submodularity ratio γ over all possible U and k, [62] showed that maximizing F using the greedy algorithm is bounded by:

$$F(A_{greedy}) \geq (1 - e^{-\gamma}) \max_{A \subseteq V} F(A). \tag{3.43}$$

We will also use two other properties of submodular functions in our discussion. First, submodular functions are closed under non-negative linear combinations. In other words, if functions $F_1, \ldots, F_n$ are submodular and $\lambda_0, \ldots, \lambda_n > 0$, then

$$F(A) = \sum_i \lambda_i F_i(A) \tag{3.44}$$

is also submodular. Second, a function G where $-G$ is submodular is called *supermodular* and is analogous to a concave function. Taken together, these properties imply that greedily minimizing a supermodular function is also near optimal.

Now, let us look at the problem of finding a set of weak visual utility filters $H_0, \ldots, H_n$ to combine in a cascade to minimize the visual utility risk. Let V be the set of all possible weak classifiers. Then, to minimize the visual utility risk, let $A \subseteq V$, then we can define $F(A)$ that we want to minimize as:

$$\begin{aligned} F(A) &= \nu(A) \\ &= D_A + D_{R_t} P(A) + C_F P(\neg R_t A) + C_M P(R_t \neg A). \end{aligned} \tag{3.45}$$

Note that D_A cannot be computed directly for an arbitrary set because it implicitly relies on an ordering of the set A that is used to define the cascade. Let $\Pi = \{\pi_0, \pi_1, \ldots, \pi_n\}$ be a permutation of the set A, then:

$$\begin{aligned} D_A =& D_{\pi_0} + D_{\pi_1} P(\pi_0) + \cdots + D_{\pi_n} P(\pi_0 \pi_1 \ldots \pi_{n-1}) \\ =& \sum_{i=0}^{n} D_{\pi_i} P(\bigcap_{j=0}^{i-1} \pi_j). \end{aligned} \tag{3.46}$$

Then, using the results from eq. (3.35), we can see that:

$$\begin{aligned} F(A \cup \{H_i\}) - F(A) =& D_i P(A) - D_{R_t}[P(A) - P(AH_i)] && \text{(3.47a)} \\ & - C_F P(\neg R_t)[P(A|\neg R_t) - P(AH_i|\neg R_t)] && \text{(3.47b)} \\ & + C_M P(R_t)[P(\neg H_i|R_t) - P(\neg A \neg H_i|R_t)]. && \text{(3.47c)} \end{aligned}$$

We will now look at each component of this equation separately and evaluate them for submodularity. In each case, $A \subseteq B \subseteq V$ and therefore $P(AY) \geq P(BY)$ where Y is any set. Starting with the computational cost (eq. (3.47a)) D_{R_t} term:

$$\begin{aligned} -D_{R_t}[P(A) - P(AH_i)] &\overset{?}{\geq} -D_{R_t}[P(B) - P(BH_i)] \\ -P(A\neg H_i) &\leq -P(B\neg H_i). \end{aligned} \tag{3.48}$$

Therefore, the D_{R_t} term is supermodular. Next, we will look at the D_i term of the computational cost:

$$\begin{aligned} D_i P(A) &\overset{?}{\geq} D_i P(B) \\ D_i P(A) &\geq D_i P(B) \end{aligned} \tag{3.49}$$

Therefore, the D_i term is submodular irrespective of the ordering associated with it. Also, we should note that $D_R \gg D_i$ in most cases, so the D_R term will dominate as long as H_i continues to filter out a significant number of examples.

Next, let us look at the false positive term (eq. (3.47b)):

$$\begin{aligned} -C_F P(\neg R_t)[P(A|\neg R_t) - P(AH_i|\neg R_t)] &\overset{?}{\geq} -C_F P(\neg R_t)[P(B|\neg R_t) - P(BH_i|\neg R_t)] \\ -[P(A\neg R_t) - P(H_i A\neg R_t)] &\overset{?}{\geq} -[P(B\neg R_t) - P(H_i B\neg R_t)] \\ -P(A\neg R_t\neg H_i) &\leq -P(B\neg R_t\neg H_i). \end{aligned} \tag{3.50}$$

Therefore, the false positive term is not submodular and is in fact supermodular. Finally, let us look at the miss term (eq. (3.47c)):

$$\begin{aligned} C_M P(R_t)[P(\neg H_i|R_t) - P(\neg A\neg H_i|R_t)] &\overset{?}{\geq} C_M P(R_t)[P(\neg H_i|R_t) - P(\neg B\neg H_i|R_t)] \\ P(\neg H_i R_t) - P(\neg A\neg H_i R_t) &\overset{?}{\geq} P(\neg H_i R_t) - P(\neg B\neg H_i R_t) \\ P(A\neg H_i R_t) &\geq P(B\neg H_i R_t). \end{aligned} \tag{3.51}$$

Therefore, the miss term is submodular. Combining all of these terms, we have a utility function to minimize of the form:

$$F(A) = G(A) + J(A). \tag{3.52}$$

where $G(A)$ is submodular and equal to:

$$G(A) = D_A + C_M P(R_t\neg A), \tag{3.53}$$

while $J(A)$ is supermodular and equal to:

$$J(A) = D_{R_t}\tau(P(A)) + C_F P(\neg R_t A). \tag{3.54}$$

This result helps us to explain why the traditional method of creating a cascade using a Neyman-Pearson test is so effective. The traditional technique, fixes the miss rate and then any computational advantage is a side effect that is not optimized for directly. This approach is equivalent to greedily minimizing the $C_F P(\neg R_t A)$ term under a constraint of a low miss rate. This term is supermodular and so the minimization, is equivalent to maximizing a submodular function. We know that in the worst case, this maximization is guaranteed to be nearly optimal (eq. (3.41)) and thus we are guaranteed to have near-optimal false positive rate for a fixed miss rate. This is explored in more detail for the Viola-Jones Felzenswalb star cascades in chapter 4.

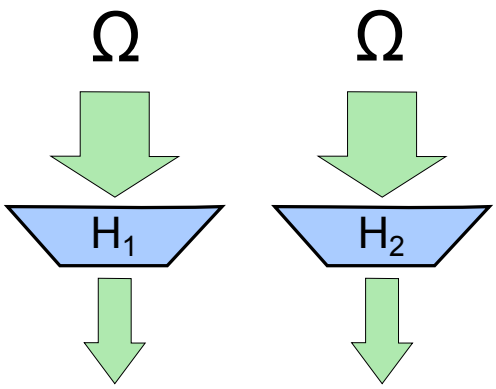


Figure 3.6: Combining two visual utility filters in an OR operation. If a location passes either of the filters, it is sent to the high-level algorithm.

3.4.4 Parallel Filters (OR operation)

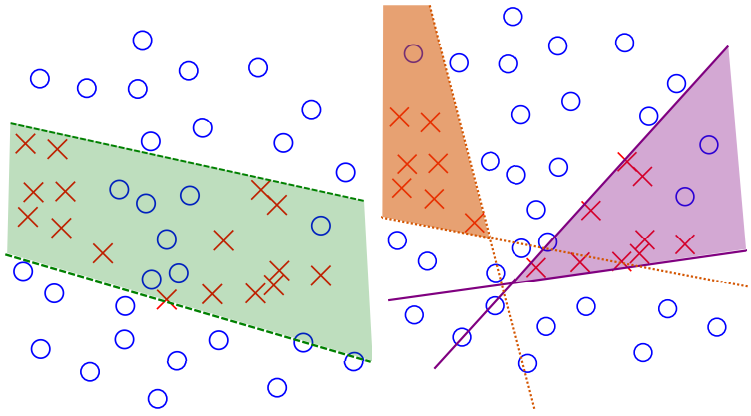
In the previous section, we talked about the benefits of using cascaded filters. However, for each level of the cascade added, the miss rate goes down because there are examples in the training set that are incorrectly filtered. We can avoid this problem by setting up the filter so that all examples pass through, but at that point on the ROC curve, we are trading off a large number of false positives for a small change in miss rate. For some problems, there is another approach: parallel filters. Just as a cascade is equivalent to a boolean AND operation, parallel filters are equivalent to an OR operation. Parallel filters are depicted in fig. 3.6 and are combined so that a region is processed by the high-level algorithm if it passes through either of the filters. As we will see, the operation is symmetric to adding a level to the cascade, except that this time, we are decreasing the miss rate and increasing the false positives.

Now, let us consider the operation of adding the $n+1$ filter. This corresponds to the situation depicted in fig. 3.6 where the aggregate of the filters 1 to n are depicted as a single filter H_1 and we would like to determine what filter to insert as H_2 (if any). To do this, we first define the cost of using all $n+1$ filters, which is given by:

$$\begin{aligned}\nu(H_1H_2) &= D_1 + D_2 + \tau(P(H_1 \vee H_2))D_{R_t} + C_F P(\neg R_t(H_1 \vee H_2)) + C_M P(R_t \neg(H_1 \vee H_2)) \\ &= D_1 + D_2 + D_{R_t}\tau[P(H_1) + P(H_2) - P(H_1H_2)] \\ &\quad + C_F[P(\neg R_t H_1) + P(\neg R_t H_2) - P(\neg R_t H_1 H_2)] \\ &\quad + C_M P(R_t \neg H_1 \neg H_2). \end{aligned} \tag{3.55}$$

Then, subtracting the risk from using n filters (eq. (3.7) with $H = H_1$), we can say that the filter to use for H_2 is the one that minimizes:

$$\begin{aligned}&\nu(H_1H_2) - \nu(H_1) \\ =&D_2 + D_{R_t}\tau[P(H_2) - P(H_1H_2)] \\ &+ C_F[P(\neg R_t H_2) - P(\neg R_t H_1 H_2)] \\ &+ C_M[P(R_t \neg H_1 \neg H_2) - P(R_t \neg H_1)] \\ =&D_2 + D_R\tau[P(H_2) - P(H_1H_2)] && \text{(3.56a)} \\ &+ C_F P(\neg R_t)[P(H_2|\neg R_t) - P(H_1H_2|\neg R_t)] && \text{(3.56b)} \\ &- C_M P(R_t)[P(\neg H_1|R_t) - P(\neg H_1 \neg H_2|R_t)] && \text{(3.56c)}\end{aligned}$$



(a) Single Cascade (b) Parallel Cascades

Figure 3.7: A toy example of where parallel cascades are more expressive and might be more efficient because positive examples form relatively distinct clusters. In a single cascade (left), the decision boundary must specify a single closed space in a multi-dimensional feature space. As a result, any cascade that passes all of the positive examples, must also pass the negative examples in between. However, with parallel cascades, tighter decision boundaries can be specified around different clusters, so that fewer negative examples are passed. Furthermore, because of the tighter boundaries, examples are filtered more quickly and thus each individual cascade can be faster than a single cascade for all examples.

Furthermore, for the inclusion of H_2 to be useful, the equation must be < 0. Let us now look at each of these components in turn. First, we examine the computational component (eq. (3.56a)). In this case, we are guaranteed to increase the computational cost by the cost of computing H_2 and number of extra examples that must be processed by the higher level algorithm.

Next, looking at the false positive term (eq. (3.56b)), we see that we will accrue a penalty for each negative example that is filtered by H_1, but missed by H_2. Note that to calculate this term, we only need to examine those examples that were filtered by H_1.

Finally, the miss term (eq. (3.56c)) says that we will accrue a benefit for each positive example that was incorrectly filtered by H_1, but is passed by H_2. To calculate this term, we only need to examine positive examples that H_1 filtered.

Taken together, these terms show that adding a visual utility filter in parallel will decrease the miss rate, but at the cost of more computation and higher false positive rate. Furthermore, the best filter is the one that focuses on passing those positive examples that are distinct from those passed by H_1. From these observations, we can see that using parallel filters would be most effective in situations where there are distinct clusters of positive examples that can be isolated quickly. An toy example of this situation is shown in fig. 3.7.

In a practical example, consider the problem of pedestrian detection in mining. In this scenario, people can be walking or standing still. Often, when people are moving, their shape changes significantly to the camera because the legs and arms are moving relative to the body. Thus, a detector that can find people that are both moving and standing still, must be able to handle the change in morphology, which will make it slower to compute. On the other hand, we could use a motion detector to quickly find regions containing moving people and then a simpler shape detector that focuses on detecting people standing still. The resulting combined filter could then be faster and more accurate.

3.5 Adaptive Submodularity

In the previous sections, we have discussed filtering an entire image to identify the set of regions that contain useful information. Another approach is to be adaptive. In this setting, regions within the image are sequentially sampled. After each sample, the expected utility of examining each unsampled region is updated and a new sample is chosen. For submodular problems, this has been shown to reduce the expected cost by $\Omega(n/log(n))$ relative to the non-adaptive approach [99].

Now, let us examine a the submodular properties of a typical approach for using visual utility estimators in an adaptive setting. In order to do this, we must first introduce *adaptive submodularity* from [99].

In this formulation, let E be the set of all locations to sample. Then, let O be the set of all possible states when evaluating the high-level algorithm. Next, let $\phi : E \to O$ be known as a realization so that $\phi(e)$ is the result of an observation at location e. So, we will sequentially choose a location e, get the observation at $\phi(e)$ and then pick the next location. Building up a sequence of observations creates a partial realization ψ that is defined on $\psi \subset E \times O$ equal to $(e, o) : \phi(e) = o$. Partial realizations are equivalent to the current belief state of the model. Then, the dominion of ψ, or the set of items observed in ψ can be defined by $dom(\psi) = e : \exists(e, o) \in \psi$. Furthermore, we can consider ϕ probabilistically so that $p(\phi)$ is the prior belief and $p(\phi|\psi)$ is the posterior belief of all locations e that are not selected yet. Finally, we can define a policy π that is a function to select the next location to sample. Formally, $\pi : \Psi \to E$.

Next, we can examine problems where we want to maximize a utility function $f : 2^E \times O^E \to \mathbb{R}_{\geq 0}$ subject to cost constraints. Let us define the Conditional Expected Marginal Benefit of e conditioned on observing ψ as:

$$\Delta(e|\psi) := \mathbb{E}[f(dom(\psi) \cup \{e\}, \psi) - f(dom(\psi), \psi)]. \tag{3.57}$$

Then, a function f is *adaptive monotone* if the benefit of selecting e at any point is always non-negative:

$$\Delta(e|\psi) \geq 0 \; \forall e \in E, \psi \in \{p(\psi) > 0\}. \tag{3.58}$$

Furthermore, f is *adaptive submodular* if the benefit of selecting e now will always be greater than or equal to the benefit of selection later:

$$\Delta(e|\psi) \geq \delta(e|\psi') \; \forall e \in \{E \setminus dom(\psi')\}, \psi \subseteq \psi'. \tag{3.59}$$

Next, let us define the adaptive greedy policy, which is defined as the one that at each step selects

$$e^* = \underset{e \in \{E \setminus dom(\psi)\}}{\arg\max} \left(\frac{\Delta(e|\psi)}{c(e)} \right), \tag{3.60}$$

where $c(e)$ is the cost of selecting e.

If f is both *adaptive monotone* and *adaptive submodular*, then the adaptive greedy algorithm will be near optimal for a given cost constraint:

$$f_{avg}(\pi_{greedy}) > (1 - e^{-1}) f_{avg}(\pi_{optimal}). \tag{3.61}$$

3.5.1 Visual Utility and Adaptive Submodularity

To examine an adaptive submodular approach that incorporates visual utility, let us consider the high-level algorithm for object detection where each region $r \in R_t$ in the image could contain the object of interest.

Furthermore, we consider the object to be found in r if the standard IOU criterion is met with respect to the true objects bounding box g:

$$IOU_{r,g} = \frac{|r \cap g|}{|r \cup g|} > 0.5. \tag{3.62}$$

Next, we can see that the visual utility estimator produces $p(\phi)$, which gives us the prior probability distribution for finding an object. This leads us to a defining $p(\phi|\psi)$ to be the probability of finding an undiscovered object after examining a number of regions with the high-level detector.

In this scenario, we can use the utility function defined as the probability mass that has been examined thus far. So, at step n, we examine the region r_n with the highest probability of finding an undiscovered object. Then, if an object is found, capture the mass probability by setting the posterior probability to 0 for all regions that meet the IOU criterion:

$$p(\phi|\psi_n) = \begin{cases} 0 & IOU_{r_n,\phi} > 0.5 \\ p(\phi|\psi_{n-1}) & \text{otherwise.} \end{cases} \tag{3.63}$$

If an object is not found, reduce the posterior probability for all regions by a function of their to IOU criterion to encode the belief that because an object was not found, the likelihood of nearby objects decreases. This can be done using a linear relationship:

$$p(\phi|\psi_n) = (1 - IOU_{r_n,\phi})p(\phi|\psi_{n-1}). \tag{3.64}$$

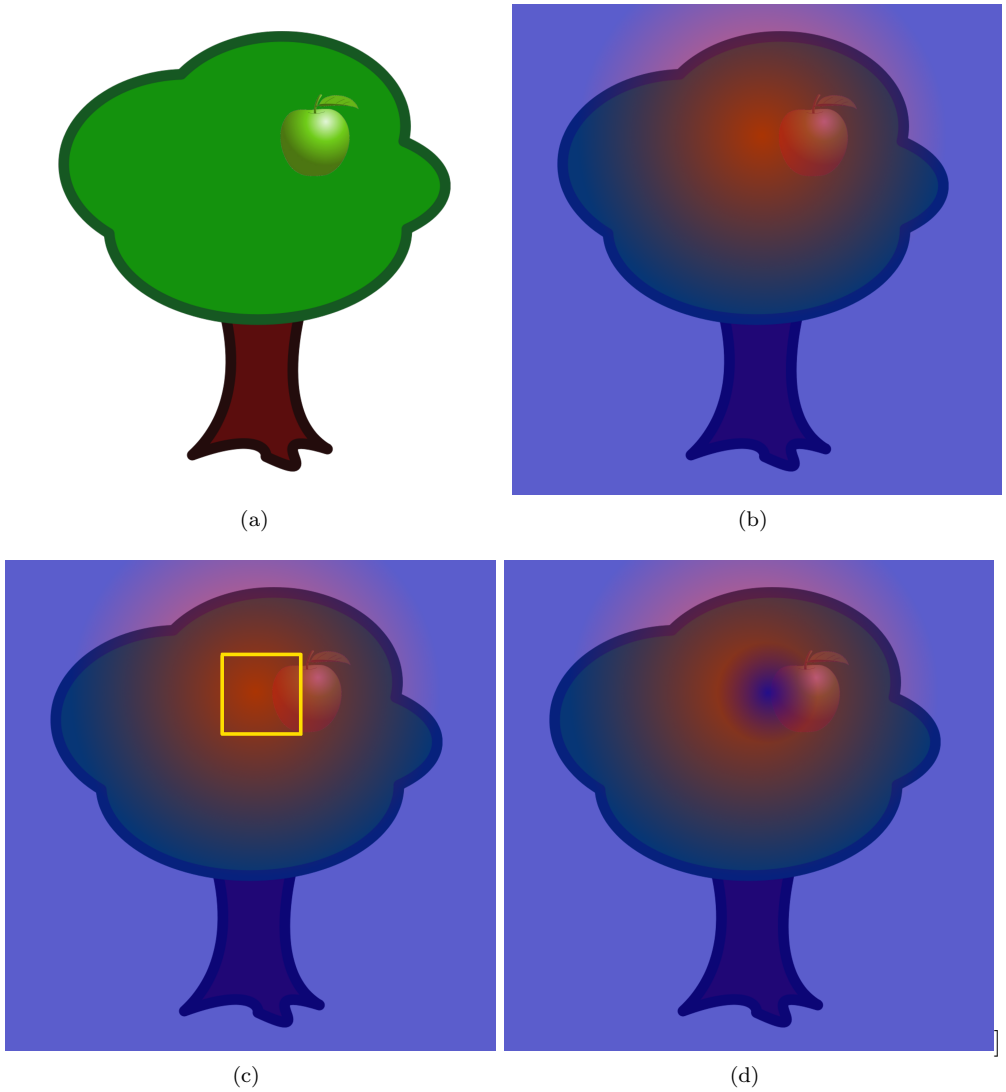
This approach yields an anytime algorithm because steps can be taken until either a computation budget is exhausted or until the probability of all regions equals to zero. This formulation is equivalent to the Adaptive Stochastic Min-Sum Cover problem from [100], which they show is adaptive submodular and thus near optimal. Furthermore, [100] show that the adaptive case will examine $\Omega(log(|E|))$ regions, while the non-adaptive approach will examine $\Omega(|E|)$. Therefore, there is a potential for significant computational savings.

Now, just as in the non-adaptive case, the choice of visual utility estimator is an important consideration. First, it must be aligned with the high-level detector so that the local maxima of the estimator coincide with the local maxima of the high-level detector. Otherwise, we could examine regions that are close to true objects but do not detect them with the high-level detector. This will cause the regions that will fire in the high-level detector to lose posterior probability mass, making them less likely to be picked (e.g. fig. 3.8). The impact of this effect is shown in an experiment in section 7.3.4.

Next, the visual utility estimator should be fast to compute. In particular, we can use the previous approach without a visual utility estimator simply by choosing a uniform prior over all the regions in R_t. To have less of a computational cost than this simple baseline, the visual utility estimator must be such that

$$\begin{aligned} D_{VU} + (D_S + D_R)\,\mathbb{E}\left[|\psi|\middle|\phi_{VU}\right] &< (D_S + D_R)\mathbb{E}\left[|\psi|\middle|\phi_{uniform}\right] \\ D_{VU} &< (D_S + D_R)\left(\mathbb{E}\left[|\psi|\middle|\phi_{uniform}\right] - \mathbb{E}\left[|\psi|\middle|\phi_{VU}\right]\right). \end{aligned} \tag{3.65}$$

where:
D_{VU} Is the computational cost of computing the visual utility estimator.
D_S Is the computational cost of selecting the next region to examine greedily.
D_R Is the computational cost of computing the high-level detector on one region.



(a) (b) (c) (d)

Figure 3.8: When performing an adaptive search, the alignment of the visual utility estimator with the high-level algorithm is very important. For example, in this toy sequence, we could be looking for apples (a) and using a high-level algorithm detector that is perfect at finding the apples. Figure (b) shows the output of the visual utility estimator that is not aligned, so the first region is examined (c). If the region selected does not result in a detection with the high-level detector, then the probability of returning to the area decreases (d).

Therefore, in the adaptive case, just like in the non-adaptive case, it is important to select a visual utility estimator that is both efficient to compute and accurate at predicting identifying the regions that are useful to the high-level algorithm.

3.6 Summary

In this chapter, he have defined visual utility for a general problem and presented some scenarios where its use could be of value. In particular, using visual utility can decrease the number of false positives and the computational requirements of an algorithm at the cost of missing some positive examples. We formalized this trade-off by minimizing a cost composed of the Bayes risk and the computational requirements in the non-adaptive case. This minimization problem is then analyzed using submodularity in order to identify techniques for combining simple visual utility filters into a more effective aggregate filter. From this analysis, we have identified algorithms for building a visual utility filter when the cost is constrained and when trying to balance computational cost and detection performance directly. Finally, we have examined the adaptive case that uses a visual utility estimator to iteratively process the useful regions in the image. We then show that it is possible to define an adaptive submodular algorithm that is both an anytime algorithm and near optimal.

Chapter 4

Theoretical Case Studies

> Time is the only critic without personal ambition.
>
> John Steinbeck

In this section, we will step through two well used algorithms from the literature and analyze them using the visual utility framework.

4.1 Viola-Jones Cascade Detector

In this section, we will use the visual utility framework in this thesis to examine the Viola-Jones cascaded face detector from [299]. First, we will detail the detector as it was presented in the original paper. Next, we will describe the detector in terms of the visual utility framework. Finally, we will use our theoretical analysis of the visual utility framework to determine the performance guarantees implicit in the detector.

4.1.1 Details of the Viola-Jones Cascade Detector

The Viola-Jones detector is a cascade of very simple features that are reminiscent of Haar basis functions. In this framework, a set of candidate bounding boxes are evaluated for whether or not they contain the object of interest. The features used for this classification are composed of two, three or four rectangles (fig. 4.1), and their value is simply the sum of the pixel intensity in the white rectangle(s) subtracted from the sum of the pixel intensity in the black rectangle(s). Each potential feature is defined by its height, width and location relative to the bounding box being evaluated. These features were chosen because they can be calculated very quickly at any scale or location using an integral image. An integral image is one for which the pixel at location x, y is equal to the sum of pixels above and to the left of x,y. More formally, an integral image is defined by:

$$II(x,y) = \sum_{x' \leq x, y' \leq y} I(x', y') \tag{4.1}$$

where $I(x,y)$ is the original image and $II(x,y)$ is the integral image. Now, using the integral image, we can see that to calculate the sum of all the pixels in a window with the upper left corner at (x,y) and a height h and width w we need to only access at the values of the integral image in the four corners of the window. Specifically:

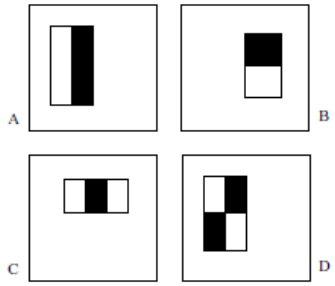


Figure 4.1: Example Haar like features used in the Viola-Jones cascade. The sum of the pixels in the white rectangles are subtracted from the sum of the pixels in the black rectangle. Examples are shown for two-rectangle (A and B), three-rectangle (C) and four-rectangle (D) features. Figure from [299].

$$\sum_{\substack{x<x'\leq x+w \\ y<y'\leq y+h}} I(x',y') = II(x+w,y+h) + II(x,y) - II(x+w,y) - II(x,y+h) \tag{4.2}$$

Furthermore, since the rectangles in each feature are adjacent, the two-rectangle features can be computed using six array lookups, while the three rectangle features take eight lookups and the four-rectangle feature takes nine lookups.

There are a large number of potential features that can be used for detection (180,000 for a detector of base resolution 24x24). Features are selected from this set using AdaBoost where the weak classifiers are decision stumps of the rectangle features. Specifically a weak classifier $h_j(x)$ is defined by:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where $f_j(x)$ is a feature evaluated on a window, θ_j is a threshold and p_j is a parity function to indicate the direction of the equality sign. The final strong classifier is then a weighted sum of the weak classifiers defined by:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq 0.5 \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where α_t are weights learned during the AdaBoost process and T are the number of features.

In order to create a faster classifier, a group of these strong classifiers are put into a cascade. A cascade is a degenerate decision tree where a window is rejected if it fails any of the classifiers in the cascade. Only if a window passes all stages in the cascade is it accepted (fig. 4.2). Therefore, as soon as an example fails a given cascade level, processing can be stopped. This also means that when learned a strong classifier for a stage in the cascade, we only need to consider those training examples that were not already filtered by the previous stages in the cascade.

At each stage i of the cascade, a classifier is trained using a modified AdaBoost algorithm that ensures that the miss rate of the stage is $\leq m$ and the false positive rate is $\leq \kappa$. The classifier is trained only with those examples that have successfully passed all previous stages.

To ensure these guarantees, AdaBoost is run except that after each feature selection, the overall threshold is adjusted until the miss rate is below m_i. Then, the false positive rate is measured. If it is $> \kappa$, the AdaBoost process continues and another feature is selected. If the false positive rate is $\leq \kappa$, the classifier is considered trained and added to the cascade as a new stage with the adjusted threshold.

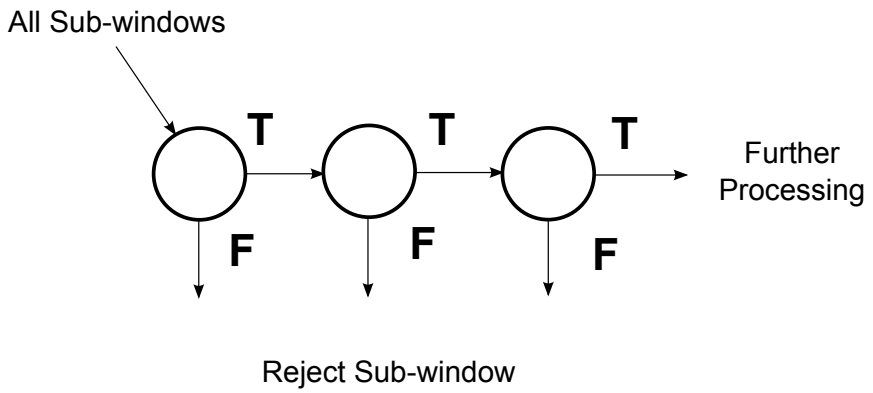


Figure 4.2: A generic object detection cascade. At each stage of the cascade, a classifier is applied to a sub-widow. If the window fails the classification, it is rejected and no further processing is done. Figure from [299]

4.1.2 Mapping to the Visual Utility Framework

Let us now examine the Viola-Jones cascade with respect to the generic visual utility framework. To start, the high-level algorithm in this case is simply a binary function that is true if the example passes through the entire cascade and false otherwise. This means that the computational cost of the high-level algorithm is 0 and the D_{R_t} term from eq. (3.35) is also 0.

Next, let us look at the false positive and miss penalties (the C_F and C_M terms). When building stage i of the cascade, the filter is built so as to have a miss rate $\leq m$ and false positive rate $\leq \kappa$. In a large training set, these bounds will be very tight. Therefore, for these penalties, we can derive a tighter bound than can be found using submodularity. In particular, for an N stage cascade, the false positive cost will be $\leq C_F P(\neg R)\kappa^N$ and the miss cost will be $\leq C_M P(R)(1-(1-m)^N)$.

Finally, the $D(A)$ term is directly related to the processing time of the boosted classifier at each stage.

4.1.3 Theoretical Analysis

The only term in $F(A)$ that we must now consider is the D_A term, which we have already shown is submodular in section 3.4.3.

Finally, to show that the Viola-Jones algorithm is near optimal given the constraints of the formulation, we must show that it is equivalent to the greedy algorithm on the constrained $F(A)$. Since the only non-constant term in $F(A)$ is the D_A term, this is equivalent to saying that the modified AdaBoost process finds the filter that uses the least number of features that meets the constraints on the false positive and miss rates.

Though AdaBoost is not guaranteed to find the filter using the least number of features, it is guaranteed to be near optimal. This can be shown by recognizing that AdaBoost is greedily minimizing an exponential loss function:

$$\mathcal{R}[f] = \sum_{n=1}^{N} e^{-y_n f(x_n)}. \tag{4.5}$$

Grubb and Bagnell [105] show that such a loss function on a set of weak predictors can be bounded by examining the submodularity ratio (eq. (3.42)). In particular, if $\mathcal{R}$ is λ-strongly convex $\forall f, f'$:

$$\mathcal{R}[f'] \geq \mathcal{R}[f] + \langle \Delta\mathcal{R}[f], f' - f \rangle + \frac{\lambda}{2}||f' - f||^2, \tag{4.6}$$

for some $\lambda > 0$ and Λ-strongly smooth:

$$\mathcal{R}[f'] \leq \mathcal{R}[f] + \langle \Delta\mathcal{R}[f], f' - f \rangle + \frac{\Lambda}{2}||f' - f||^2, \tag{4.7}$$

for some $\Lambda > 0$, then the submodularity ratio $\gamma \geq \frac{\lambda}{\Lambda}\lambda_{\min}(C)$ where $\lambda_{\min}(C)$ is the minimum eigenvalue of the covariance matrix for the weak predictors.

Now if we convert the problem of minimizing the loss function $\mathcal{R}[f]$ to the one of maximizing the submodular function:

$$\max_f \mathcal{R}[f] - \mathcal{R}[f] \tag{4.8}$$

then, using Theorem 5.6 from [105], at any stage, AdaBoost will find a solution such that

$$\mathcal{R}[f_{\text{ada}}] < \min_f \mathcal{R}[f] + e^{-\gamma}(\max_f \mathcal{R}[f] - \min_f \mathcal{R}[f]). \tag{4.9}$$

It has been shown in [204] that AdaBoost is guaranteed converge to within ϵ of the optimal solution after $O(ln(1/\epsilon))$ boosting iterations. Therefore, for stage i, the number of weak classifiers required to reach a given bound can be bounded and D_{A_i} can be given by

$$D_{A_i} < O(\gamma \ln(\max_{f_i} \mathcal{R}[f_i] - \min_{f_i} \mathcal{R}[f_i])) \tag{4.10}$$

Therefore, the Viola-Jones algorithm is guaranteed to limit the visual utility risk for an N stage cascade to:

$$\begin{aligned} F(A) <& C_F P(\neg R)\kappa^N + C_M P(R)(1 - (1 - m)^N) + \\ & \sum_{i=1}^{N} O(\gamma \ln(\max_{f_i} \mathcal{R}[f_i] - \min_{f_i} \mathcal{R}[f_i])) \end{aligned} \tag{4.11}$$

4.1.4 Discussion

This exercise shows how to describe the Viola-Jones cascade using the visual utility framework. The bound detailed in eq. (4.11) cannot be used in practice for two reasons. First, it relies on the maximum and minimum values of $\mathcal{R}$, which in general cannot be found. Second, the bound is only specified up to an order of computational complexity (eq. (4.10)).

Though the derived bound by itself is not of practical use, we can examine the assumptions used to derive it in order to identify areas of improvement for cascade algorithms. First, when building each stage, Viola-jones limits the filter to one that satisfies a miss rate $\leq m$ and a false positive rate $le\kappa$. However, it is not guaranteed that such a filter will also minimize the visual utility risk once computation time is considered. In other words, the Viola-Jones cascade achieves a computational speedup as a by-product of the construction process and is not guaranteed.

The second assumption is that AdaBoost will find a composite classifier that nearly minimizes the number of features used. AdaBoost is necessary because the features in the Viola-Jones cascade are so

simple that a simple decision stump would not be discriminative enough. A different approach would be to use a more limited set of slightly more complex features as candidates in the cascade. This more limited set could be examined exhaustively to guarantee that the best one is found.

Lastly, Viola-Jones uses an ad-hoc method based on a Neyman-Pearson test to deal with the asymmetry of the scenario; there will be many more negative than positive cases in a given image. In this case, a round of boosting is done and then the threshold of the resulting classifier is adjusted to meet the Neyman-Pearson criteria for miss rate. A better approach would be to handle the asymmetry directly in the learning process either by minimizing visual utility risk directly, or using other approaches like those described in [197, 297, 316].

4.2 Star Cascade for Deformable Parts Models

In this section, we will examine the star cascade from [84] using our visual utility framework and show that the construction of the cascade is a submodular process and is thus guaranteed to be near optimal.

4.2.1 Details of the Star Cascade

The star cascade operates on the deformable parts model detector from [83]. The deformable parts model is defined in the HOG feature space. The model consists of a root filter, that operates at a lower resolution, and a set of part filters that operate at a higher resolution. Each part filter covers a sub region of the model and has an ideal location relative to the placement of the root filter. For example, in the model of a human, the root filter may correspond to the rough shape of the person, while the parts correspond to the arms, legs, head and torso. However, we should note that the parts are found latently from the data and thus, the previous example is for only to illustrate the concept. In reality, the parts may not correspond to features that humans would identify.

Formally, using the notation from [84], let Ω be the space of locations within an image. For example, this space could be all locations and scales. Then, a model of an object consisting of n parts is defined by a root filter F_0 and a set of parts $(P_1, \ldots, P_n)$ where $P_i = (F_i, a_i)$. In this case, F_i is the filter for the i-th part, a_i specifies the ideal location of the part relative to the root. Now, for $\omega \in \Omega$, let $m_i(\omega)$ be the score of placing P_i in location ω. In the case of HOG filters, $m_i(\omega)$ is the filter response with the image. Now, let Δ be a space of displacements relative to the root filter and let $\oplus : \Omega \times \Delta \to \Omega$ be a binary operator taking a root location and a displacement to find the displaced location. Finally, let $d_i(\delta)$ specify the deformation cost for a displacement δ of P_i from its ideal location location a_i. In the HOG-based deformable parts model, $d_i(\delta)$ is defined as:

$$d_i(\delta) = b_i \cdot (\tilde{x}_i, \tilde{y}_i) + c_i \cdot (\tilde{x}_i^2, \tilde{y}_i^2) \tag{4.12}$$

where:

$$(\tilde{x}_i, \tilde{y}_i) = \frac{(x_i, y_i) - 2(x, y) + v_i}{s_i} \tag{4.13}$$

where v_i is a 2D vector specifying the ideal spatial displacement of the part, s_i is a 2D vector specifying the scale of the part, while b_i and c_i are 2D vectors specifying the coefficients of a quadratic function.

Now, the overall score of a model at a given location with parts displacements is given by:

$$\text{score}(\omega, \delta_1, \ldots, \delta_n) = m_0(\omega) + \sum_{i=1}^{n} m_i(a_i(\omega) \oplus \delta_i) - d_i(\delta_i) \tag{4.14}$$

Then, to find the score of a given location, we must search through all the displacements of the parts to find the maximum. In other words,

$$\text{score}(\omega) = m_0(\omega) + \sum_{i=1}^{n} \max_{\delta_j \in \Delta} (m_i(a_i(\omega) \oplus \delta_j) - d_i(\delta_j)) \tag{4.15}$$

In the original algorithm, presented in [83], this search was done exhaustively, but using a distance transform in order to avoid recalculating the filter responses m_i. In order to speed up this approach, Felzenswalb et al. present a star cascade which prunes this search in two ways: hypothesis pruning and deformation pruning [84]. Both pruning approaches take advantage of the fact that $\text{score}(\omega) > T$ for an example to be accepted. If we examine equation eq. (4.15), we see that the evaluation for each part is independent of the other parts. Therefore, we can choose an ordering to evaluate the parts and keep track of the total accumulated score thus far. If this partial score is below an intermediate threshold, then we can prune the evaluation tree. Specifically, let

$$\begin{aligned} \text{score}_k(\omega) &= m_0(\omega) + \sum_{i=1}^{k} \max_{\delta_j \in \Delta} (m_i(a_i(\omega) \oplus \delta_j) - d_i(\delta_j)) \\ &= \text{score}_{k-1} + \max_{\delta_j \in \Delta} (m_k(a_k(\omega) \oplus \delta_j) - d_k(\delta_j)) \end{aligned} \tag{4.16}$$

then, hypothesis pruning is stopping the evaluation at location ω if $\text{score}_k(\omega) < t_k$. In other words, placing parts k through n will not make $\text{score}(\omega)$ go above T. Next, deformation pruning skips the evaluation of $m_k(a_k(\omega) \oplus \delta_j)$ if $\text{score}_{k-1}(\omega) - d_k(\delta_j) < t_k^{'}$. In other words, displacing the part be δ is too expensive to allow $\text{score}(\omega)$ to go above T.

Of course, we must also find the thresholds t_k and $t_k^{'}$. To do this, a set positive examples E are randomly sampled from the n positive training samples. Then, the tightest thresholds are chosen which allow all of the examples in E to pass the cascade. In the paper, it is shown that this approach bounds the probability to be $\leq \gamma$ that the threshold is within ϵ of optimal as long as $|E| \geq 2n/\epsilon \ln(2n/\gamma)$ [84].

4.2.2 Mapping to the Visual Utility Framework

Now that we have described the star cascade as it is presented by its authors in [84] let us see how it is equivalent to a cascade of visual utility filters that near-optimally minimizes the visual utility risk from eq. (3.7). Let us start with describing the high-level algorithm from the visual utility framework. That is simply evaluating equation eq. (4.15). Note that because of deformation pruning, not all δ_j in the maximum will be passed to the high-level algorithm. It simply uses those locations that are passed to it. Therefore, the set of regions to evaluate in the visual utility filter $\bar{\Omega}$ is given by the location of the root filter and the displacements of all the parts. Formally,

$$\bar{\Omega} = \{I(\omega, \delta_1, \ldots, \delta_n)\} \tag{4.17}$$

Now, the star cascade is defined as a set of visual utility filters that operate on a given permutation of the parts $\Pi = \{\pi_1, \ldots, \pi_n\}$. For each part, there are two associated visual utility filters: a deformation filter and a hypothesis filter. The deformation filter for part in position π_k uses the visual utility estimator

$$\text{score}_{k-1}(\omega) - d_k \delta_j \geq t_k' \tag{4.18}$$

The hypothesis filter uses the visual utility estimator

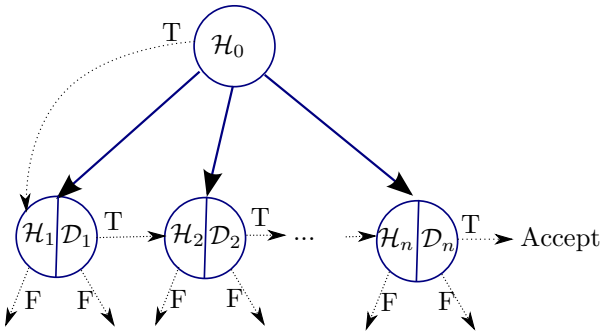


Figure 4.3: The star cascade (dotted) operating on the deformable parts model (solid), which is composed of a root filter and n parts. For each part, there are two filters: a hypothesis filter and a deformation filter.

$$\text{score}_k(\omega) \geq t_k \tag{4.19}$$

4.2.3 Theoretical Analysis

Now, let us look at the process of selecting all the t_k and t'_k values. Remember that this is done using a set of positive training examples E and selecting the tightest thresholds that will pass all of the examples. The thresholds for each part are selected in order from the root filter to part π_n because the intermediate scores are needed. We will now show how this process greedily minimizes a super modular function in the same way as described in the general case in section 3.4.3 and is thus near optimal.

First, let us define the set of possible filters V to be the set of all hypothesis and deformation filters that pass all of the positive examples. Therefore, for part π_k the set of hypothesis filters is defined by

$$\mathcal{H}_k = \{x | \text{score}_k(I(\omega)) \geq x \forall I \in E\}, \tag{4.20}$$

and the set of deformation filters for part π_k is defined by,

$$\mathcal{D}_k = \{x' | \text{score}_{k-1}(I(\omega)) - d_k\delta_j \geq x' \forall I \in E\}. \tag{4.21}$$

Therefore, the set V is defined by

$$V = \mathcal{H}_0 \cup \bigcup_{k=1}^{n} \mathcal{D}_k \cup \mathcal{H}_k. \tag{4.22}$$

Now, setting the thresholds is equivalent to selecting a set $A \subseteq V$ where for each part π_k,

$$t_k = \begin{cases} \min\{h_k \in A\} & \text{if } \exists h_k \text{ for } \pi_k \\ -\infty & \text{otherwise} \end{cases} \tag{4.23}$$

and

$$t'_k = \begin{cases} \min\{d_k \in A\} & \text{if } \exists d_k \text{ for } \pi_k \\ -\infty & \text{otherwise} \end{cases} \tag{4.24}$$

Next, using this formulation, let us examine the Bayes Risk (BR) of the resulting detector. From chapter 3, remember, that $BR = C_F P(\neg RH) + C_M P(R\neg H)$ where C_F is the cost of a false positive, C_M is the cost of a miss, R is the set of examples that are used by the high-level algorithm and H is the set of examples that pass the aggregate visual utility filter. Let us start our analysis with the miss term. By construction, all of the filters in V are those that pass every example from the training subset. Therefore, the miss rate on the training subset is 0 and we can ignore the miss term. Next, let us look at the false positive term. We have already shown, in eq. (3.50) that the cost of a false positive is a supermodular function when building any cascade of visual utility filters. Therefore, in the star cascade, setting the thresholds is also a supermodular function with respect to the false positives.

Finally, let us examine the computational cost D. In our baseline scenario, when $A = \emptyset$, all possible locations in the image are passed through the entire cascade. Let $D(\emptyset) = \bar{D}$. Now, for any location evaluated by the cascade, the computational cost cannot increase if part of the cascade is skipped. Therefore, filtering out more examples will lead to a monotonically decreasing computational cost. In other words $\forall A \subseteq B \subseteq V$,

$$D(B) \leq D(A) \leq \bar{D}. \tag{4.25}$$

Now, to show that D is supermodular, remember that a function F is supermodular if $\forall A \subseteq B \subseteq V$,

$$F(A \cup \{x\}) - F(A) \leq F(B \cup \{x\}) - F(B). \tag{4.26}$$

Let us examine adding a new filter x in the four possible scenarios. The first scenario is where x is associated with a part π_k and there is already a filter $y \in A$ with a lower threshold than the threshold in x. In that case equations eq. (4.24) and eq. (4.23) minimize the two values and thus the cascade remains unchanged. Therefore,

$$\begin{aligned} D(A \cup \{x\}) - D(A) &\overset{?}{\leq} D(B \cup \{x\}) - D(B) \\ 0 &= 0. \end{aligned} \tag{4.27}$$

Next, let us look at scenario where the new filter has a lower threshold. If the lower threshold is lowered by ϵ such that the evaluation of the filter at any location on any training image is unchanged, then once again, the resulting cascade is unchanged and as before, the supermodularity criterion is met.

Finally let us look at the two scenarios where the threshold is tightened such that some instances ω are now filtered on part π_k when they were not earlier by the filters in set A. Let us call the set of these instances Z. For all the examples in Z the computational savings comes from not having to evaluate those examples further down the cascade. Therefore, $D(A \cup \{x\}) - D(A) = \tau(Z) < 0$ where τ is a monotonically decreasing function.

There are two cases here that depend on the entries in B. In the first case, there are some filters in $B \setminus A$ that change the thresholds for parts $\leq k$. In this scenario, let Y be the set of instances filtered before part π_k by the filters $B \setminus A$. Now, for instances in $Y \cap Z$, the savings from not evaluating further down the cascade are already accounted for in $D(B)$. Therefore, $D(A \cup \{x\}) - D(A) \leq D(B \cup \{x\}) - D(B)$ and the supermodularity criterion is met.

In the second case, there are some filters in $B \setminus A$ that change the thresholds for parts $> k$. In this case, the computational savings from filtering out the examples in $Y \cap Z$ using x will be reduced. Therefore, once again, $D(A \cup \{x\}) - D(A) \leq D(B \cup \{x\}) - D(B)$ and the supermodularity criterion is met.

Since these four scenarios cover all the possibilities for x, A and B and in all of them, the supermodularity criterion is met, the process of adding a new filter is a supermodular process on the computational

cost. Finally, since the addition of two supermodular functions is also supermodular, the combination of the computational cost and the Bayes Risk is supermodular in this case.

Therefore, building a star cascade is equivalent to minimizing a supermodular function, which we know can be done near optimally using a greedy approach.

Finally, we must show that the process of setting the thresholds to their tightest bound from the root filter to part n is in fact equivalent to the greedy process. Within a part, this is true because choosing an x that has a tighter threshold will always decrease the cost over a looser threshold and so the greedy thing to do is always to choose the tightest threshold. Across parts, this is also true because the computational savings from filtering early is always going to be greater than the savings from filtering later in the process. Furthermore, the model is constructed in [83] so that the parts are ordered greedily by their ability to discriminate positive examples on the training set. This implies that filters on the earlier parts will necessarily filter out more locations.

4.2.4 Discussion

In this section, we have shown that even though it is trained greedily, the star cascade is a near-optimal implementation of a visual utility filter for a deformable parts model. It is guaranteed to speed up detections using the deformable parts model, while on a training set, it is guaranteed to reduce the false positive rate while keeping the miss rate constant. As long as the training set is representative of the data stream in the application, these strong results will hold. This theoretical exercise helps to explain the strong performance of the technique shown in the literature, while at the same time, provides a concrete example of our visual utility framework implemented in a state-of-the art algorithm.

Chapter 5

SCATAT Algorithm

> If its slower than me and stupider than me ... pass the salt.
>
> Anthony Bourdain

In chapter 4, we show that foundational cascade algorithms attempt to minimize the visual utility risk (eq. (3.7)) by first constraining the problem and then using a greedy approach. However, we have to question if we can do better by optimizing the underlying cost function directly. In this chapter, we present a fast iterative algorithm that approximately minimizes the visual utility risk directly. The algorithm is called Submodular Cascade Assembly with Time Accuracy Trade-off (SCATAT) and is based on a generic SupSub procedure from [133].

5.1 Minimizing the Difference of Two Submodular Functions

Recall from section 3.4.3, that the visual utility risk, which is being minimized, can be written in the form

$$F(A) = G(A) + J(A), \tag{5.1}$$

where $G(A)$ is submodular and equal to:

$$G(A) = D_A + C_M P(R_t \neg A), \tag{5.2}$$

while $J(A)$ is supermodular and equal to:

$$J(A) = D_{R_t} \tau(P(A)) + C_F P(\neg R_t A). \tag{5.3}$$

Minimizing $F(A)$ is equivalent to minimizing the difference between two submodular functions $G(A)$ and $-J(A)$. Approximate algorithms to this generic problem are presented in [133, 209]. All of these approaches are discrete versions of the convex-concave optimization from [328]. In order to achieve the approximate solution, these algorithms inductively build up a sequence $A_0, A_1, \ldots, A_i, A_{i+1}, \ldots$ such that $F(A_0) \geq F(A_1) \geq \cdots \geq F(A_i) \geq F(A_{i+1}) \geq \ldots$. At each step, at least one of the terms in the function is replaced by a modular approximation that is tight at A_i and is an upper bound on the true function. Then, a standard submodular optimization algorithm is used to find A_{i+1}. In particular, the SubSup algorithm replaces $J(A)$ and then minimizes the resulting submodular function, the SupSub algorithm

replaces $G(A)$ and minimizes the resulting supermodular function and the ModMod algorithm replaces both and minimizes the resulting modular function. Formally, the approximation function $K_i(A)$ is one that meets the three criteria in table 5.1.

	SubSup	**SupSub**	**ModMod**	
$K_i(A_i) =$	$J(A_i)$	$G(A_i)$	$G(A_i) + J(A_i)$	
$K_i(A) \geq$	$J(A)$	$G(A)$	$G(A) + J(A)$	$\forall A \in 2^V$
F_i is	submodular	supermodular	modular	

Table 5.1: Requirements of the modular approximation function K_i for the different approximate minimization algorithms

Once we have this approximation function, we pick $A_{i+1} = \arg\min_{A \in 2^V} F_i(A)$. In the SubSup case, F_i is submodular and can be minimized in polynomial time [130]. Unfortunately, the polynomial time algorithm from [130] runs in $O(n^8 log^2 n)$ and so is not practical for building cascades out of very large sets of candidate filters. In the SupSub case, F_i is supermodular and can be minimized using the greedy approach to generate a near optimal solution in linear time [35]. Finally, in the ModMod case, the function can be minimized exactly in linear time [133]. In practice, all three of these approaches perform the minimization similarly, but the SubSup algorithm is significantly slower because solving the generic submodular problem is much slower [133]. Therefore, for the SCATAT algorithm, we employ the SupSub approach which is reproduced in alg. 1. The next section describes the SCATAT algorithm in detail.

Algorithm 1 SupSub algorithm from [133].

Input: G, J
Output: Approximate minimizer of $G + J$

```
1:  i = 0
2:  min = inf
3:  improvementFound = false
4:  while improvementFound do
5:     K_i = ModularApproximation(G, A_{i-1})
6:     A_i = arg min_{A⊂V, A≠∅} (J + K_i)(A)
7:     val = (J + K_i)(A_n)
8:     i = i + 1
9:     if val < min - δ then
10:       min = val
11:       improvementFound = true
12:    end if
13: end while
14: return A_{i-1}
```

5.2 SCATAT Algorithm

In order to describe the SCATAT algorithm, we must examine two parts of the generic SupSub algorithm in more detail. We will first look at the supermodular minimization that moves the solution from A_i to A_{i+1} (line 6). Then, we will describe the modular approximation $K_i(A)$ that provides an upper bound on $G(A)$ (line 5).

5.2.1 Minimizing the Supermodular Function

When minimizing the submodular function, we start with the cascade represented by A_i and then greedily modify the cascade in order to minimize:

$$F_i(A) = J(A) + K_i(A). \tag{5.4}$$

The minimization is described by alg. 2. Each step in the greedy minimization can either remove a stage in the cascade (line 4), or add a new one at the end (line 6). The minimization terminates where no additional change will reduce F_i.

Algorithm 2 Supermodular minimization that is part of the SCATAT algorithm.

Input: K_i, J, A_i
Output: Approximate minimizer of $J + K_i$

```
1:  B = A_i
2:  F_i = J + K_i
3:  while True do
4:     x^- = arg min_{j∈B} F_i(B \ j)
5:     δ^- = F_i(B \ x^-)
6:     x^+ = arg min_{j∈V\B} F_i(B ∪ j)
7:     δ^+ = F_i(B ∪ x^+)
8:     if δ^- < F_i(B) and δ^- < δ^+ then
9:        B = B \ x^-
10:    else if δ^+ < F_i(B) then
11:       B = B ∪ x^+
12:    else
13:       break
14:    end if
15: end while
16: return B
```

5.2.2 Modular Approximation

Many different approximation functions could be used for the modular approximation as long as they meet the criteria in table 5.1. [209] defines a universal approximation function as follows. Given π, which is any permutation of V, let $W_i = \{\pi(1), \pi(2), \ldots, \pi(i)\}$. Then, we can define a function $K : V \to \mathbb{R}$:

$$K(\pi(i)) = \begin{cases} G(W_1) & \text{if i = 1} \\ G(W_i) - G(W_{i-1}) & \text{otherwise} \end{cases} \tag{5.5}$$

which defines K for all of the single element subsets of V. For all $A \subseteq V$, it is defined by:

$$K(A) = \sum_{x \in A} K(x). \tag{5.6}$$

Now that K_i is defined, we must still specify the permutation needed in order to calculate it. [209] uses a random permutation and show that the solution cannot be improved by adding or deleting a single element to the subset. This means that only $O(n)$ permutations need to be examined to find the optimal solution. In practice, [133] observed that ordering with respect to the greatest improvement relative to A_i is effective.

In the SCATAT algorithm, we take a slightly different approach and define the modular approximation using the structure of the cascades. In particular, we take advantage of the following modular bound that [133] established for submodular functions:

$$f(Y) \leq f(X) - \sum_{j \in X \setminus Y} f(j|V \setminus j) + \sum_{j \in Y \setminus X} f(j|X), \tag{5.7}$$

where V is the set of all candidate filters and $f(X|Y) \triangleq f(X \cup Y) - f(Y)$ is the change of adding X starting with cascade Y. This results in the following modular approximation function.

$$K_i(A) = G(A_i) - \sum_{j \in A_i \setminus A} G(j|V \setminus j) + \sum_{j \in A \setminus A_i} G(j|A_i), \tag{5.8}$$

Let us look at each of the summation terms separately. The first one represents the change in cost from removing stages in A_i:

$$G(j|V \setminus j) = D_j P(j) + C_M[P(R_t \neg V) - P(R_t \neg (V \setminus j))], \tag{5.9}$$

where $P(j)$ is the probability of a region getting to stage j. In the scenario of building cascades, there will typically be at least one filter that blocks all of the candidate regions. Therefore, if all of the potential filters in V are in the cascade, every stage will be missed and $P(R_t \neg V) = P(R_t \neg (V \setminus j))$. Therefore, the change in cost from removing the stage reduces to:

$$G(j|V \setminus j) = D_j P(j). \tag{5.10}$$

Next, let us look at the change in cost to add a new stage:

$$\begin{aligned} G(j|A_i) &= D_j P(A_i) + C_M[P(R_t \neg (A_i j)) - P(R_t \neg A_i)] \\ &= D_j P(A_i) + C_M P(A_i R_t \neg j). \end{aligned} \tag{5.11}$$

Thus, the upper bound on the cost to add a new stage is the sum computation cost of j and the cost of any new misses in stage j if j were to be added to A_i. Combining eqs. (5.8), (5.10) and (5.11)

$$K_i(A) = G(A_i) - \sum_{j \in A_i \setminus A} D_j P(j) + \sum_{j \in A \setminus A_i} D_j P(A_i) + C_M P(A_i R_t \neg j), \tag{5.12}$$

we can see that K_i only depends on A_i. Therefore, K_i only needs to be computed once at the beginning of alg. 2 for all candidate stages, reducing the processing requirements.

Alas, in alg. 2, the $J(A)$ term must be recalculated in each iteration, significantly increasing the training time of the cascade. Fortunately, $J(A)$ is supermodular and we can thus use the lazy evaluation approach from [202]. In this approach, a priority queue is initialized for all filters that can be added to or removed from the cascade with their change in cost from A_i. Then, during each iteration, the filter that produces the lowest cost is popped off the queue and reevaluated. If it remains the best filter, then it is chosen and otherwise it is replaced in the priority queue with its new change in cost. Supermodularity guarantees that minimum change in cost from adding or removing a filter occurs when the cascade is A_i. Therefore, if a filter is reevaluated and is still the one that produces the minimal cost, it is guaranteed to be the best modification. In practice, this can reduce the training time by order of magnitude in deep cascades.

5.3 Conclusion

In this chapter, we have detailed the SCATAT algorithm which builds a cascade by approximately minimizing the visual utility risk. It is an iterative process that adapts the SupSub algorithm from[133]. At each step, the submodular component of the visual utility risk is approximated by a tight modular approximation and then greedy minimization is performed with lazy evaluation. The resulting algorithm is fast to run and is the first one proposed which explicitly trades off computation time and task accuracy.

Chapter 6

Visual Utility in Deep Neural Networks

I like nonsense; it wakes up the brain cells

Dr. Seuss

6.1 Introduction

Recently, there has been a revolution in computer vision caused by the re-popularization of deep neural networks. This chapter examines these new approaches using the lens of visual utility and submodularity in order to both analyze theses approaches and highlight known, successful examples. In particular, we will examine visual utility filtering for deep networks, adaptive searching, object detectors with integrated region evaluation and finally, discuss techniques to construct deep networks.

The revolution started in 2012 with Krizhevsky et al. using a deep convolutional neural network to demolish the competition on the ImageNet Large Scale Visual Recognition Challenge [244]. Neural networks had fallen out of favor in the late 1990's and early 2000's in favor of engineered features and statistical techniques. However, in their paper, Krizhevsky et al. were able to show that a properly trained neural network could perform significantly better than the engineered approaches [160]. Three main changes had occurred to enable this result. First, computational power has increased significantly, especially the availability of GPUs for processing, which are architecturally well suited for neural networks. Second, improvements have been discovered in how to train these kinds of networks (e.g. [118]). Finally, large, labeled datasets, like ImageNet, are now available.

Throughout most of this thesis, computational cost is treated synonymously with processing time, which is most relevant in situations like robotics which have a time deadline. In a single core CPU world, each region you do not examine directly saves you time. This is not the case when parallelization can happen very easily like on GPUs, which are critical for efficient neural networks. However, GPUs still incur a processing cost on a per region basis, not necessarily in time, but in throughout. Thus, the visual utility framework for deep neural networks is more appropriate for applications where the high-level algorithm is very complex and there are many images to process, like at the web scale. In these settings, the computational cost should is proportional to the throughput lost.

Since the seminal result on ImageNet, there has been a rush to apply deep neural networks to a number of problems. In the computer vision community, let us examine the problems that directly use convolutional neural networks like the one that Krizhevsky et al. used on ImageNet. There have been a

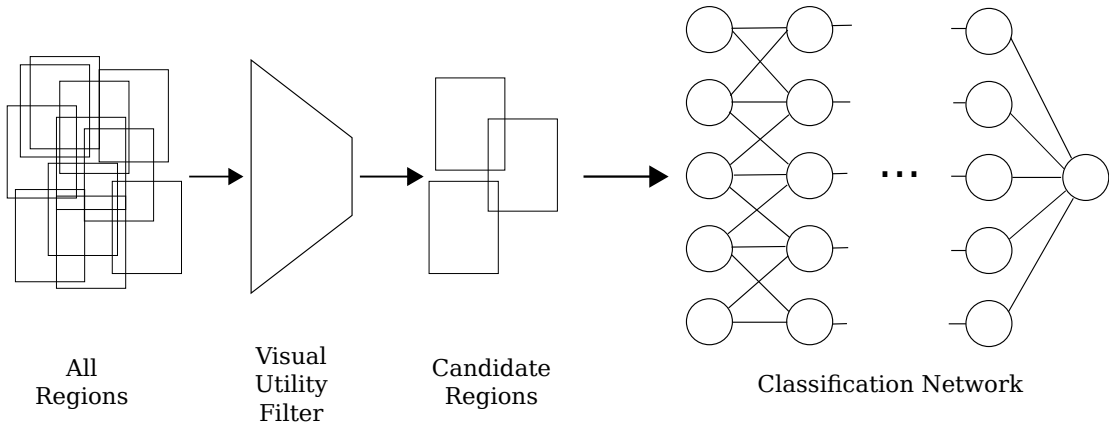


Figure 6.1: Deep neural networks can be used with visual utility filters. The filter reduces the set of candidate regions that the network classifies, thus reducing the computation cost to process an image.

few evolutions of these kinds of networks, with deeper ones (e.g. GoogLeNet[271] and VGGNet[261]) or wider ones (e.g. [14]). However, all of these networks are designed to take a single image and output a confidence that the image contains each type of object. In order to use these networks for more complex tasks, like object detection, something extra is needed.

6.2 Visual Utility Filtering

One approach to use the image classifications networks for object detection is simply to apply a sliding window approach and evaluate each candidate window using the network (fig. 6.1). However, the neural networks can easily take 50ms to evaluate a region [185]. This is quite fast, however, when applied to hundreds of thousands of candidate regions, the resulting runtime becomes intractable. This leaves us in exactly the same situation examined in HOG-based object detection in chapter 7, where visual utility estimators can be effective. In this scenario, a visual utility filter can be chosen to reduce the number of candidate regions evaluated by the neural network. This estimator could be built either by training a new neural network or using another learning technique with potentially different features. Furthermore, multiple estimators could be combined in a cascade to create an aggregate estimator that can be both fast and accurate. This can be done near optimally using the SCATAT algorithm from chapter 5.

In fact, we see many examples of visual utility estimators being used for object detection with deep neural networks. For example, a popular estimator is an objectness estimator called Selective Search. In this work, Uijlings et al. generate candidate bounding boxes by hierarchically grouping together super pixel regions. Their fast mode produces approximately 2000 candidate regions in 3.8 seconds with 98% recall on the PASCAL 2007 dataset [289]. Edge Boxes [335] and MultiBox [79] are other examples of new objectness estimators that run even more efficiently than those examined explicitly in the experiments of this thesis.

Once a much smaller set of candidate regions are proposed, they are then evaluated by a neural network. Szegedy et al. do this with the GooLeNet network and just use the resulting classification [270], whereas Girshick et al. train a convolutional network to generate features and then evaluate those features using an SVM [97]. A faster version that reuses computation and performs dimensionality reduction is presented in [96]. Their resulting architecture is called R-CNN and has been used for more complex tasks. For example, Karpahy et al. feed the top 19 R-CNN objects into a CNN [150] or a recurrent neural network [149] in order to generate sentences describing each image.

As shown throughout this thesis, the more complex the higher level task, the more likely that the

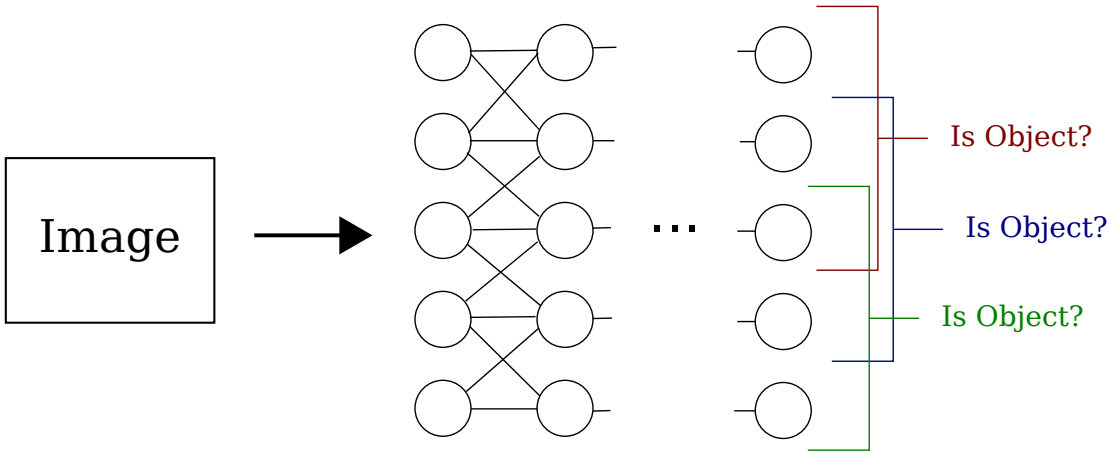


Figure 6.2: A deep neural classification network can be converted to one for detection by removing the last, fully connected layers so that the spatial information is not lost. Then, a classifier can be applied on sub portions of the final layer in order to detect the desired objects. This is equivalent to converting the image into a different feature space, which (ideally) makes classification easier so that it is faster and more accurate.

benefits of using visual utility estimators will be greater than their cost and so it is not surprising to see them used as early steps for complex tasks involving deep neural networks. However, visual utility filtering for deep neural networks has been used thus far in an ad hoc manner. The approaches in this thesis and especially the SCATAT algorithm from chapter 5 allow the filters to be used in a more principled way to trade off task accuracy and computational cost.

6.3 Integrated Region Evaluation

Instead of using a visual utility estimator in front of a image classification net, it is possible to change the network for object detection. In the original networks, there are fully connected layers at the end that destroy the spatial relationships from the image. So, these layers can be removed and replaced with a classifier that can be applied efficiently at each location on the neural network features (fig. 6.2). This approach reuses the computation for lower level features and is thus more efficient. Long et al. describe that classification as more neural layers that eventually output a classification confidence for each pixel and can thus be used for segmentation [185]. He et al. use spatial pyramid pooling layers to generate features that allow many scales and aspect ratios to be examined quickly by an SVM classifier [115]. Finally, the OverFeat architecture applies a small number of small, fully connected layers, in a sliding window approach that reuses a lot of the calculations. This allows the confidence at each potential location to be evaluated quickly in a similar way to the integral HOG features from our experiments. Then, instead of using non-maximal bounding box suppression, they combine the scores of similar boxes to merge them and find the most likely location of various object in the scene [253].

Changing the networks for object detection extracts features from a mid layer that are highly discriminatory allowing a per region evaluation to be done very quickly. In the visual utility framework of this thesis, this situation is equivalent to transforming the input space in order to enable a low cost, high-level algorithm. As our theoretical framework (chapter 3) describes, in this scenario, there is little value of having extra visual utility filters.

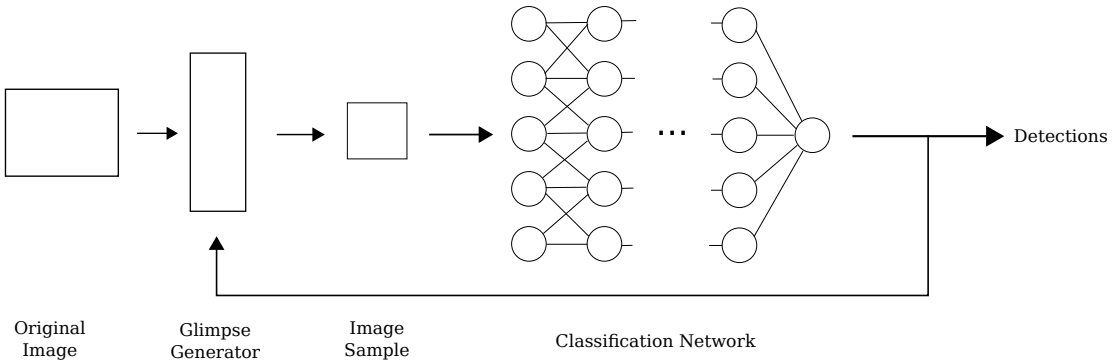


Figure 6.3: An adaptive approach can also be used to detect things of interest in an image with a deep neural network. In this case, a sequence of samples are taken of the image and evaluated by a deep neural network. A sample could be a portion of the image, or even a foveal representation with varying resolution. A glimpse generator uses the previous results to decide where the next sample should be taken. This generator could be another neural network (like a Long Short Term Memory module [119]) or some other kind of predictor.

6.4 Adaptive Searching

Thus far we have discussed two ways to build a detection system. First, one can use visual utility filters to reduce the regions inspected by a classification network. Second, one can integrate region evaluation into the network in order to produce predictions at each location. A third approach is inspired by what animals do when they move their eyes around (Section 2.1). As an animal saccades his/her eyes, information is acquired that both helps the task and also informs where to look next. We can use a similar, adaptive approach, when processing a visual scene by machine. In this approach, a sequence of glimpses are taken. At each glimpse, visual information is acquired at a specific location and passed on to a higher level algorithm. Then, the location to sample next is based on the posterior likelihood of finding the most valuable information given what has already been seen. This process is depicted in fig. 6.3. In section 3.5 we show that such a process can be adaptive submodular, and thus near optimal. Furthermore, the adaptive process naturally generates an anytime algorithm and will have a computational cost that is $\Omega(log(n))$ whereas the non-adaptive equivalent will have a computational cost that is $\Omega(n)$ [65].

A neural-inspired system using this glimpse approach was presented by Larochelle and Hinton. They use Restricted Boltzman Machines (RBMs) along with a foveal transformation of the visual data in order to acquire information about the scene to make a classification. At each step, a controller is used which estimates the probability vector for each hidden unit in the RBM [172]. A more recent example of a of glimpse network is presented in [15, 203]. It uses convolutional neural networks along with a Long Short Term Memory (LSTM) module in a Recurrent Neural Network (RNN) in order to decide where next to look. The LSTM module encodes the relevant history of where the network has "looked" but can also optionally forget in order to limit the amount of memory required. The resulting network is trained using reinforcement learning because the attention model does not optimize a differentiable loss function. Xu et al. take a further step with this approach by adding two components. First, instead of simply classifying an image, the RNN generates a sentence describing the image. Each word in the sentence is then associated with a particular glimpse. Second, a soft attention model is presented which doesn't reduce the computational cost in current hardware, but is differentiable and thus, easier to train [318].

Glimpse networks and attention models are just starting to become an area of interest in the deep neural network community. We can now examine these early proposals based on our understanding of the adaptive search process that is described in detail in section 3.5. For such a process to be near-optimal requires two conditions. First, the utility function must be adaptive submodular. Second, the process that decides the next glimpse must do so greedily at each step in order to maximize the utility function.

In [15, 203], the utility function is a delayed reward function equal to 1 if the object is correctly classified at time t and 0 otherwise. This is not a submodular function because an attention at time $t-1$ would not be able to correctly identify the object whereas choosing it at time t would. Xu et al. optimize the log likelihood of a sentence being generated using the sampled trajectory [318]. This is also not submodular because it is equivalent to solving a POMDP, which is known to not be submodular [99].

The second condition for these approaches to be near optimal are that the selection of locations is done greedily. All of the glimpse networks mentioned above encode a strategy that attempts to minimize an objective function. However, it is unclear that the resulting choices are going to be equal to the greedy ones at each step because there are no formal guarantees known for these types of networks. That being said, if we could bound the error at each step relative to the greedy choice, even probabilistically, then we can bound the overall performance. In particular, as proven in [99], if the network is guaranteed to be within ϵ at each step, then after k steps, we get the following guarantee:

$$f_{avg}(\pi) \geq (1 - e^{-1}) f_{avg}(\pi_{opt}) - k\epsilon \tag{6.1}$$

Overall, we can see that existing approaches to use deep neural networks to adaptively sample an image are not adaptive submodular. In order to use a submodular process, an adaptive submodular function must be defined on the network output and the resulting trained networks that predict the next glimpse must be bounded relative to the greedy approach. We have defined an initial adaptive submodular function in section 3.5 that can be used, while we have identified how the performance of the trained network must be bounded. However, both of these components will benefit from advancements in the theoretical understanding of how deep neural networks are trained and the functions they encode. These advancements are left for future work.

6.5 Building Near Optimal Networks

So far, we have seen approaches that take an existing neural network and either adapt them or wrap them to change their focus of attention. Next, let us examine a different aspect of deep neural networks that could benefit from submodularity: the structure of the network itself. Currently, finding the structure of a successful network usually requires significant trial and error because learning the structure on top of the weights can be computationally intractable. Unfortunately, the cascade building approach from chapter 5 cannot be used directly to build networks because neural networks pass partial information to later layers using multiple paths, while the submodular guarantees in chapter 3 require that visual information is stopped by the filter. However, there have been a few examples of building the network structure by constraining the structure definition problem and these approaches actually use submodularity. In this section, we will examine hierarchical modular optimization [321] and layer-wise learning [11, 25, 117].

6.5.1 Hierarchical Modular Optimization

Hierarchical Modular Optimization (HMO) is a technique presented in [321] to build a neural network using a combination of subnetworks that have a specific structure and are called hierarchical convolutional models. In particular, each subnetwork contains a sequence of layers that operate in the following order: filter, threshold, normalize and finally pool. Furthermore, any of the operations in the stack can be skipped for a given subnetwork. Finally, N subnetworks are combined using a fully connected layer.

The subnetworks to use are chosen iteratively. At each step i, a network n is chosen from all the possible subnetworks that maximizes the following score function:

$$score(n, S) = \sum_{s \in S} 1 \text{ if } F(n, s) \text{ and } \nexists\, m \in N_{i-1} | F(m, s) \tag{6.2}$$

where $F(n, s)$ is true if the network n correctly classifies the training example s and N_{i-1} is the set of subnetworks chosen by the end of step $i-1$. This is a classic example of a submodular set coverage function and thus, this learning process is near optimal.

Intriguingly, the primary motivation in Yamnis et al.'s work is to determine if there are artificial neural networks that are structured similarly to the neural structures in the primate brain. Therefore, they correlate the responses in the trained network with the neural responses in a Macaque monkey and find a strong correlation. This implies that the neural structures in the brain might be growing in a submodular way.

6.5.2 Layer-wise Learning

Another way to build a deep network is to iteratively select layers and attach them in a network. This was shown first by Hinton et al. and analyzed by Bengio et al. where each layer must be a Restricted Boltzman Machine. The structure of the network is trained unsupervised on the data in order to be able to capture the information in the dataset and then a supervised learning step is used to update the weights in the network. In the unsupervised step, successive layers are chosen iteratively such that they minimize the KL-divergence between the output of the previous layer and the dataset [24, 25, 117]. We know from [134] that KL-divergence is submodular and thus, Hinton et. al's process is approximating a submodular optimization. However, the process used to train the RBM at each layer does not guarantee the minimum KL-divergence and therefore, the overall algorithm does not achieve the tightest submodular bound. That being said, it is known to be a good approximation [45] and can still be considered near optimal.

Another greedy approach to build a network layer-wise is presented by Arora et al. In this approach, the type of network is limited to layers that are connected sparsely and randomly. This type of structure is not typically used in practice. For instance, convolutional neural networks are connected in a structured, localized way. Therefore, the results presented are more of theoretical interest than of practical use at this time. That being said, they present some provable bounds for their processes that creates an auto-encoding network in an unsupervised manner [11].

6.6 Conclusions

The field of computer vision has change significantly in the last few years with the reintroduction of deep neural networks. In this chapter, we have examined multiple ways to use visual utility and submodularity in order to both optimize task accuracy in neural nets or to trade off task accuracy with computational cost. In particular, we seen that visual utility filtering, adaptive searching and constructing deep neural networks can all be done near optimally by structuring the problem in a submodular way. We have also shown that some prior work in these areas is provably near optimal and have discussed some techniques to improve on the ad hoc status quo.

Chapter 7

Experimental Case Study in Object Detection

> Insanity is relative. It depends on who has who locked in what cage.
>
> Ray Bradbury

In this chapter, we present a complete experimental analysis for pedestrian detection. We use pedestrian detection because it is a well studied problem and is indicative of the generic object detection problem as evidenced by the results of PASCAL challenges where the performance on pedestrian detection is strongly correlated with performance on overall object categories (fig. 7.1). We will first describe the pedestrian detection problem and its possible applications. Then, we will detail how our generic visual utility framework can be implemented to solve this problem. Finally, we will detail four classes of experiments that characterize the performance of using visual utility in different scenarios.

7.1 Problem Description

Automatic people detection and tracking can be useful in many scenarios. It can be used in surveillance situations, pedestrian traffic analysis or interactive robotics. In this analysis, we will focus on pedestrian detection for automated vehicles that operate in minimally structured real world environments. For example, an automated car that travels on public roads and must never hit a pedestrian. Another relevant scenario would be for mining vehicles. The size of these vehicles can make it challenging for the operator to see around the vehicle and know if a person is in a dangerous location. Cameras around the vehicle could watch for people and alert the operator if the person is in danger. In both of these scenarios, the vehicles operate in a challenging environment where all computing hardware must be rugged and portable. Therefore, computation is expensive and it could be valuable to minimize the computation required. Furthermore, the pedestrian detection would be present in a safety critical system and thus must have high detection rates, while at the same time minimize the false positives that an operator must pay attention to.

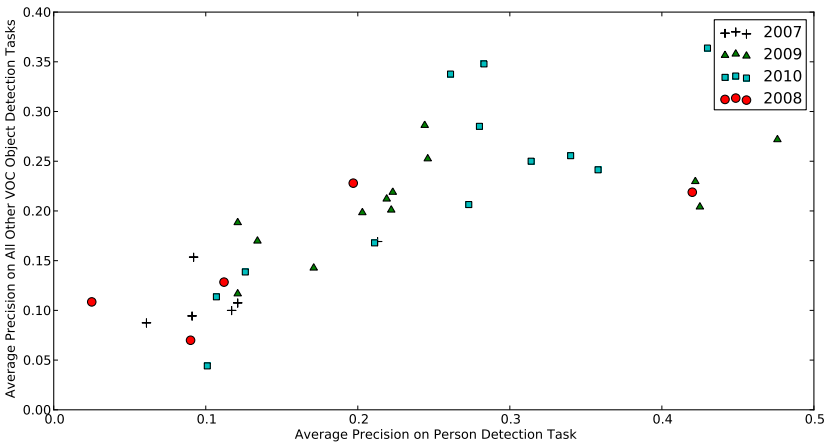


Figure 7.1: The average precision on pedestrian detection in the PASCAL VOC object detection challenges from 2007-2010 are strongly correlated with the mean average precision on all other tasks. Spearman rank coefficient = 0.891182 ($p < 1e\text{-}10$)

7.2 Experimental Framework

These experiments are conducted by specializing the generic visual utility filtering framework from fig. 1.3. In this case, our high-level algorithm is the Histograms of Oriented Gradients (HOG) detector from [60] as implemented in OpenCV 2.3.1 using the default pedestrian detector. The detector was trained on Dalal and Triggs' INRIA dataset and is a model that is 64x128 pixels. Using this model, our search space is a set of bounding boxes where the ratio of height to width is known. Therefore, the initial set of candidate windows is $\Omega = \{I(x, y, s)\}$ where x and y are the coordinates of the box and s is the scale.

We chose not to use the HOG detector instead of more recent detectors, like the parts-based pedestrian detector from [83], because it presents a harder challenge. As shown in chapter 3, the theoretical ability of a visual utility system to speedup the overall system is partially governed by how fast the high-level algorithm runs; the faster the high-level algorithm runs, the less potential for an overall speedup. Given that the HOG detector is already quite fast to compute, we felt it was a more challenging choice for this study.

Unless specified otherwise, the initial set of bounding boxes are uniformly sampled throughout each image in steps of 8 pixels spatially and by 10% increments in scale. We then evaluate the visual utility estimation for each bounding box and if it surpasses a threshold, it is passed on for evaluation by the HOG detector. Computational cost is measured by overall processing time including the amount of time spent computing the visual utility estimator on the entire frame plus the time to evaluate the HOG detector on all of the windows that were passed by the estimator. All of the experiments are performed on a 2.66GHz Intel Core 2 machine with 8GB of RAM.

7.2.1 Datasets

All experiments for this task are conducted using two datasets: the Heimonen Mining dataset [116] and the ETH Pedestrian dataset [80].

The Heimonen HIMA dataset contains 7 annotated video sequences of a few people walking in mine settings around equipment (fig. 7.2). Data was taken simultaneously using wide angle stereo, narrow angle stereo, high dynamic range and infrared cameras. The frames from the stereo sequences that we use in this

Figure 7.2: Examples from the Heimonen HIMA dataset.

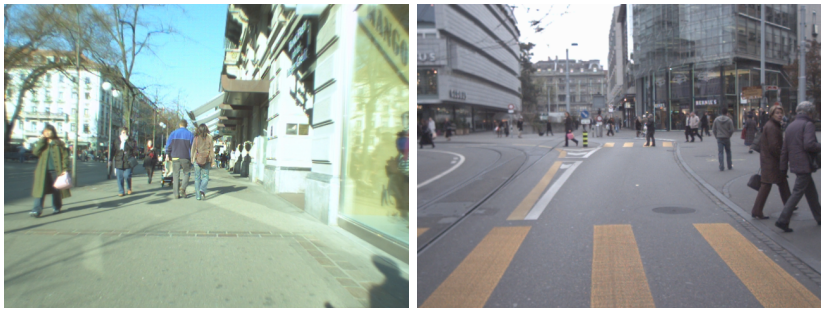


Figure 7.3: Examples from the ETH dataset.

thesis are 1024x768 pixels. Some sequences are from a stationary vehicle, while others are from a moving platform.

The ETH Pedestrian dataset contains 7 annotated video sequences of people walking in a crowded pedestrian mall in Zurich (fig. 7.3). Each frame is 640x480 pixels. The video is taken from a platform moving into the scene in a similar manner as a car. The scenes are very densely packed with people, although only people greater than about 50 pixels are labeled. Stereoscopic data is also available, but not used in this study.

7.2.2 Evaluation Criteria

All evaluation is done on a per-frame basis. For each frame, a bounding box that is identified by the visual utility procedure is compared to the ground truth boxes in the dataset using the PASCAL overlap criterion [81]; the box h is a hit if there exists some ground truth box on the frame g such that:

$$\frac{|h \cap g|}{|h \cup g|} > 0.5 \tag{7.1}$$

where $|h \cap g|$ is the area of intersection of the two bounding boxes and $|h \cup g|$ is the union. Recall is then defined as the number of people labeled in the ground truth data who had at least one hit divided by the total number of people in the ground truth data.

In order to capture a measure of overall performance for a given set point (and thus speedup), we use precision, recall and F-score. F-score is defined as the harmonic mean of the precision and recall or mathematically:

$$F = 2\frac{precision * recall}{precision + recall} \tag{7.2}$$

Finally, we look at the speedup of a given run. The speedup is defined by:

$$Speedup = \frac{t_{hog}}{t_{tot}} \tag{7.3}$$

where t_{hog} is the average computation time per frame of running the HOG detector on the entire frame and t_{tot} is the average computation time per frame of running the visual utility filter to prefilter the bounding boxes and then running the HOG detector on those bounding boxes:

$$t_{tot} = t_{vu} + t_{hog}\tau(p(H)) \tag{7.4}$$

where t_{vu} is the time to compute the visual utility estimator for the entire frame, t_{hog} is the time to run the high-level object detector on the entire frame, $p(H)$ is the probability that a candidate region passes the visual utility filter and τ is a monotonically increasing function from 0 to 1 representing the fraction of time to compute the object detector on a portion of the image. The function τ accounts for the non-linearity of the processing time when processing portions of an image.

We can thus change the speedup by changing the threshold on the visual utility filter. As explained in section 3.4, as long as the computation time of the visual utility estimator is less than the the computation time of the HOG detector, tightening the threshold of the visual utility filter will speed up the overall runtime.

7.3 Experiments

7.3.1 A Range of Visual Utility Estimators

In this set of experiments, we examine the impact of different visual utility estimators. To do this, we use a variety of different visual utility estimators, vary the threshold for each one, and examine the resulting system performance. We set the HOG threshold to be its default value of 0.

In our theoretical formulation detailed in section 3.4, we show that as long as the computational cost of the visual utility estimator is less that the cost of running the HOG detector, tightening the threshold on the visual utility filter will have three effects. First, the overall computational time will decrease. Second, the number of false positives (FP) and true positives (TP) will either stay the same or decrease. Third, the number of missed examples will either stay the same or increase. In other words, recall must either stay the same or go down, while precision could go up or down. Specifically, the precision will go up if:

$$\begin{aligned}
Precision_{HOG} &< Precision_{VU} \\
\frac{TP}{TP+FP} &< \frac{TP-\Delta TP}{TP-\Delta TP+FP-\Delta FP} \\
TP(TP-\Delta TP+FP-\Delta FP) &< (TP-\Delta TP)(TP+FP) \\
TP^2 - TP\Delta TP + TP\cdot FP - TP\Delta FP &< TP^2 + TP\cdot FP - TP\Delta TP - FP\Delta TP \\
TP\Delta FP &> FP\Delta TP \\
\frac{TP}{FP} &> \frac{\Delta TP}{\Delta FP}
\end{aligned} \tag{7.5}$$

where TP and FP are the number true and false positives at the set points for the HOG detector and ΔTP and ΔFP and the true and false positives filtered by the visual utility detector.

7.3.1.1 Visual Utility Estimators

There are seven different visual utility estimators in this experiment along with one control for comparison. Each visual utility estimator must be defined for a given bounding box, so, for those that are defined on a pixel-by-pixel basis, we calculate the relative entropy of the actual visual utility inside the box compared to a hypothetical visual utility estimator that assigns visual utility uniformly throughout the frame. Specifically, for box $B \in \Omega$, let $q(B)$ be the fraction of visual utility encompassed by the box if the visual utility estimator produced a uniform estimate across the image.

$$q(B) = \frac{A_B}{A_I}, \tag{7.6}$$

where A_B is the area of the box and A_I is the area of the image. Then, let $r(B)$ be the fraction of visual utility actually in the box:

$$r(B) = \frac{\sum_{i \in B} V(i)}{\sum_{i \in I} V(i)}, \tag{7.7}$$

$$r(b) = \frac{\sum_{i \in B} V(i)}{\sum_{i \in I} V(i)} \tag{7.8}$$

where $V(i)$ is the visual utility estimate at position i. Then, the score for box B is given by the relative entropy of the visual utility distribution compared to a uniform estimate:

$$H(B) = \begin{cases} -q(B) \log \frac{q(B)}{r(B)} - & \\ (1 - q(B)) \log \frac{1-q(B)}{1-r(B)} & \text{if } \; q(B) < r(B) \\ r(B) \log \frac{r(B)}{q(B)} + & \\ (1 - r(B)) \log \frac{1-r(B)}{1-q(B)} & \text{otherwise} \end{cases} \tag{7.9}$$

Note that this score can be quickly calculated for many boxes using an integral image of the visual utility score. Now, using this measure of relative entropy, we can define the visual utility score of any bounding box. Next, we will detail the seven metrics used in our evaluation.

Spectral Residual Saliency

This is the generic per-pixel visual utility metric defined by [121] that uses the observation that in natural images, the average amplitude of the Fourier spectrum obeys a $1/f$ relationship. Therefore, we can find those areas of the image that are not typical by subtracting out the average Fourier spectrum and re-projecting the result back into image space. Furthermore, we can approximate the average spectrum by convolving the Fourier spectrum of the input image with a box filter. In summary, to calculate the spectral residual saliency $S(x)$ of an image $I(x)$, we use the following equations:

$$\begin{aligned} L(f) &= \log(\Re(\mathcal{F}[I(x)])) \\ P(f) &= \Im(\mathcal{F}[I(x)]) \\ R(f) &= L(f) - h_n(f) * L(f) \\ S(x) &= (\mathcal{F}^{-1}[\exp(R(f) + P(f)])^2 \end{aligned}$$

where $\mathcal{F}$ is a Fourier transform, $h_n(f)$ is a box filter, $L(f)$ is the log spectrum, $P(f)$ is the phase spectrum, and $R(f)$ is the residual. Once the spectral residual saliency is calculated for every pixel, it is calculated for each box using the relative entropy measure mentioned previously.

Laplacian

The discrete Laplacian operator is the sum of the second derivatives in the x and y direction on the image. This can be calculated very quickly and the absolute value of the Laplacian is a good measure of local edge energy [225]. Since edge energy can correspond to areas of interest, we define a visual utility measure as the absolute value of the laplacian calculated over an aperture of 5x5 pixels. This creates a pixel-by-pixel measure of visual utility, which is converted for each box using the relative entropy measure.

Center Surround Histograms

This visual utility measure is inspired by the work presented in [232], which uses center-surround boxes in order to identify regions that are different than their local surrounds. In that work, a pixel-by-pixel saliency measure is defined by first sampling the image with many different overlapping boxes at different scales. For each box, there is defined a region around it. The feature space is then binned. This feature space could include color, motion, intensity, or other image features. Once the feature space is binned, histograms are built for both the box and the surrounding region and then normalized to sum to one. The visual utility of the box is defined as the chi-squared distance between the resulting center and surround histograms. Rahtu et al. then convert that per-box score into a per-pixel score by taking for each pixel, the maximal box score from all the boxes that contain that pixel. In our measure, we simply stop at the per-box definition.

For our implementation, we use the intensity as the only feature and bin the range into 32 bins. To make the computation of many boxes faster, we use an integral histogram. An integral histogram is similar to an integral image except that for each pixel, there is a complete histogram of all the pixels above and to the left of that current pixel. Equivalently, it is a set of integral images stacked together, one for each bin in the histogram. Formally and integral histogram is defined as

$$IH(x, y, b) = \sum_{\substack{0 \leq x' < x \\ 0 \leq y' < y}} \mathbb{1}[I(x', y') \in b] \tag{7.10}$$

Then, the visual utility of a box whose upper left corner is at (x, y) of height h and width w can be calculated by:

$$\begin{aligned} H_{center}(b) &= IH(x, y, b) + IH(x + w, y + h, b) - IH(x + w, y) - IH(x, y + h) \\ H_{surr}(b) &= IH(x - \theta_x, y - \theta_y, b) + IH(x + w + \theta_x, y + h + \theta_y, b) \\ &\quad -IH(x + w + \theta_x, y - \theta_y) - IH(x - \theta_x, y + h + \theta_y) - H_{center}(b) \\ V(B) &= \chi^2 \left[\frac{H_{center}}{\sum_b H_{center}(b)}, \frac{H_{surr}}{\sum_b H_{surr}(b)} \right] \end{aligned}$$

where θ_x and θ_y are the margins to specify the size of the surround region. We grow the region so that each dimension is $\sqrt{2}$ times as large as the box of interest.

Objectness

This is the generic visual utility estimator proposed by [233]. It is a complex feature designed to estimate the likelihood that an a bounding box contains some kind of *object*. It uses a combination of four different features. The first feature is the superpixel straddling feature from [4]. The idea behind this feature is that an object should be contained completely inside the bounding box, so, if the object is only partially in the box, it is a bad box. Furthermore, there is an assumption that an object is composed

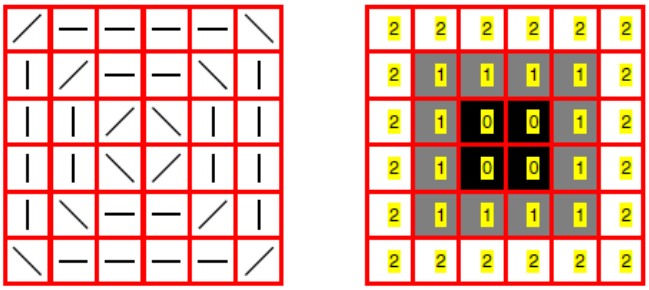


Figure 7.4: To calculate the boundary edge measure, the candidate window is broken into a 6x6 grid and each cell is assigned a weight and orientation. Left: orientations assigned to each cell. Right: weights assigned to each cell. Figure from [233]

of regions of uniform appearance and thus composed of superpixels. To calculate this feature, the image is first oversegmented in order to create a set of superpixels. Then, for each superpixel that crosses the boundary of the bounding box, we measure the degree to witch that superpixel straddles the bounding box. Formally, the superpixel straddling measure is defined by:

$$SS(B) = 1 - \sum_{s \in \mathcal{S}} \frac{\min(|s \setminus B|, |s \cap B|)}{|B|} \tag{7.11}$$

where $\mathcal{S}$ is the set of superpixels, $|s \cap B|$ is the area of the superpixel inside the bounding box and $|s \setminus B|$ is the area of the superpixel outside of the bounding box.

The second feature is similar to the superpixel straddling except that it is calculated on the super pixel bounding boxes. It is called the superpixel boundary integral and is calculated by first creating a binary image composed of all the superpixel boundary boxes overlayed on one another. In this binary image, a pixel is on if it belongs to the bounding box of some superpixel and off otherwise. That binary image is then smoothed and the boundary integral feature is calculated as the integral of the around the window being evaluated. Formally, the boundary integral is defined by:

$$BI(B) = \frac{\sum_{p \in \mathcal{B}(B)} I_s(p)}{perimeter(I_s)} \tag{7.12}$$

where I_s is the Gaussian smoothed version of the binary image and $\mathcal{B}(B)$ is the set of boundary pixels for the window being evaluated.

The third feature is the boundary edge distribution. It is calculated by first splitting the bounding box of interest into a 6x6 grid. Then, each cell y_l in that grid is assigned a weight γ_l an an orientation d_l. The orientation can be either horizontal , vertical or ± 45 degrees. The weights and orientations used for each cell are shown in fig. 7.4. The final measure is sum of the weighted gradient magnitude in the direction of interest over all of the cells. Specifically,

$$BE(B) = \frac{1}{Z} \sum_{l=1}^{L} \gamma_l \sum_{p \in y_l} G_{d_l}(p) \tag{7.13}$$

where Z is a normalization term equal to the maximum value from all the candidate windows in the image and $G_{d_l}(p)$ is the gradient magnitude in direction d_l at pixel p.

The last feature is a window symmetry feature. It breaks up the candidate bounding box into a 4x4 grid. For each grid cell, the gradient histogram is calculated using four different orientation bins. Then, a

histogram intersection is calculated for each pair of cells opposite either the vertical symmetry line or the horizontal one. Finally, these histogram intersections and summed together and normalized over all the candidate windows in the image.

The different features are combined using a weighted sum. The weights for each sum are learned from a dataset of different objects. In our case, we use the default model from the authors, which is trained on the PASCAL VOC 2007 object database. The PASCAL dataset has a number of object categories including people and thus, the detector is optimized to find these types of objects.

The objectness estimator is significantly different than the other visual utility estimators we use in this experiment. The other estimators are very simple and easy to compute, while this metric is much more complicated and thus more computationally expensive. Even so, it has been suggested in the literature the speed of an overall object detection system can still be improved with these kinds of complex features [4, 5, 233]. However, nobody has actually analyzed the final runtime as we do in this experiment.

Motion

The motion-based visual utility estimator assumes that the predominant motion in the image is from the camera motion and that any motion significantly different that this gross motion corresponds to an area of interest. In order to calculate the gross motion of the image, we calculate the affine motion between two images using the fast iterative process from [237]. On each iteration of this technique, the best known affine transform is used to project one image into the other. The difference between the two images is then taken, and then, using the optical flow equations, the affine transform is adjusted. This process is continued until convergence. Once the affine transform has been has been computed, the two images are aligned and the visual utility estimate is calculated as the distance in LAB color space. This per-pixel measure of visual utility is then converted into a per-box measure using the relative entropy technique mentioned above.

Half Resolution HOG Detector

This is one of two application specific detectors. The high-level algorithm in this experiment uses a 64x128 pixel HOG model of pedestrians trained using the INRIA pedestrian dataset [60]. In order to estimate the likelihood that this detector will fire, we simply define the visual utility estimator to be a 32x64 pixel HOG model trained on the same data and run it on an image that is half of the resolution in each dimension. By construction, this visual utility estimator approximates both the high-level detector and is tuned for the specific task at hand. This lower resolution version of the detector will execute faster because there are significantly less pixels to process and the convolution with the model is smaller. However, it will not be as accurate as the high resolution version. Note that using this visual utility estimator is very similar to the coarse-to-fine approach from [222] with only one level. The only difference is that the coarse-to-fine approach uses local, adaptive thresholds in the visual utility estimator instead of a global one and only one entry is passed on from each local region, similar to non-maximal suppression.

Haar Cascade

The last visual utility estimator is a Viola-Jones cascade [298] of Haar features trained on pedestrian images from the CBCL dataset [161, 218]. It is an application specific detector because it is trained to find people, however, it is trained on a different dataset than the one used in the high-level HOG detector and so is not specific to the high-level algorithm being used. In order to generate different processing speeds, we truncate the cascade at different stages. The longer the cascade, the less candidate locations will be passed to the high-level algorithm.

Resample Boxes

This control is a model for trying to speed up a detector without the use of visual utility. We use a model in order to be able to draw some analytical conclusions in section 7.3.2.

Instead of sampling the bounding boxes for the HOG detector in the image at our default rate of every 8 pixels and 10% increase in scale, we can sample more sparsely. This will speed up the processing because

less locations are evaluated, but there is no visual utility estimator to calculate first. The resampled bounding boxes will have the same distribution throughout the frame as the original set. Therefore, $P(b|b \in \Omega) = P(b|b \in H)$ where Ω is the set of initial bounding boxes, H is the set of sampled bounding boxes and b is true if it contains a person and is identified by the HOG detector. Therefore, the precision should stay the same.

Now, let us examine the recall. Recall cannot go up, while as more boxes are filtered out, the probability increases that all of the positives boxes associated for a given person-frame are dropped. In particular, if we make the assumption that the positives boxes for a given person-frame i are clustered around a point in the scale-location space, then as long as the new sampling period is smaller than the extent of the examples, the person will still be found. In other words, if the number of samples is reduced by a factor of k so that $|H| = |\Omega|/k$, then for person-frame i with n_i positive, the person will still be found if $n_i \geq k$, but if $n_i < k$, the probability of the person still being found is reduced to n_i/k. This allows us to compute the expected recall of the resampled boxes to be:

$$E[\text{Recall}] = \frac{\sum_{i \in \wp} \min(\frac{n_i}{k}, 1)}{|\wp|} \tag{7.14}$$

where $\wp$ is the set of all person-frames in the sequence of interest. Furthermore, the variance of the recall can be determined by first looking at the variance of finding each person-frame i. If $n_i \geq k$, then the person will always be found and thus the variance is 0. If $n_i < k$, then the variance is given by:

$$\begin{aligned} \text{var}[i] &= P_1(1 - E[i])^2 + P_0(0 - E[i])^2 \\ &= \frac{n_i}{k}\left(1 - \frac{n_i}{k}\right)^2 + \left(1 - \frac{n_i}{k}\right)\left(-\frac{n_i}{k}\right)^2 \\ &= \frac{n_i}{k} - \left(\frac{n_i}{k}\right)^2 \end{aligned} \tag{7.15}$$

Now, since the each person-frame is independent, the covariances are 0 and the variance in the number of people found is:

$$\text{var}[\text{nFound}] = \sum_{i \in \wp} \frac{n_i}{k} - \left(\frac{n_i}{k}\right)^2 \tag{7.16}$$

and the variance of the recall is:

$$\text{var}[\text{Recall}] = \frac{\sum_{i \in \wp} \frac{n_i}{k} - \left(\frac{n_i}{k}\right)^2}{|\wp|} \tag{7.17}$$

7.3.1.2 Results

In order to examine the results of this experiments, we must first look at the computation time of the various visual utility estimators. These are detailed in table 7.1. As we can see, executing the HOG detector on an entire frame takes a significant amount of time when computed naively, while computing the various visual utility estimators range from milliseconds for the Haar Cascade to many seconds for the Objectness metric. We include the timing for the HOG detector with and without caching of the HOG cells. If the HOG features are extracted densely throughout the image, then the HOG cells will be used by multiple HOG evaluations. Therefore, by caching the cell response, we can achieve a significant speedup (˜6x) while still evaluating the identical function and thus having no impact on performance. All of the following results for the visual utility filters are measured using the cached implementation.

Dataset	HIMA	ETH
HOG Detector (No Caching)	13.65 ± 1.85	3.99 ± 0.03
HOG Detector (With Caching)	2.07 ± 0.4	0.63 ± 0.01
Spectral Residual Saliency	0.26 ± 0.03	0.200 ± 0.013
Laplacian	0.054 ± 0.004	0.05 ± 0.001
Center Surround Histograms	1.00 ± 0.02	0.881 ± 0.011
Objectness	11.6 ± 2.6	4.45 ± 0.55
Motion	1.13 ± 0.56	0.62 ± 0.17
Half Resolution HOG Detector	0.73 ± 0.04	0.238 ± 0.016
Haar Cascade	0.074 ± 0.006	0.027 ± 0.003
Resample Boxes	0	0

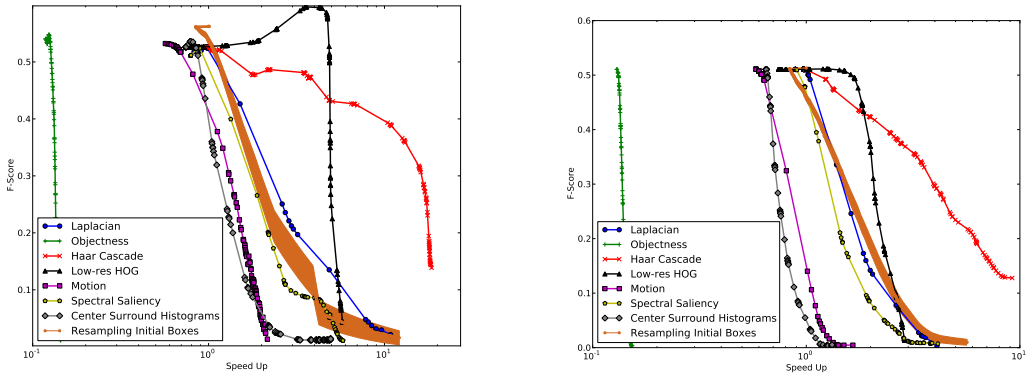
Table 7.1: The computation time in seconds of running the visual utility estimators on a full frame in the default configuration. The bounds specified are standard deviations.

Looking at table 7.1, we can quickly see that using objectness as a visual utility estimator will never speed up the overall performance of the detector because computing objectness is more computationally expensive than the HOG detector. Furthermore, there are limited improvements in speedup possible when using the motion or center surround histogram estimators because their computation time is very close to that of the HOG detector.

Now, for the other visual utility estimators, as we tighten the threshold on the visual utility estimator, the overall process can speed up as the HOG detector must be evaluated on fewer windows. However, at the same time, we are also potentially filtering out some of the boxes that the HOG detector would fire on. Figures 7.5 to 7.7 shows the result of this trade-off on both the HIMA and ETH datasets. In these figures, let us start by looking at the f-score because it is a combined metric. For a given speedup, none of the generic visual utility measures perform better than the simple approach of resampling the set of candidate boxes. However, the two task specific metrics (Haar cascade and lower resolution HOG detector) do perform more effectively than the naive strategy. The lower resolution HOG detector shows almost identical performance to the original detector for lower levels of speedup, however, it is much more expensive to compute than the Haar cascade and is thus limited to maximum speedup of around 2.5x. Therefore, when the speedup approaches this limit, the f-score drops below that of the resampling strategy. On the other hand, the Haar cascade shows some f-score degradation throughout, but can achieve a much higher potential speedup as it is so fast to compute. Furthermore, the f-score degradation occurs at a much slower rate than the resampling strategy, meaning that if a user is willing to sacrifice some task performance for significant speedups, this approach is reasonable.

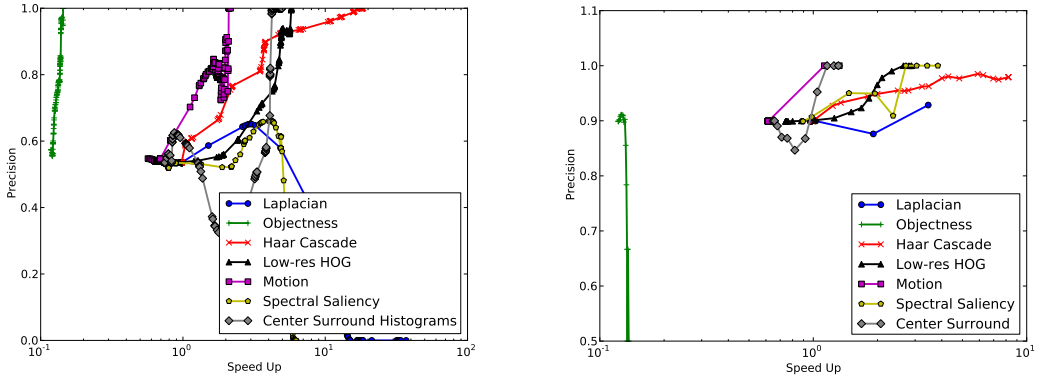
Recall that when using visual utility, it is theoretically possible to see an improvement in performance as false positives are removed. We see this in the f-score results for the HIMA dataset for the lower resolution HOG detector, but that is the only instance of this phenomenon. Let us examine the precision (fig. 7.6) and recall (fig. 7.7) in more detail to see why that might be occurring. First, note that the baseline precision is quite high, while the baseline recall is fairly low. The f-score is the harmonic mean of these two values. The harmonic mean naturally becomes dominated by the lower value if two values are significantly different. As a result, we see that the f-score is dominated by the recall, which cannot increase. If we look at the precision graph, we do in fact see that precision does increase for most of the visual utility estimators, which, according to eq. (7.5), shows that candidate boxes are being removed more effectively than random.

Though precision increases as a tighter visual utility threshold is used, recall is also decreasing. If we are only interested in the performance of the resulting system and not the runtime, this effect can also be achieved simply by raising the threshold on the full resolution HOG detector. We can compare these two approaches by examining the precision-recall curves in fig. 7.8. Note that the visual utility approaches cannot achieve a recall rate higher than the baseline used for the HOG detector. In this graph, we see



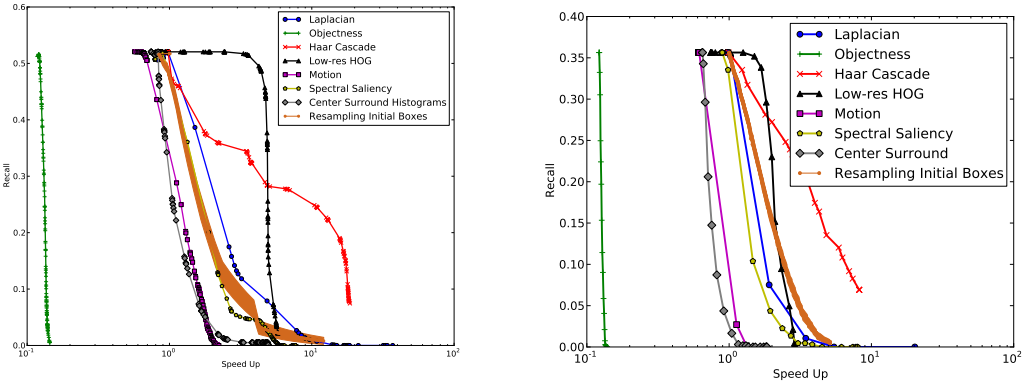
(a) HIMA (b) ETH

Figure 7.5: F-Score vs. Speedup when using visual utility with the HOG detector.



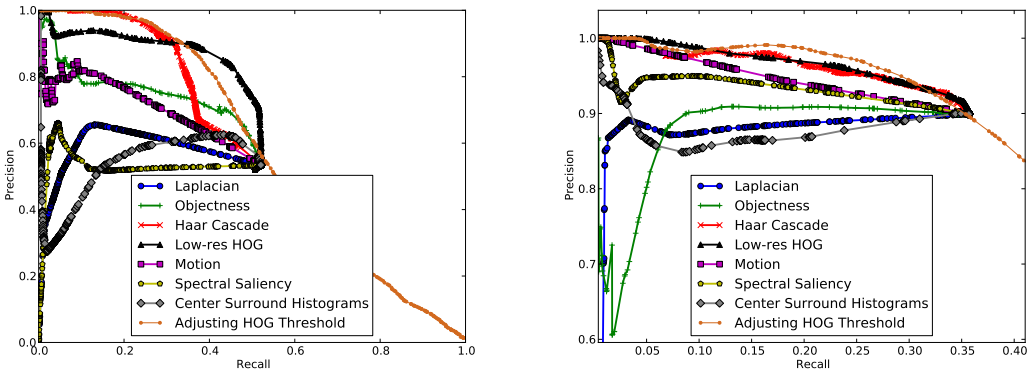
(a) HIMA (b) ETH

Figure 7.6: Precision vs. Speedup when using visual utility with the HOG detector.



(a) HIMA (b) ETH

Figure 7.7: Recall vs. Speedup when using visual utility with the HOG detector.



(a) HIMA

(b) ETH

Figure 7.8: The precision recall curves created by changing the threshold of the visual utility estimators based on a set point for the HOG detector of 0. As a baseline, the precision recall curve created by changing the threshold on the HOG detector is also shown.

that as the visual utility uses a tighter threshold, the set point moves to the left. All of the generic visual utility estimators perform more poorly than the baseline approach of adjusting the HOG threshold. The lower resolution HOG detector shows no significant change on the ETH dataset and a small improvement on the HIMA dataset. This is consistent with other work that has shown an improvement in performance when incorporating HOG features at multiple resolutions in a classifier [222]. Finally, the Haar cascade shows a small decrease on the HIMA dataset and a small increase on the ETH dataset.

These results can also be seen in table 7.2, which shows the change in average precision compared to the baseline over the valid recall ranges for each of the visual utility estimators. Table 7.2 also shows the average precision of the visual utility estimator for finding those boxes that are useful to the higher level algorithm. In other words, those that find a person and that the HOG detector will fire on. Most of these average precisions are very low, meaning that many extra boxes are sent to the full resolution HOG detector for a given recall point. As a result, we see a correlation between the average precision of the visual utility estimator and the change in average precision of the overall system. However, it is a correlation and not a direct relationship because the stream resulting from the visual utility filter is also filtered by the HOG detector, which operates in a high precision regime. Therefore, most of the false positive mistakes made in the visual utility filter are caught by the higher level algorithm, while false negative mistakes will persist. For example, the half resolution HOG detector has an average precision over 10 times higher than the Haar Cascade, but the resulting system performance is only slightly better.

Based on this analysis, we can conclude that if an appropriate, task specific visual utility estimator is used, we can use a visual utility filter to move up and to the left along the precision-recall curve, while reducing the computational requirements.

7.3.2 Applying Multiple Object Detectors

In our analysis thus far, we have used the cached and optimized version of the HOG detector as the high-level algorithm. This is already a relatively fast algorithm and so pre-filtering a candidate window only saves a small amount of time. However, this is only evaluating a single HOG model. There are two situations where we would want to evaluate multiple HOG models at each candidate location, thus creating a more expensive visual utility estimator. In the first case, we might want to find not just a single type of object in a scene, but actually understand the whole scene. This involves finding many different kinds of object simultaneously and thus different HOG models would be needed for each object. In the second

Metric	VU AP		Overall ΔAP	
Dataset	HIMA	ETH	HIMA	ETH
Spectral Residual Saliency	8.0×10^{-4}	2.4×10^{-3}	64%	97%
Laplacian	6.3×10^{-4}	1.9×10^{-3}	71%	92%
Center Surround Histograms	5.2×10^{-4}	1.0×10^{-3}	65%	91%
Objectness	1.3×10^{-3}	2.2×10^{-3}	87%	92%
Motion	2.4×10^{-3}	4.4×10^{-3}	82%	95%
Half Resolution HOG Detector	0.33	0.46	102%	99%
Haar Cascade	0.016	0.062	96%	99%

Table 7.2: VU AP: Average precision of the visual utility estimator for identifying those boxes that are useful to the full resolution HOG detector. In other words, those that overlap a person and the HOG detector will fire on. Overall ΔAP: Percent of the average precision over the valid range in the overall system PR curve (fig. 7.8) compared to the baseline.

scenario, we would want to evaluate different HOG models for different visual subcategories for the object type. For instance, one subcategory could be a pedestrian from the side and the other from the front. A detailed analysis of how to create such a combined detector can be found in [72], but for typical objects, using on the order of 20 subcategories is valuable.

In the first scenario, we could imagine using a generic visual utility estimator to filter the set of candidate boxes. So let us run a quick thought experiment to see if these generic detectors can realistically provide a speedup in this scenario. In particular, we will examine how many HOG detectors must be evaluated at a given recall level in order to reach the speed performance of our baseline, which uniformly resamples the candidate boxes.

First, let us restrict this analysis to objects with the same aspect ratio. Though objects come in many different aspect ratios, different aspect ratios require a different set of candidate bounding boxes, which requires the visual utility estimators to be recomputed. Therefore, applying more HOG models of different aspect ratios will not improve the performance compared to the baseline.

Second, we should observe that evaluating a HOG model on a candidate window is composed of two parts. In the first part, the HOG signature is extracted. In the second part, the signature is compared to the model using a dot product. The signature can be reused in subsequent model evaluations and therefore, the cost of those evaluations stems only from the dot product, which in our experiments took 4.8ms.

Now, we would like to find the number of HOG models k for each recall level such that:

$$t_{vu} + (k-1)w_{vu}t_{dot} = t_b + (k-1)w_b t_{dot} \tag{7.18}$$

where:

t_{vu} Time to process an entire image using a single HOG detector with the visual utility filter

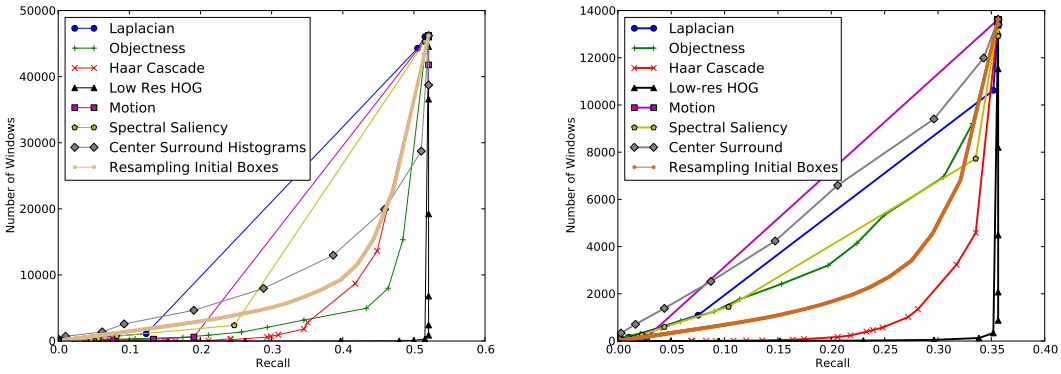
w_{vu} Number of candidate windows accepted by the visual utility filter

t_{dot} Time to perform the dot product for a single window

t_b Time to process an entire image using a single HOG detector by resampling

w_b Number of windows passed to the HOG detector(s) when resampling

Looking at this equation, we can see that there will only be a solution for k that represents a speed up if $w_{vu} < w_b$. To examine this, see fig. 7.9, which shows the number of windows passed to the high-level algorithm for different levels of the pedestrian recall and compares that to the windows needed when resampling the initial set of candidate boxes. On the ETH dataset, we can see that the only algorithms



(a) HIMA

(b) ETH

Figure 7.9: Number of windows processed by the high-level HOG detector for various overall system recall rates. For the ETH dataset, the generic visual utility estimators require more windows than the simple window resampling strategy.

that meet this criteria are the application specific visual utility estimators. However, these estimators already result in a faster system performance and furthermore, they are trained to find people so are not applicable to the generic scene understanding problem.

In contrast to the ETH dataset, the HIMA dataset shows more promise. We see that many of the generic visual utility estimators identify less windows than the resampling strategy because the scenes are not as cluttered with people as the ETH dataset. All of the generic visual utility estimators use a variation on pop-out from the surrounding. If there are a lot of people in the scene, the uniform resampling will catch people more often and a pop-out detector will fire less frequently. This results in the k values that we see in fig. 7.10. Unfortunately, for most of the generic visual utility estimators, the value of k is very high, necessitating the use of a logarithmic scale. In the best case, the spectral saliency approach breaks even between 10-100 HOG detectors, while in the worst case, the objectness measure breaks even when over 1000 HOG detectors are used. It is very rare to encounter a problem that requires evaluating 100's of detectors on a single frame. Therefore, for most problems, using the generic visual utility estimators are too expensive to compute relative to their performance gain when using a reasonably fast high-level detector.

7.3.3 SCATAT Algorithm

In this section, we will examine a set of experiments using the SCATAT algorithm. In this case, the SCATAT algorithm uses a set of integral HOG features [334]. Normally, when calculating HOG features, a histogram is generated for each block. This histogram is built for each block by binning the orientation space into 9 bins and then interpolating the component for each pixel into the two adjacent bins. Then, a gaussian spatial smoothing process is applied. Many of these blocks, which are potentially overlapping, are then combined to build the descriptor for the entire window. The feature vector is then normalized using a 2-norm [60].

In an integral HOG feature, we first build an integral histogram over the entire image. This integral histogram is exactly the same as an integral image, except that for each pixel in the image, there is an associated 9-bin histogram. For each pixel, we still interpolate between the adjacent angular bins. Once the integral histogram is built, the histogram for a block can be extracted using 4 reads:

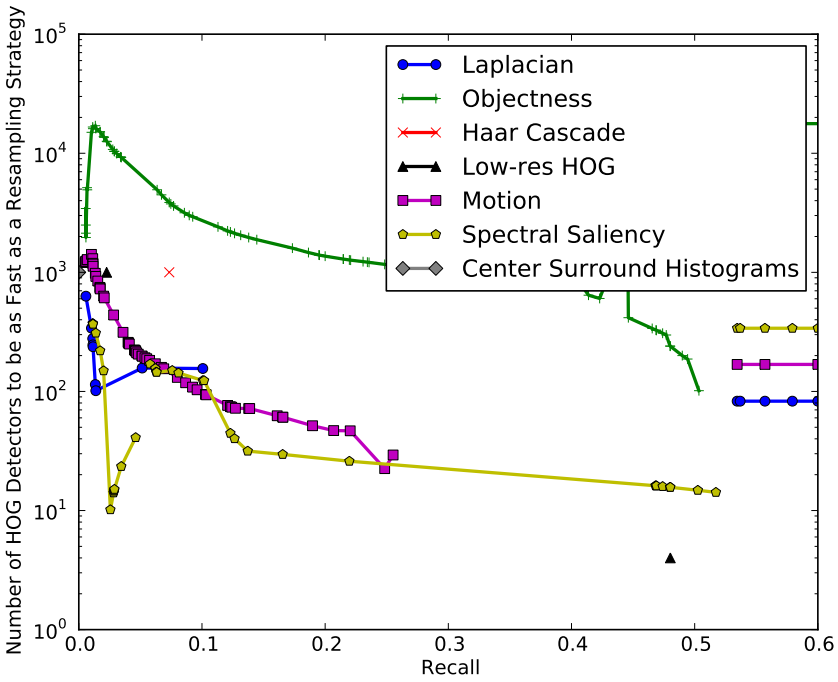


Figure 7.10: Number of high resolution HOG detectors k to evaluate on each window for the HIMA dataset when the runtime of the visual utility based approach has the same runtime as the baseline resampling strategy. Entries are not shown for those cases where the visual utility based approach is already faster or where there is no k that will make it as fast.

Process	HIMA	ETH
Generate Integral Histogram	115	42
Fill Cache for One Block Size	13	4
SVM Evaluation of One Window (min)	4×10^{-5}	4×10^{-5}
SVM Evaluation of One Window (max)	1.7×10^{-3}	1.7×10^{-3}
Max Number of Windows	46203	16326

Table 7.3: The computation time in milliseconds for each significant component when evaluating an integral HOG feature on a window.

$$h_(x_0 \leq x < x_1, y_0 \leq y < y_1, \theta) = I_h(x_1, y1, \theta) + I_h(x_0, y_0, \theta) - I_h(x_1, y_0, \theta) - I_h(x_0, y_1, \theta). \quad (7.19)$$

where h is the histogram bin of a box at an orientation θ and I_h is the integral histogram.

Each block is then normalized using the L-1 norm because the normalization factor can be calculated using an integral image. The gaussian spatial smoothing process cannot be computed efficient and is thus left out.

Once the integral histogram and integral image are calculated, evaluating a block becomes very fast and therefore, the integral HOG feature is fast to compute. However, using an L-1 norm instead of L-2 and omitting the gaussian spatial smoothing does degrade performance [60]. Therefore, we use the same approach as [334] and also examine HOG features calculated using different block sizes. In the standard scenario from Dallal & Triggs, a pedestrian window would be 64x128 pixels and blocks would be 16x16. We allow blocks to range from 16 pixels on a side up to the window size with a step size of 8. Also, we allow the HOG feature to be limited to a subwindow of the natural 64x128 window with varying widths, heights and offsets. The widths and heights range from one block to the full window size in increments of 2 blocks and the offsets start at 0 and step by block widths.

An SVM classifier for each of these integral HOG features is then trained using the INRIA pedestrian dataset. This is the same dataset used to train the high-level HOG detector. Then, when using SCATAT, we allow the threshold on the SVM classification to change. Therefore, the set of candidate filters considered by SCATAT are all possible thresholds for all of the different integral HOG features.

The processing time for each candidate filter features consists of three significant components. First, the integral histogram is calculated, which is constant for all features and must only be calculated once. Second, a cache is filled with the HOG feature value of every block for a given block size. This must be redone for each block size. Third, the SVM evaluation is performed. This evaluation is dependent on the size of the feature vector and the number of candidate windows being evaluated. The processing time for each of these stages is given in table 7.3.

In order to generate performance curves for an integral HOG cascade, we train a sequence of different cascades built using different weights of the C_M, C_F and D terms from eq. (3.7). We set the relative weights of the C_M and C_F terms to put an equal weight for false positives and misses of the entire system as detailed in section 3.2.1. Then we vary the relative cost of processing time relative to the cost of the system's accuracy in order to generate a curve.

We should note here that in practice, SCATAT requires such a detailed understanding of the components of an algorithm's runtime. This understanding is needed because the modular approximation at each point (eq. (5.8)) requires recalculating the computational cost of candidate filters at each stage in the cascade. Though this can be simulated, that approach will increase the runtime of the training significantly. It is more efficient to have a good understanding of your implementation. For example, when the first filter is added to the cascade for the integral HOG features, the integral histogram is calculated and the cache is filled for one block size. This means that the marginal computational cost of adding another filter of the same block size will only be the cost of the svm evaluation, and the marginal cost of a filter of a different

block size will be the cost to fill the cache and to perform the svm evaluation.

The effort to understand the runtime components of a given algorithm makes SCATAT expensive to implement in terms of engineering time. Furthermore, doing such an analysis will identify other improvements to an implementation that will not change the task accuracy at all. These improvements should be implemented first in most cases. Therefore, using SCATAT is most helpful in those situations where the additional speed up is worth the considerable engineering investment. For example, when computational resources are constrained, like on a Mars rover, or when a system is operating at a very large scale, like the size of the web.

Now that we have detailed the implementation of the SCATAT algorithm in order to build an integral HOG cascade, we will examine the results of two different experiments. The first experiment will compare the results of a visual utility filter built using SCATAT in comparison to the other visual utility filters we have seen in this chapter. The second experiment will compare to a filter built where processing time is not included explicitly in the cascade optimization. This is equivalent to comparing against the cascade built in [334].

7.3.3.1 Comparison With Other Visual Utility Estimators

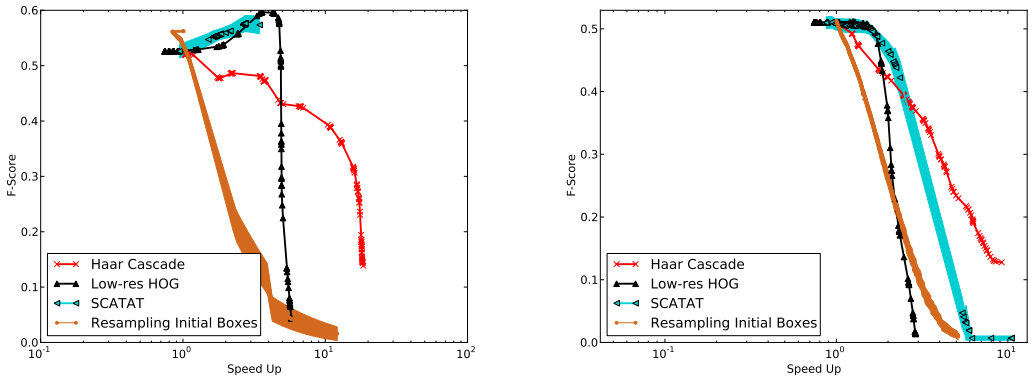
In this section, we compare the task specific visual utility estimators from section 7.3.1 with an integral HOG cascade built using SCATAT.

In each of the graphs, the standard error of the resampling baseline and the SCATAT algorithm is shown as a translucent range. This range is generated analytically for the resampled baseline. For the SCATAT algorithm is is generated using leave-one-out cross validation between with the video sequences in each of the dataset.

Figure 7.11 shows the f-score vs. speedup. Figure 7.13 shows the recall vs. speedup. Figure 7.12 shows the precision vs. speedup. Finally, fig. 7.14 shows the precision recall curve. In these graphs, we see that the integral HOG cascades generated using the SCATAT algorithm perform as expected. The precision is slightly higher than the baseline of using just the high-level HOG detector (92% vs. 90% on the ETH dataset), while the recall stays constant until around a 2x speedup and then begins to slowly degrade as recall is sacrificed for speed. At high levels of speedup, the variance becomes very large because the cascade is being trained to be as fast as possible, while accuracy is being ignored. Given that there are many different filters with nearly identical cost on the training set, the one selected is almost arbitrary. As a result, we can conclude that using the SCATAT algorithm is not appropriate in the regime of very high speedup.

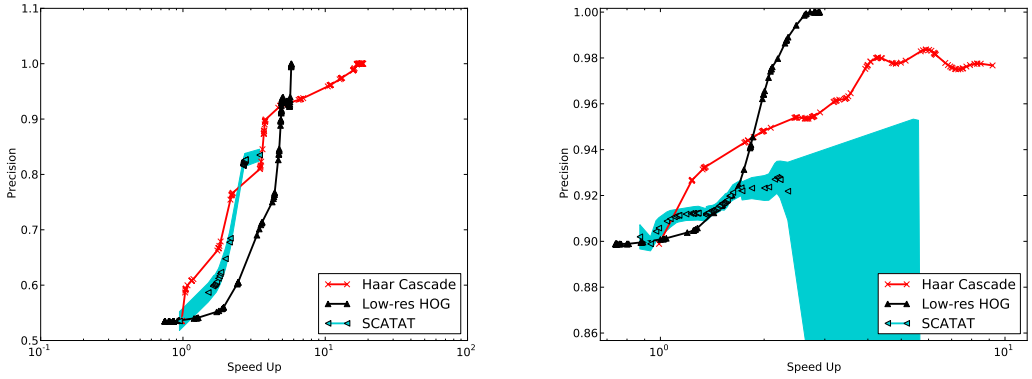
Now, let us compare the results of the integral HOG cascade built using SCATAT to the resampling baseline. It is clear from the graphs that the SCATAT approach performs significantly better than the random resampling baseline. With SCATAT, we are able to achieve a greater speedup, while sacrificing less task accuracy than the simple baseline. However, because precision stays fairly constant as speed is increasing, the precision-recall curve (fig. 7.14) shows that using SCATAT is only effective for adjusting the operating point of the detector close to the original operating point. If one wishes to make significant changes to the operating point, then it is better to change the parameters of the high-level HOG detector.

Compared to the other task specific visual utility estimators, we see that the SCATAT cascade performs similarly to the low resolution HOG detector by keeping a constant recall at low levels of speedup. This shows that both of these approaches are effectively estimating the value of the high-level HOG detector. However, SCATAT has the ability to reach a higher speedup (10x for ETH) than the low resolution HOG detector. However, the integral HOG cascade cannot reach as high of a speedup as the Haar cascade. This is unsurprising because the Haar cascade is built using features that are designed to be incredibly fast. Each Haar features can be calculated using only 6 reads on an integral image, whereas an integral HOG feature requires 36 reads for each box (9 histogram bins at the 4 corners) and there are many boxes per window. Then, a dot product between the feature vector and the SVM decision boundary must be



(a) HIMA (b) ETH

Figure 7.11: F-Score vs. Speedup for the SCATAT algorithm compared to other visual utility estimators.



(a) HIMA (b) ETH

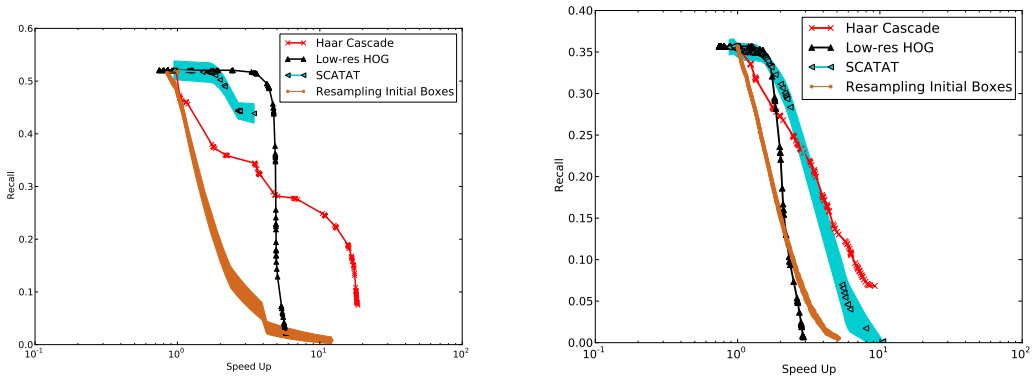
Figure 7.12: Precision vs. Speedup for the SCATAT algorithm compared to other visual utility estimators.

calculated. In contrast, the Haar feature is a simple comparison. Though the Haar cascade can achieve a higher speedup than the integral HOG cascade, it is also less accurate, so in the range where the integral HOG cascade can effectively operate, it shows much better performance.

7.3.3.2 Comparison to Other Cascade Techniques

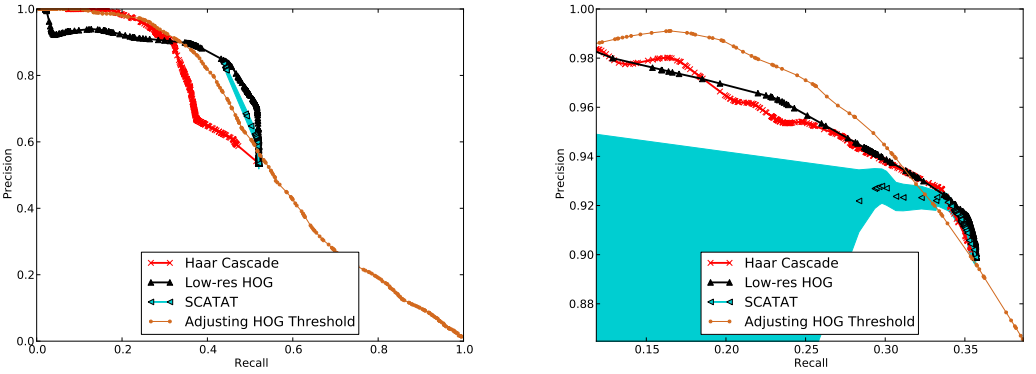
There are many other techniques to train a cascade. In section Section 4.1, we analyze the Viola-Jones [298] technique from a visual utility perspective. In this experiment, we will compare SCATAT to the cascade training algorithm from [334]. In [334], Zhu et al. introduce the integral hog cascade to speed up the HOG detector from [60] and generate a cascade. In the visual utility formulation in this thesis, their algorithm is greedily minimizing the Bayes Risk, which is just the visual utility risk without the explicit term for the computational cost from eq. (3.7), D. So, to compare their technique with SCATAT, we simply run our optimization procedure with D set to 0. Then, the various performance curves are generated by varying the cost of false positives C_F relative to the cost of a miss C_M.

Figures 7.15 to 7.18 show the performance of Zhu et al.'s cascade relative to SCATAT. The plots of performance relative to speedup demonstrate that the resulting cascade executes more slowly than the



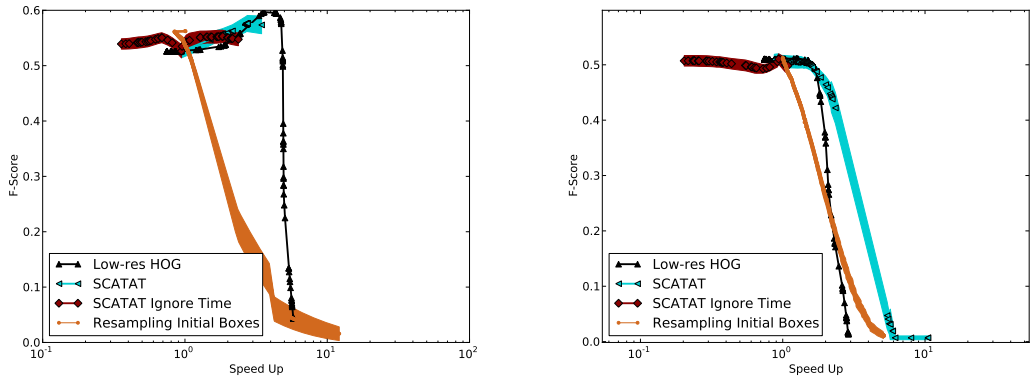
(a) HIMA (b) ETH

Figure 7.13: Recall vs. Speedup for the SCATAT algorithm compared to other visual utility estimators.



(a) HIMA (b) ETH

Figure 7.14: Precision vs. Recall when using visual utility with the HOG detector including building an integral HOG cascade using SCATAT.



(a) HIMA (b) ETH

Figure 7.15: F-Score vs. Speedup for the SCATAT algorithm compared to the integral HOG cascade from [334].

full SCATAT cascade. This observation is unsurprising given that SCATAT explicitly tries to optimize for processing time.

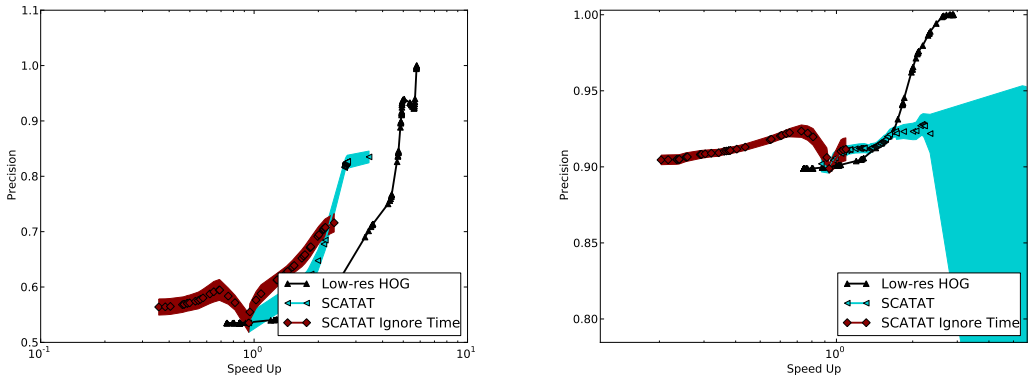
In fig. 7.15 we can also see that the time agnostic cascade can hold a constant F-score for most of its operating range. However, the full version of SCATAT will trade off F-score performance for processing speeds not reachable by the time-blind cascade. Therefore, if higher speeds are required, SCATAT is able to achieve them more effectively and in a more principled way.

7.3.4 Impact of Non-Maximal Suppression

When examining biological visual systems, Klein proved that in order to function effectively, they require an inhibition of return mechanism [154]. When the eye executes saccades around a scene, inhibition of return makes it less likely to return to locations that have been focused on previously. Without this mechanism, the eye would get stuck moving between a small set of locations and never examine new regions. In our case of pedestrian detection operating on a single frame, non-maximal suppression on the output of the visual utility estimator can act like an inhibition of return mechanism. Non-maximal suppression reduces the scores of regions around local maxima so that they are less likely to pass the visual utility filter. As a result, less frames are passed to the high-level algorithm in a particular region of the image, which has the potential to speed up processing.

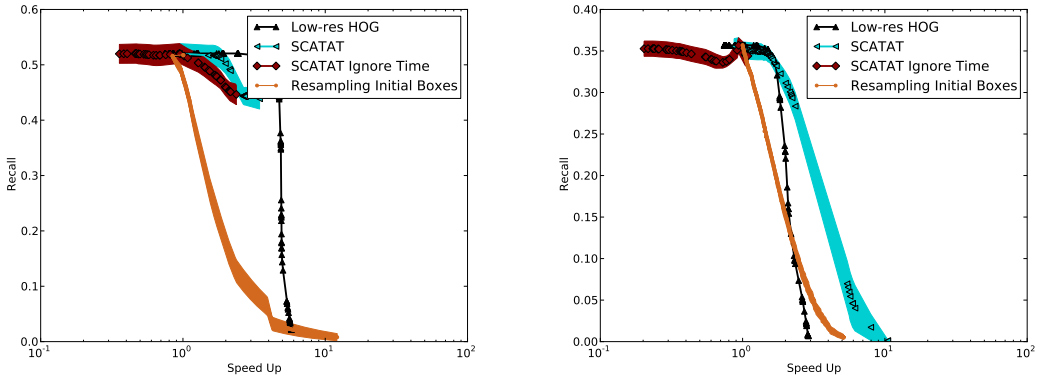
Our implementation of non-maximal suppression (alg. 3) is similar to that from [295] and [233]. We use the greedy approach because it is fast to compute and the computation time must be included in our analysis of overall system speedup. In this approach, all candidate regions are first sorted in descending order by their visual utility score. Then, this list is traversed until the visual utility threshold is hit. For each candidate region, it is accepted if it does not overlap any previously selected regions by more than 0.5 using the pascal overlap criteria explained in eq. (7.1). Otherwise, it is rejected. The resulting set of candidate regions are then passed to the high-level algorithm as before.

Figures 7.19 and 7.20 show the results of including non-maximal suppression. For simplicity, we only included the f-score vs. speedup graph and the precision-recall curve for comparison with the performance without non-maximal suppression. In both of these figures, we see a significant decrease in performance for all of the visual utility estimators section 7.3.1.1. This indicates that there are many cases where the best scoring box according to visual utility either does not fire on the HOG detector or does not overlap enough of the person, whereas one of the nearby boxes that are suppressed, does. This could be caused by an



(a) HIMA (b) ETH

Figure 7.16: Precision vs. Speedup for the SCATAT algorithm compared to the integral HOG cascade from [334].



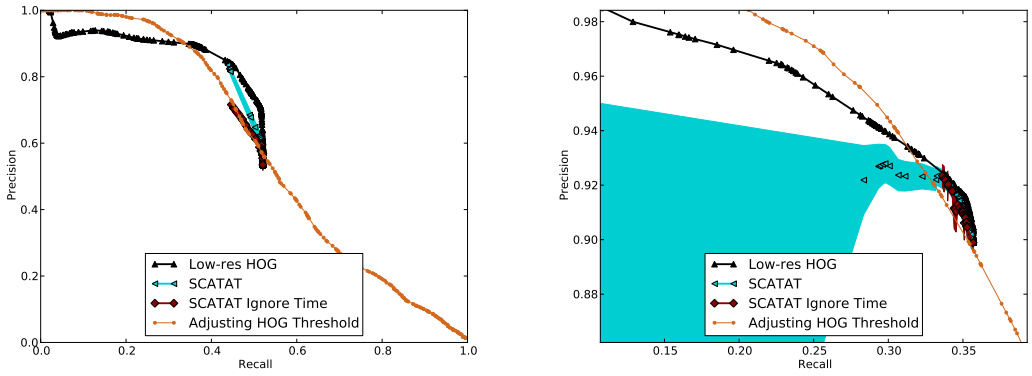
(a) HIMA (b) ETH

Figure 7.17: Recall vs. Speedup for the SCATAT algorithm compared to the integral HOG cascade from [334].

Algorithm 3 Greedy Non-Maximal Suppression

Input: Set of all candidate regions $\bar{X} = \{X_1, X_2, \cdots X_n\}$
Output: Set of regions $\bar{Y}$ with a locally maximum score

$\bar{Y} = \emptyset$
for x in sort($\bar{X}$ by VuScore(x)) **do**
  if $\max(\{|x \cap y|/|x \cup y| \forall y \in \bar{Y}\}) > 0.5$ **then**
    $\bar{Y} = \bar{Y} \cup x$
  end if
end for
return $\bar{Y}$



(a) HIMA

(b) ETH

Figure 7.18: Precision vs. Recall for the SCATAT algorithm compared to the integral HOG cascade from [334].

alignment issue in the visual utility model relative to the HOG detector, or it could simply be a result of a very noisy visual utility estimator. To determine which is the case, fig. 7.21 shows the correlation between the scores from two of the visual utility detectors, the HOG detector score and the ground truth overlap. The graphs show very weak correlations, indicating that there is a lot of noise both in the visual utility estimators and in the high-level HOG detector. Even the best correlation, between the low resolution and high resolution HOG detector is still quite weak, with the top overlapping regions spread throughout the space. Therefore, there is a reasonable probability that the local maxima in the visual utility scores does not correspond to a region that both overlaps the ground truth and will be detected by the high-level detector.

7.4 Summary

In this chapter, we have described a series of experiments exploring visual utility for pedestrian detection as being indicative of object detection in general. Crucially, this is the first study to explicitly measure the processing time of the entire visual utility system including the time to compute the visual utility estimator. We have shown that the choice of visual utility estimator has a profound impact on the potential impact of the overall visual utility system. A task specific visual utility estimator will usually perform more effectively than a generic one. Furthermore, for some applications, like the one explored in this case study, a simple approach of sampling the image space more sparsely can be surprisingly effective compared to more complex visual utility approaches. In practice, it would be reasonable to test this simple approach first to see if it meets the system requirements before applying a visual utility approach.

We have also shown that the SCATAT algorithm from chapter 5 can be used to build a cascade of integral HOG detectors that speeds up overall processing while sacrificing less task accuracy than other approaches. However, there are limits to this approach. It will fail if the user attempts to build a cascade that is too fast for the underlying features.

Though we show that task specific estimators are more effective than generic ones for finding people, we also explore the multi object problem. In this case, a generic estimator could be used to find candidate locations of objects and then many object detectors could be applied. We show that on the order of hundreds to thousands of different HOG objects must be evaluated at each location in order to break even with simply evaluating those detectors at every location in the image.

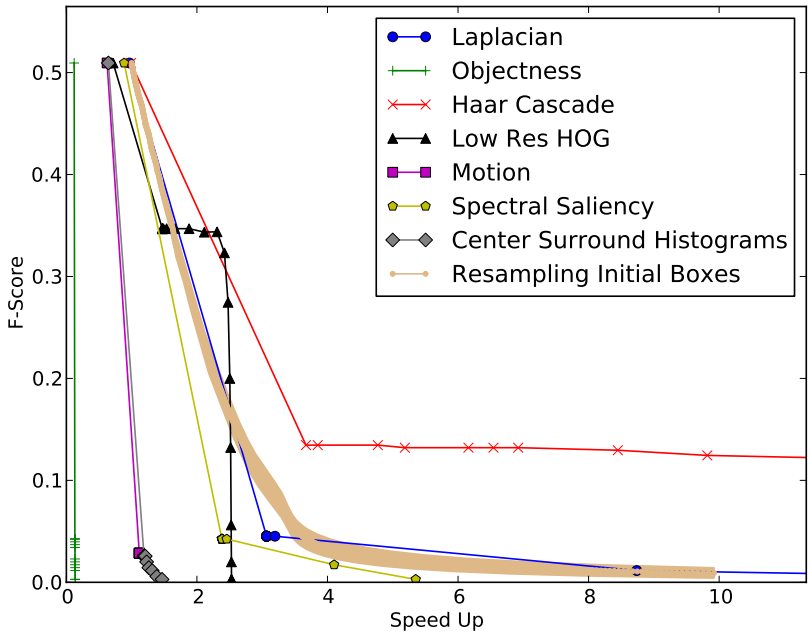


Figure 7.19: The F-score vs. speedup when using non-maximal suppression on the ETH dataset

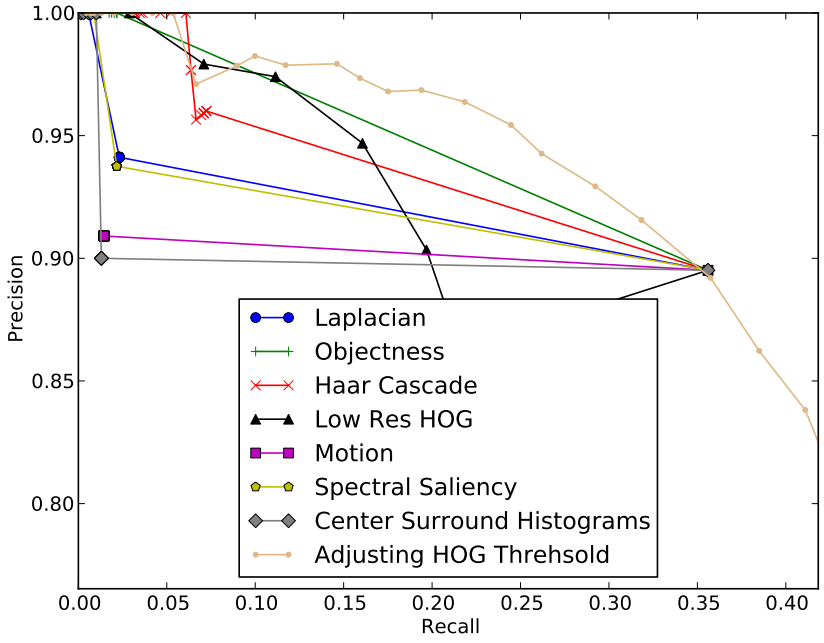
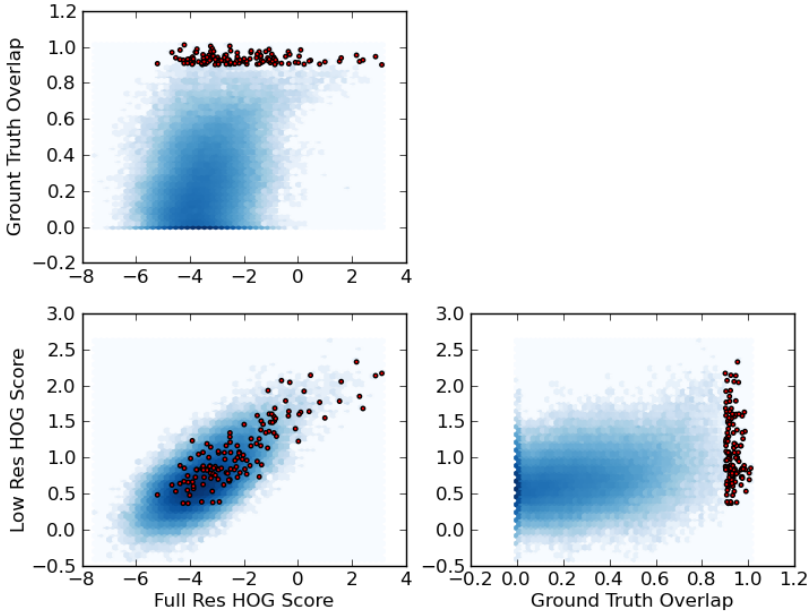
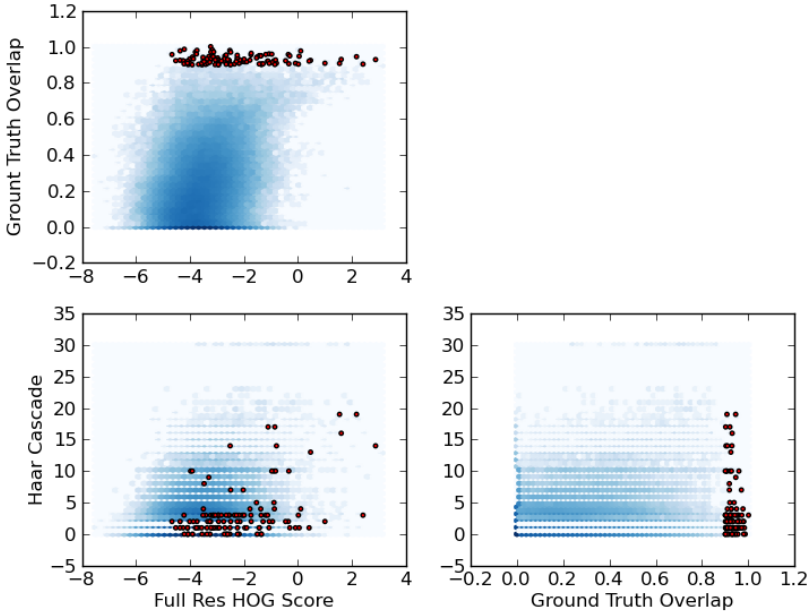


Figure 7.20: The PR curve when using non-maximal suppression on the ETH dataset



(a) Low Res HOG



(b) Haar Cascade

Figure 7.21: Correlation plots of the Haar Cascade and low res HOG visual utility estimators with the full res HOG detector and the ground truth overlap score. The darkness of the blue indicates the density in the plot, while red dots correspond to instances that have over 90% overlap with the ground truth. The data for these plots comes from the ETH dataset.

Finally, we examine the impact of non-maximal suppression of the visual utility estimator in the visual utility process. We find that using non-maximal suppression actually hurts performance and show that this is because the visual utility estimators examined in these experiments are still very noisy.

Chapter 8

Conclusions

> The light at the end of the tunnel is just the light of an oncoming train.
>
> Robert Lowell

As computer vision matures as a field, the amount of data being used in order to understand complex scenes is trending up. This data comes in three forms. First, higher resolution sensors are used in order to capture more specific information about the scene or to resolve objects further away. Second, context within images is used in order to better resolve ambiguities in the scene (e.g. [113]). For instance, a small lamp is more likely to sit on a table and thus, if a lamp is found, it is more likely that the object under it is a table (fig. 8.1). Lastly, web scale datasets are being used for many applications like identifying all the possible objects in an indoor scene (e.g. [148]), to building accurate 3D models of cities [265].

This increase in data, along with algorithms that exploit more relationships between image regions, causes state-of-the art computer vision algorithms to be very computationally expensive. Moore's law cannot counter this trend. Therefore, to be applied to real world problems, computer vision approaches must be run more efficiently. Many people have developed techniques to speed up specific vision algorithms (e.g. SURF [23] vs. SIFT [190]), but in this thesis, we have examined a class of approaches that could be used to reduce the cost of *any* computer vision algorithm. These approaches, which we call visual utility approaches, are fundamentally inspired by the biological vision system. Visual utility approaches use a fast prediction of where the useful information is in a scene in order to reduce the amount of data that a higher level algorithm must process.

In this thesis, we have shown that:

The visual utility framework can be used to reduce the computational cost of computer vision algorithms in a lossy way such that the degradation of task accuracy is minimized.

We have used the theoretical formulation of visual utility in chapter 3 in order to identify when computational cost will decrease while allowing task accuracy to degrade. We showed examples of visual utility being applied in previous work throughout our literature analysis in chapter 2. We have identified that building a cascade of visual utility estimators is a submodular process (section 3.4.3). Using that insight, we have developed SCATAT in chapter 5, an algorithm that efficiently builds a near optimal cascade and allows an explicit trade off between computational cost and task accuracy. In chapter 6, we examined different approaches to use visual utility and submodularity in deep neural networks. Finally, we have validated the SCATAT algorithm and the rest of the visual utility framework in an experimental case study on object detection in chapter 7.

Figure 8.1: A lamp is more likely to sit on a table than to float on the air or to sit on a chair. Therefore, if we find a table in a scene, we can infer that the lamp like object on top of it is more likely to be a lamp. This kind of context-based modeling can significantly improve object detection and scene understanding, but it increases the computational complexity of the algorithm.

8.1 Contributions

In this thesis, we have presented the generic visual utility framework and formally defined it. This framework can be used to analyze approaches that filter visual data before processing by a higher level algorithm or iteratively present visual samples to a higher level algorithm.

Using the formal definition of visual utility, we proved that building a cascade of visual utility filters is a submodular process. We then used this insight in order to show that two seminal cascade algorithms are near optimal if the optimization is restricted. In both of these cases, a reduced computational cost is a by-product of building the cascade. Then, we presented SCATAT, an algorithm that explicitly trades off test computational cost for task accuracy in order to efficiently build a near optimal cascade from a large set of candidate filters.

We also use our theoretical understanding of visual utility systems and submodularity in order to examine different aspects of deep neural networks and recent work in that space. We show that computational cost of deep neural networks can be reduced using both visual utility filtering and adaptive sampling with a visual utility prior. Finally, we explore how submodularity has been used to iteratively build deep neural networks.

Then we have presented an experimental case study for using visual utility for object detection. This case study is the first to consider the processing time of the entire visual utility system including the processing time for the visual utility estimator and the high-level algorithm. We have examined a variety of visual utility estimators and are the first to compare them to a simple baseline of resampling the image more sparsely. We showed that surprisingly, this simple baseline performs remarkably well compared to other generic visual utility estimators.

In another set of experiments, we showed that the SCATAT algorithm can be used to build a cascade that speeds up the overall system, with less loss in task accuracy than other visual utility estimators. Further, we show that SCATAT builds a cascade that is more effective for a given processing time than previous cascade algorithms.

Finally, we are one of the first to examine the computational cost for the case of multiple object

search. In this scenario, a generic visual utility estimator is used to find candidate regions, which are then evaluated by k different object detectors. We showed that in order to have the same computational cost as the baseline of resampling the image more sparsely, k must be on the order of hundreds to thousands.

8.2 Future Work

As we showed in our literature analysis in chapter 2, a form of visual utility has been used for many different applications. However, this is the first work to examine the full system performance on any of these tasks. We focused on object detection in our experimental work because it is the most natural application for visual utility and because it is central to much of the current vision research. However, further studies should examine the system performance of visual utility and the SCATAT algorithm on some of these other problems. Especially interesting would be those where useful regions in the image are not as clearly defined as for object detection and our formulation might need to be expanded using a metric like information gain. For instance, when mapmaking, all of the information acquired is useful to some small extent, but the first picture of a new region contains significantly more information than a later one. Examples of these problems include:

- Mapmaking
- Surveillance
- Simultaneous Localization and Mapping (SLAM)
- Image Search
- Object Discovery

Next, our formulation has been limited to evaluation on a per-frame basis. It would be valuable to expand our visual utility framework to account for tasks operating over time. This will require keeping a memory of recent visual utility, which must be efficiently merged and tracked over time.

Next, as described in chapter 6, the re-emergence of neural networks in computer vision opens new frontiers for visual utility. The framework defined in this thesis, the SCATAT algorithm and submodularity could be could help practitioners trade off performance and computational cost in this brave new world. Furthermore, there is limited theoretical understanding of how deep neural networks are trained and the functions they encode. Therefore, only a few submodular guarantees can be currently defined. Future work could expand our fundamental understanding of deep neural networks and unify that theory with submodularity. This unification could help to define the structure and uses of these networks to be both more efficient and more effective.

Finally, there are some tasks where visual utility has not been used yet, but might have some benefit. Most likely, visual utility not been used most because they are very hard vision problems, thus, the focus of research at this time is to find solutions to these problems without worrying about computational cost. For instance, for both pose detection and scene understanding, the relationship between discovered objects is used to both reason about the image and to aid in the underlying detection. However, capturing the value of these relationships has been difficult and thus most work has been focused on how best to capture the value from all the information in the image. Visual utility could be used to limit that information to only those portions of the image that are informative, however, we know of no work that has done so thus far.

8.3 Parting Thoughts

Computer vision is a very appealing solution to many sensing problems in the world because it is intuitive for us humans to see the world in order to understand it. However, as anybody who has worked in

computer vision can attest to, teaching a computer to see is not as simple a task as it appears to be. When applied to real world scenarios, many algorithms fail to live up to their reported performance and/or are notoriously computationally expensive. It is my hope that this thesis will help to make computer vision a more practical solution for some of these problems.

Chapter 9

Bibliography

[1] A. L Abbott and N. Ahuja. Active surface reconstruction by integrating focus, vergence, stereo, and camera calibration. In *Computer Vision, 1990. Proceedings, Third International Conference on*, pages 489–492, 1990.

[2] R. Achanta, F. Estrada, P. Wils, and S. S\sstrunk. Salient region detection and segmentation. *Computer Vision Systems*, pages 66–75, 2008.

[3] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. Frequency-tuned salient region detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1597–1604, 2009.

[4] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 73–80, 2010.

[5] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the Objectness of Image Windows. *IEEE transactions on pattern analysis and machine intelligence*, 2012.

[6] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1(4):333–356, 1988.

[7] M. K Alsmadi, K. B. Omar, S. A NOAH, and I. ALMARASHDEH. Fish recognition based on robust features extraction from color texture measurements using back-propagation classifier. *J. Theor. Applied Inform. Technol*, 6:1088–1094, 2010.

[8] S. An, P. Peursum, W. Liu, and S. Venkatesh. Efficient subwindow search with submodular score functions. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1409–1416, 2011.

[9] Martin Anthony. Margin-based generalization error bounds for threshold decision lists. *JOURNAL OF MACHINE LEARNING RESEARCH*, 5:189–217, 2003.

[10] N. E. Apostoloff and A. Zisserman. Who are you? real-time person identification. In *Proceedings of the 18th British Machine Vision Conference, Warwick, UK*, pages 509–518, 2007.

[11] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. *arXiv preprint arXiv:1310.6343*, 2013.

[12] F. Arrebola, C. Urdales, P. Camacho, and F. Sandoval. Vision system based on shifted fovea multiresolution retinotopologies. In *Industrial Electronics Society, 1998. IECON'98. Proceedings of the 24th Annual Conference of the IEEE*, volume 3, pages 1357–1361, 1998.

[13] T. Avraham and M. Lindenbaum. Esaliency (extended saliency): Meaningful attention using stochastic image modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 693–708, 2009.

[14] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in Neural*

Information Processing Systems, pages 2654–2662, 2014.

[15] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple Object Recognition with Visual Attention. *arXiv:1412.7755 [cs]*, December 2014. arXiv: 1412.7755.

[16] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

[17] D. H Ballard. Animate vision. *Artificial intelligence*, 48(1):57–86, 1991.

[18] S. Baluja and D. Pomerleau. Dynamic relevance: Vision-based focus of attention using artificial neural networks. *Artificial Intelligence*, 97(1-2):381–395, 1997.

[19] Cesar Bandera. *Foveal Machine Vision Systems*. PhD thesis, University of Buffalo, 1990.

[20] J. E Banta, Y. Zhien, X. Z. Wang, G. Zhang, M. T. Smith, and M. A Abidi. A best-next-view algorithm for three-dimensional scene reconstruction using range images. In *Proc. SPIE*, volume 2588, pages 418–429, 1995.

[21] A. Basu and K. Wiebe. Enhancing videoconferencing using spatially varying sensing. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 28(2):137–148, 1998.

[22] J. Batista, P. Peixoto, and H. Araujo. Real-time active visual surveillance by integrating peripheral motion detection with foveated tracking. In *Visual Surveillance, 1998. Proceedings., 1998 IEEE Workshop on*, pages 18–25, 1998.

[23] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[24] Yoshua Bengio. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2 (1):1–127, 2009.

[25] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, and others. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[26] B. Benson, J. Cho, D. Goshorn, and R. Kastner. Field Programmable Gate Array (FPGA) Based Fish Detection Using Haar Classifiers. In *American Association of Underwater Sciences Symposium*, 2009.

[27] A. Bernardino and J. Santos-Victor. A binocular stereo algorithm for log-polar foveated systems. In *Biologically Motivated Computer Vision*, pages 119–134, 2010.

[28] F. Berton, G. Sandini, and G. Metta. Anthropomorphic visual sensors. *Encyclopedia of Sensors*, 10: 1–16, 2006.

[29] O. Boiman and M. Irani. Detecting irregularities in images and in video. *International Journal of Computer Vision*, 74(1):17–31, 2007.

[30] Ali Borji and Laurent Itti. State-of-the-art in visual attention modeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2012.

[31] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 236–243, 2005.

[32] C. Breazeal and B. Scassellati. A context-dependent attention system for a social robot. In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pages 1146–1151, 1999.

[33] S. Brubaker, M. Mullin, and J. Rehg. Towards optimal training of cascaded detectors. *Computer VisionECCV 2006*, pages 325–337, 2006.

[34] N. Bruce and J. Tsotsos. Saliency based on information maximization. *Advances in neural information processing systems*, 18:155, 2006.

[35] Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 649–658, 2012.

[36] W. Burgard, M. Moors, C. Stachniss, and F. E Schneider. Coordinated multi-robot exploration. *Robotics, IEEE Transactions on*, 21(3):376–386, 2005.

[37] P. J Burt. Smart sensing within a pyramid vision machine. *Proceedings of the IEEE*, 76(8):1006–1015, 1988.

[38] N. J Butko, L. Zhang, G. W Cottrell, and J. R Movellan. Visual saliency model for robot cameras. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2398–2403, 2008.

[39] S. Cadieux, F. Lalonde, and F. Michaud. Intelligent system for automated fish sorting and counting. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2, pages 1279–1284, 2000.

[40] A. Califano, R. Kjeldsen, and R. M Bolle. Data and model driven foveation. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume 1, pages 1–7, 1989.

[41] P. Camacho, F. Coslado, M. Gonzlez, and F. Sandoval. Multifoveal imager for stereo applications. *International journal of imaging systems and technology*, 12(4):149–165, 2002.

[42] S. A Campbell. *The science and engineering of microelectronic fabrication*, volume 1. Oxford University Press, 2001.

[43] C. Capurro, F. Panerai, and G. Sandini. Dynamic vergence using log-polar images. *International Journal of Computer Vision*, 24(1):79–94, 1997.

[44] M. Carrasco, P. E Williams, and Y. Yeshurun. Covert attention increases spatial resolution with or without masks: Support for signal enhancement. *Journal of Vision*, 2(6), 2002.

[45] Miguel A. Carreira-Perpinan and Geoffrey E. Hinton. On contrastive divergence learning. In *Proceedings of the tenth international workshop on artificial intelligence and statistics*, pages 33–40. Citeseer, 2005.

[46] A. R Cassandra, L. P Kaelbling, and J. A Kurien. Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 2, pages 963–972, 1996.

[47] M. Cerf, J. Harel, W. Einh\user, and C. Koch. Predicting human gaze using low-level saliency combined with face detection. *Advances in neural information processing systems*, 20:241–248, 2008.

[48] T. Chen, M. M Cheng, P. Tan, A. Shamir, and S. M Hu. Sketch2photo: internet image montage. *ACM Transactions on Graphics (TOG)*, 28(5):1–10, 2009.

[49] X. Chen and A. L. Yuille. A time-efficient cascade for real-time object detection: With applications for the visually impaired. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pages 28–28, 2005.

[50] M. M Cheng, G. X Zhang, N. J Mitra, X. Huang, and S. M Hu. Global contrast based salient region detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 409–416, 2011.

[51] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proc. ICCV*, volume 2, 2007.

[52] J. J Clark. Spatial attention and saccadic camera motion. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 3247–3252, 1998.

[53] J. J Clark and N. J Ferrier. Modal control of an attentive vision system. In *Proceedings of the International Conference on Computer Vision*, pages 514–523, 1988.

[54] J. J Clark, Z. M Hafed, and L. Jie. Attention and action. *Computational vision in neural and machine systems*, pages 129–148, 2007.

[55] C. Colombo, M. Rucci, and P. Dario. Integrating selective attention and space-variant sensing in machine vision. *Image Technology: Advances in Image Processing, Multimedia and Machine Vision*,

page 109, 1996.

[56] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[57] C. Connolly. The determination of next best views. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 432–435, 1985.

[58] C. J Costello, C. P Diehl, A. Banerjee, and H. Fisher. Scheduling an active camera to observe people. In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 39–45, 2004.

[59] J. L Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 674–680, 1989.

[60] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, page 886, 2005.

[61] K. Daniilidis, C. Krauss, M. Hansen, and G. Sommer. Real-time tracking of moving objects with an active camera. *Real Time Imaging*, 4:3–20, 1998.

[62] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*, 2011.

[63] John G. Daugman. Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(7):1169–1179, 1988.

[64] A. J Davison, I. D Reid, N. D Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052, 2007.

[65] Brian C. Dean, Michel X. Goemans, and Jan Vondrk. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

[66] Thomas Dean, Mark A. Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, Accurate Detection of 100,000 Object Classes on a Single Machine. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[67] A. Del Bimbo and F. Pernici. Distant targets identification as an on-line dynamic vehicle routing problem using an active-zooming camera. In *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 97–104, 2005.

[68] J.R. del Solar, C Nowack, and B Schneider. VIPOL: A virtual polar-logarithmic sensor. In *Proceedings of Scandinavian Conference on Image Analysis*, pages 739–744, Finland, 1997.

[69] R. Desimone and J. Duncan. Neural mechanisms of selective visual attention. *Annual review of neuroscience*, 18(1):193–222, 1995.

[70] C. Dima. *Active learning for outdoor perception.* PhD thesis, Carnegie Mellon University, 2006.

[71] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, 2001.

[72] S. K. Divvala, A. A. Efros, and M. Hebert. How important are Deformable Parts in the Deformable Parts Model? *Arxiv preprint arXiv:1206.3714*, 2012.

[73] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics and Automation Magazine*, 13(2):99–110, 2006.

[74] S. Dutta Roy, S. Chaudhury, and S. Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446, 2004.

[75] D. R Edgington, D. E Cline, D. Davis, I. Kerkez, and J. Mariette. Detecting, tracking and classifying animals in underwater video. In *OCEANS*, pages 1–5, 2006.

[76] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[77] I. Endres and D. Hoiem. Category independent object proposals. *Computer VisionECCV 2010*, pages 575–588, 2010.

[78] U. M Erdem and S. Sclaroff. Look there! predicting where to look for motion in an active camera network. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.*, pages 105–110, 2005.

[79] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2155–2162. IEEE, 2014.

[80] A. Ess, K. Schindler, B. Leibe, and L. Van Gool. Object detection and tracking for autonomous navigation in dynamic environments. *The International Journal of Robotics Research*, 29(14):1707–1725, 2010.

[81] M. Everingham, L. Gool, C.K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[82] N. Fairfield, G. Kantor, and D. Wettergreen. Real-Time SLAM with Octree Evidence Grids for Exploration in Underwater Tunnels. *Journal of Field Robotics*, 24(1-2):3, 2007.

[83] P. Felzenszwalb, D. McAllester, and D. Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008.

[84] P. F Felzenszwalb, R. B Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2241–2248, 2010.

[85] Cornelia Fermuller and Yiannis Aloimonos. The role of fixation in visual motion analysis. *International Journal of Computer Vision*, 11(2):165–186, October 1993.

[86] F. Ferrari, J. Nielsen, P. Questa, and G. Sandini. Space variant imaging. *Sensor Review*, 15(2): 18–20, 1995.

[87] Nicola J. Ferrier and James J. Clark. The Harvard Binocular Head. *International Journal of Pattern Recognition and Artificial Intelligence*, 07(01):9, 1993.

[88] S. Firestone, T. Ramalingam, and S. Fry. *Voice and video conferencing fundamentals*. Cisco Press, 2007. ISBN 1-58705-268-7.

[89] T. Fisher and R. Juday. A programmable video image remapper. *Digital and optical shape representation and pattern recognition*, pages 122–128, 1988.

[90] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325, 2006.

[91] S. Frintrop, P. Jensfelt, and H. I Christensen. Attentional landmark selection for visual slam. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2582–2587, 2006.

[92] S. Frintrop, M. Klodt, and E. Rome. A real-time visual attention system using integral images. In *Proceedings of the 5th international conference on computer vision systems*, 2007.

[93] Satoru Fujishige, Takumi Hayashi, and Shigueo Isotani. The Minimum-Norm-Point Algorithm Applied to Submodular Function Minimization and. In *Kyoto University, Kyoto Japan*, 2006.

[94] D. Gao and N. Vasconcelos. Discriminant saliency for visual recognition from cluttered scenes. *Advances in neural information processing systems*, 17:481–488, 2005.

[95] D. Gao, V. Mahadevan, and N. Vasconcelos. On the plausibility of the discriminant center-surround hypothesis for visual saliency. *Journal of Vision*, 8(7), 2008.

[96] Ross Girshick. Fast R-CNN. *arXiv:1504.08083 [cs]*, April 2015. arXiv: 1504.08083.

[97] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[98] S. Goferman, L. Zelnik-Manor, and A. Tal. Context-aware saliency detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2376–2383, 2010.

[99] Daniel Golovin and Andreas Krause. Adaptive Submodularity: A New Approach to Active Learning and Stochastic Optimization. In *COLT*, pages 333–345, 2010.

[100] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, pages 427–486, 2011.

[101] H. Gonzlez-Ba\ nos, E. Mao, J. C Latombe, T. M. Murali, and A. Efrat. Planning robot motion strategies for efficient model construction. In *Robotics Research International Symposium*, volume 9, pages 345–352, 2000.

[102] S. G Goodridge and M. G Kay. Multimedia sensor fusion for intelligent camera control. In *Multisensor Fusion and Integration for Intelligent Systems, IEEE/SICE/RSJ International Conference on*, pages 655–662, 1996.

[103] S. Gould, J. Arfvidsson, A. Kaehler, B. Sapp, M. Meissner, G. Bradski, P. Baumstarck, S. Chung, and A. Y Ng. Peripheral-foveal vision for real-time object recognition and tracking in video. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.

[104] G. Grisettiyz, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437, 2005.

[105] A. Grubb and J. A. Bagnell. Speedboost: Anytime prediction with uniform near-optimality. In *AISTATS*, volume 15, pages 458–466, 2012.

[106] M. Grtschel, L. Lovsz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[107] C. Guo and L. Zhang. A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression. *Image Processing, IEEE Transactions on*, 19(1):185–198, 2009.

[108] J. Han, K. N Ngan, M. Li, and H. J Zhang. Unsupervised extraction of visual attention objects in color images. *Circuits and Systems for Video Technology, IEEE Transactions on*, 16(1):141–145, 2006.

[109] B. Hanczar, J. Hua, C. Sima, J. Weinstein, M. Bittner, and E.R. Dougherty. Small-sample precision of ROC-related estimates. *Bioinformatics*, 26(6):822–830, 2010.

[110] D.J. Hand. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine learning*, 77(1):103–123, 2009.

[111] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. *Advances in neural information processing systems*, 19:545, 2007.

[112] Hedi Harzallah, Fr\'ed\'eric Jurie, and Cordelia Schmid. Combining efficient object localization and image classification. In *International Conference on Computer Vision*, September 2009.

[113] James Hayes. An Empirical Study of Context in Object Detection. In *Computer Vision and Pattern Recognition (CVPR), 2009 IEEE Conference on*, 2009.

[114] M. M Hayhoe, A. Shrivastava, R. Mruczek, and J. B Pelz. Visual memory and motor planning in a natural task. *Journal of Vision*, 3(1), 2003.

[115] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer VisionECCV 2014*, pages 346–361. Springer, 2014.

[116] Teuvo Heimonen and Janne Heikkil. A human detection framework for heavy machinery. In *Proc. 20th International Conference on Pattern Recognition (ICPR 2010), Istanbul, Turkey*, pages 416–419, 2010.

[117] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[118] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[119] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[120] D. Hoiem, A. A Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15, 2008.

[121] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR07). IEEE Computer Society*, pages 1–8, 2007.

[122] David H. Hubel, Janice Wensveen, and Bruce Wick. *Eye, brain, and vision.* Scientific American Library, New York, 1995.

[123] S. A Hutchinson and A. C Kak. Planning sensing strategies in a robot work cell with multi-sensor capabilities. *Robotics and Automation, IEEE Transactions on*, 5(6):765–783, 1989.

[124] T. H\bner and R. Pajarola. Real-Time Feature Acquisition and Integration for Vision-Based Mobile Robots. *Advances in Visual Computing*, pages 189–200, 2009.

[125] L. Itti. Automatic foveation for video compression using a neurobiological model of visual attention. *Image Processing, IEEE Transactions on*, 13(10):1304–1318, 2004.

[126] L. Itti and P. Baldi. A principled approach to detecting surprising events in video. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 631–637, 2005.

[127] L. Itti and P. Baldi. Bayesian surprise attracts human attention. *Vision Research*, 49(10):1295–1306, 2009.

[128] L. Itti and C. Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2 (3):194–203, 2001.

[129] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(11):1254–1259, 1998.

[130] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

[131] Satoru Iwata. Submodular function minimization. *Mathematical Programming*, 112(1):45–64, 2008.

[132] Satoru Iwata and James B. Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1230–1237, 2009.

[133] Rishabh Iyer and Jeff Bilmes. Algorithms for Approximate Minimization of the Difference Between Submodular Functions, with Applications. In *In Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, USA, July 2012.

[134] Rishabh Iyer and Jeff A. Bilmes. Submodular-Bregman and the Lova _sz-Bregman Divergences with Applications. In *Advances in Neural Information Processing Systems*, pages 2942–2950, 2012.

[135] H. Jasso. *A reinforcement learning model of gaze following.* PhD thesis, University of California,

San Diego, 2007.

[136] Javal. *Physiologie de la lecture et de l'criture*. F. Alcan, 1905.

[137] V. Javier Traver and A. Bernardino. A review of log-polar imaging for visual perception in robotics. *Robotics and Autonomous Systems*, 58(4):378–398, 2010.

[138] S. Jegelka and J. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1897–1904, 2011.

[139] Huaizu Jiang, Jingdong Wang, Zejian Yuan, Yang Wu, Nanning Zheng, and Shipeng Li. Salient Object Detection: A Discriminative Regional Feature Integration Approach. *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[140] Zhuolin Jiang and Larry S. Davis. Submodular Salient Region Detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[141] Michael Jones and Paul Viola. Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, 3:14, 2003.

[142] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2106–2113, 2009.

[143] D. M Jun and D. L Jones. An energy-aware framework for cascaded detection algorithms. In *Signal Processing Systems (SIPS), 2010 IEEE Workshop on*, pages 1–6, 2010.

[144] T. Kadir and M. Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[145] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. *Computer Vision-ECCV 2004*, pages 228–241, 2004.

[146] K. Kakusho, T. Kitahashi, K. Kondo, and J. C Latombe. Continuous purposive sensing and motion for 2d map building. In *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, volume 2, pages 1472–1477, 1995.

[147] E. R Kandel, J. H Schwartz, T. M Jessell, S. Mack, and J. Dodd. *Principles of neural science*, volume 3. Elsevier New York:, 1991.

[148] Hongwen Kang, Martial Hebert, Alexei A. Efros, and Takeo Kanade. Connecting missing links: object discovery from sparse observations using 5 million product images. In *Computer VisionECCV 2012*, pages 794–807. Springer, 2012.

[149] Andrej Karpathy and Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, 2015. arXiv: 1412.2306.

[150] Andrej Karpathy, Armand Joulin, and Fei Fei F. Li. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in Neural Information Processing Systems*, pages 1889–1897, 2014.

[151] W. Kienzle, F. A Wichmann, B. Scholkopf, and M. O Franz. A nonparametric approach to bottom-up visual saliency. *Advances in neural information processing systems*, 19:689, 2007.

[152] H. Kim, B. Lau, and J. Triesch. Adaptive object tracking with an anthropomorphic robot head. In *From animals to animats 8: proceedings of the seventh [ie eighth] International Conference on Simulation of Adaptive Behavior*, page 104, 2004.

[153] W. N Klarquist and A. C Bovik. Fovea: A foveated vergent active stereo vision system for dynamic three-dimensional scene recovery. *Robotics and Automation, IEEE Transactions on*, 14(5):755–770, 1998.

[154] R. M Klein. Inhibition of return. *Trends in Cognitive Sciences*, 4(4):138–147, 2000.

[155] J. M Kleinberg. On-line search in a simple polygon. In *Proceedings of the fifth annual ACM-SIAM*

symposium on Discrete algorithms, pages 8–15, 1994.

[156] B. C Ko and J. Y Nam. Object-of-interest image segmentation based on human attention and semantic region clustering. *Journal of the Optical Society of America. A, Optics, image science, and vision*, 23(10):2462–2470, 2006.

[157] C. Koch and S. Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. *Hum Neurobiol*, 4(4):219–27, 1985.

[158] A. Krause and D. Golovin. Submodular Function Maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2012.

[159] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, 2008.

[160] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[161] H. Kruppa, M. Castrillon-Santana, and B. Schiele. Fast and robust face finding via local context. In *Joint IEEE Internacional Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 157–164, 2003.

[162] E. Kruse, R. Gutsche, and F. M. Wahl. Efficient, iterative, sensor based 3-D map building using rating functions in configuration space. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 2, pages 1067–1072, 1996.

[163] Y. Kuniyoshi, N. Kita, K. Sugimoto, S. Nakamura, and T. Suehiro. A foveated wide angle lens for active vision. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 3, pages 2982–2988, 1995.

[164] Y. Kuniyoshi, N. Kita, T. Suehiro, and S. Rougeaux. Active stereo vision system with foveated wide angle lenses. *Recent developments in computer vision*, pages 191–200, 1996.

[165] K. N Kutulakos and C. R Dyer. Recovering shape by purposive viewpoint adjustment. *International Journal of Computer Vision*, 12(2):113–136, 1994.

[166] K. N Kutulakos, C. R Dyer, and V. J Lumelsky. Provable strategies for vision-guided exploration in three dimensions. In *IEEE International Conference on Robotics and Automation*, pages 1365–1365, 1994.

[167] C. H Lampert, M. B Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008.

[168] F. Lampi, S. Kopf, M. Benz, and W. Effelsberg. An automatic cameraman in a lecture recording system. In *Proceedings of the international workshop on Educational multimedia and multimedia education*, pages 11–18. ACM Press New York, NY, USA, 2007.

[169] M. F. Land. Motion and vision: why animals move their eyes. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 185(4):341–352, 1999.

[170] M. F Land and M. Hayhoe. In what ways do eye movements contribute to everyday activities? *Vision research*, 41(25-26):3559–3565, 2001.

[171] M. F Land and P. McLeod. From eye movements to actions: how batsmen hit the ball. *Nature Neuroscience*, 3(12):1340–1345, 2000.

[172] Hugo Larochelle and Geoffrey E. Hinton. Learning to combine foveal glimpses with a third-order Boltzmann machine. In *Advances in neural information processing systems*, pages 1243–1251, 2010.

[173] O. Le Meur, P. Le Callet, D. Barba, and D. Thoreau. A coherent computational approach to model bottom-up visual attention. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28 (5):802–817, 2006.

[174] Dan Levi, Shai Silberstein, and Aharon Bar-Hillel. Fast Multiple-Part Based Object Detection Using KD-Ferns. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, Portland, OR, USA, 2013.

[175] B. C Levy. *Principles of signal detection and parameter estimation*. Springer Verlag, 2008.

[176] J. Li, Y. Tian, T. Huang, and W. Gao. A dataset and evaluation methodology for visual saliency in video. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 442–445, 2009.

[177] Liang Li, Wei Feng, Liang Wan, and Jiawan Zhang. Maximum Cohesive Grid of Superpixels for Fast Object Localization. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[178] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900, 2002.

[179] R. Lienhart and A. Wernicke. Localizing and segmenting text in images and videos. *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(4):256–268, 2002.

[180] E. L. Lim, S. Venkatesh, and G. A. W. West. Resolution consideration in spatially variant sensors. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 1, pages 795–799, 1996.

[181] Q. Liu, D. Kimber, J. Foote, L. Wilcox, and J. Boreczky. FLYSPEC: A multi-user video camera system with hybrid human and automatic control. In *Proceedings of the tenth ACM international conference on Multimedia*, page 492, 2002.

[182] Q. Liu, X. Shi, D. Kimber, F. Zhao, and F. Raab. An online video composition system. In *IEEE International Conference on Multimedia & Expo*, 2005.

[183] T. Liu, J. Sun, N. N Zheng, X. Tang, and H. Y Shum. Learning to detect a salient object. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[184] J.M. Lobo, A. Jimnez-Valverde, and R. Real. AUC: a misleading measure of the performance of predictive distribution models. *Global Ecology and Biogeography*, 17(2):145–151, 2008.

[185] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1411.4038*, 2014.

[186] P. Longhurst, K. Debattista, and A. Chalmers. A GPU based saliency map for high-fidelity selective rendering. In *Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 21–29, 2006.

[187] M. Lopes and J. Santos-Victor. A developmental roadmap for learning by imitation in robots. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):308–321, 2007.

[188] M. Lopes, R. Beira, M. Praa, and J. Santos-Victor. An anthropomorphic robot torso for imitation: design and experiments. In *Intelligent Robots and Systems*, volume 1, pages 661–667, 2004.

[189] Lszl Lovsz. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.

[190] D. G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[191] Long Mai, Yuzhen Niu, and Feng Liu. Saliency Aggregation: A Data-driven Approach. *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[192] A. A Makarenko, S. B Williams, F. Bourgault, and H. F Durrant-Whyte. An experiment in integrated exploration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 534–539, 2002.

[193] Sabira K Mannan, Keith H Ruddock, and David S Wooding. Fixation patterns made during brief examination oftwo-dimensional images. *Perception*, 26(8):1059–1072, 1997.

[194] E. Marchand and F. Chaumette. Active vision for complete scene reconstruction and exploration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(1):65–72, 1999.

[195] Ran Margolin, Ayellet Tal, and Lihi Zelnik-Manor. What Makes a Patch Distinct? *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[196] J. Martinez and L. Altamirano. FPGA-based pipeline architecture to transform Cartesian images into foveal images by using a new foveation approach. In *2006 IEEE International Conference on Reconfigurable Computing and FPGA's (ReConFig 2006)*, pages 1–10, 2006.

[197] H. Masnadi-Shirazi and N. Vasconcelos. High detection-rate cascades for real-time object detection. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–6, 2007.

[198] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 417–433, 1993.

[199] D. Meger, P. E Forssn, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. J Little, and D. G Lowe. Curious george: An attentive semantic robot. *Robotics and Autonomous Systems*, 56 (6):503–511, 2008.

[200] G. Metta, F. Panerai, R. Manzotti, and G. Sandini. Babybot: an artificial developing robotic agent. *SAB 2000 Paris, France. Sep. 11*, 16, 2000.

[201] E. Milios, M. Jenkin, and J. Tsotsos. Design and performance of trish, a binocular robot head with torsional eye movements. *International journal of pattern recognition and artificial intelligence*, 7: 51–51, 1993.

[202] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pages 234–243, 1978.

[203] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent Models of Visual Attention. *arXiv:1406.6247 [cs, stat]*, June 2014. arXiv: 1406.6247.

[204] I. Mukherjee, C. Rudin, and R. E. Schapire. The rate of convergence of AdaBoost. *arXiv preprint arXiv:1106.6024*, 2011.

[205] K. Murphy, A. Torralba, and W. Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. *Advances in neural information processing systems*, 16, 2003.

[206] D. W Murray, I. D Reid, and A. J Davison. Steering and navigation behaviours using fixation. In *Proceedings of the 7th British Machine Vision Conference, Edinburgh*, pages 635–644, 1996.

[207] B. Nabbe and M. Hebert. Extending the path-planning horizon. *The International Journal of Robotics Research*, 26(10):997, 2007.

[208] H. H Nagel and B. Neumann. On 3-D reconstruction from two perspective views. In *Proc. IJCAI81*, volume 2, 1981.

[209] M. Narasimhan and J. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Proc. Conf. Uncertainty in Artifical Intelligence, Edinburgh, Scotland, July. Morgan Kaufmann Publishers*, 2005.

[210] L. Natale, S. Rao, and G. Sandini. Learning to act on objects. In *Biologically Motivated Computer Vision*, pages 89–100, 2010.

[211] V. Navalpakkam and L. Itti. Modeling the influence of task on attention. *Vision Research*, 45(2): 205–231, 2005.

[212] V. Navalpakkam and L. Itti. A goal oriented attention guidance model. In *Biologically Motivated Computer Vision*, pages 81–118, 2010.

[213] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functionsI. *Mathematical Programming*, 14(1):265–294, 1978.

[214] M. E Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*,

pages 722–729, 2008.

[215] H. C Nothdurft. Salience from feature contrast: additivity across dimensions. *Vision Research*, 40 (10-12):1183–1201, 2000.

[216] A. Oliva, A. Torralba, M. S Castelhano, and J. M Henderson. Top-down control of visual attention in object detection. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I–253, 2003.

[217] F. Orabona, G. Metta, and G. Sandini. Object-based Visual Attention: a Model for a Behaving Robot. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 89–89, 2005.

[218] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 193–199, 1997.

[219] N. Ouerhani, R. von Wartburg, H. H\gli, and R. M\ri. Empirical validation of the saliency-based model of visual attention. *Electronic Letters on Computer Vision and Image Analysis*, 3(1):13–24, 2004.

[220] F. Pardo, I. Llorens, F. Mic, and J. A Boluda. Space variant vision and pipelined architecture for time to impact computation. In *camp*, page 122, 2000.

[221] D. Parkhurst, K. Law, and E. Niebur. Modeling the role of salience in the allocation of overt visual attention. *Vision Research*, 42(1):107–123, 2002.

[222] M. Pedersoli, J. Gonzlez, A. Bagdanov, and J. Villanueva. Recursive coarse-to-fine localization for fast object detection. *Computer VisionECCV 2010*, pages 280–293, 2010.

[223] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A Coarse-to-fine approach for fast deformable object detection. *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011.

[224] J. B Pelz and R. Canosa. Oculomotor behavior and perceptual strategies in complex tasks. *Vision Research*, 41(25-26):3587–3596, 2001.

[225] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, 1990.

[226] M. Perreira Da Silva, V. Courboulay, A. Prigent, P. Estraillier, et al. Evaluation of preys/predators systems for visual attention simulation. *VISAPP 2010*, 2010.

[227] C. Pinhanez and A. F Bobick. Intelligent studios: Using computer vision to control TV cameras. In *Proc. of IJCAI*, volume 95, pages 69–76, 1997.

[228] R. Pito. A solution to the next best view problem for automated surface acquisition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(10):1016–1030, 1999.

[229] M. I Posner, R. D Rafal, L. S Choate, and J. Vaughan. Inhibition of return: Neural basis and function. *Cognitive Neuropsychology*, 1985.

[230] C. M Privitera and L. W Stark. Algorithms for defining visual regions-of-interest: Comparison with eye fixations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(9):970–982, 2000.

[231] F. Z Qureshi and D. Terzopoulos. Planning ahead for PTZ camera assignment and handoff. In *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, pages 1–8, 2009.

[232] E. Rahtu, J. Kannala, M. Salo, and J. Heikkil. Segmenting salient objects from images and videos. *Computer VisionECCV 2010*, pages 366–379, 2010.

[233] E. Rahtu, J. Kannala, and M. Blaschko. Learning a category independent object detection cascade. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1052–1059, 2011.

[234] S. Ramanathan, H. Katti, N. Sebe, M. Kankanhalli, and T. S Chua. An eye fixation database for

saliency detection in images. *Computer VisionECCV 2010*, pages 30–43, 2010.

[235] B. S Reddy and B. N. Chatterji. An FFT-based technique for translation, rotation, and scale-invariant image registration. *Image Processing, IEEE Transactions on*, 5(8):1266–1271, 1996.

[236] L. W Renninger, J. Coughlan, P. Verghese, and J. Malik. An information maximization model of eye movements. *Advances in neural information processing systems*, 17:1121–1128, 2005.

[237] D. Robinson and P. Milanfar. Fast local and global projection-based methods for affine motion estimation. *Journal of Mathematical Imaging and Vision*, 18(1):35–54, 2003.

[238] A. S Rojer and E. L Schwartz. Design considerations for a space-variant visual sensor with complex-logarithmic geometry. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume 2, pages 278–285, 1990.

[239] R. Rosenholtz. A simple saliency model predicts a number of motion popout phenomena. *Vision Research*, 39(19):3157–3163, 1999.

[240] R.Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley and Sons Inc., 1981.

[241] Dmitry Rudoy, Dan B. Goldman, Eli Shechtman, and Lihi Zelnik-Manor. Learning video saliency from human gaze using candidate selection. *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[242] J. Ruesch, M. Lopes, A. Bernardino, J. Hornstein, J. Santos-Victor, and R. Pfeifer. Multimodal saliency-based bottom-up attention a framework for the humanoid robot icub. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 962–967, 2008.

[243] Yong Rui, Anoop Gupta, Jonathan Grudin, and Liwei He. Automating lecture capture and broadcast: technology and videography. *Multimedia Systems*, 10(1):3–15, June 2004.

[244] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, and others. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, pages 1–42, 2014.

[245] U. Rutishauser, D. Walther, C. Koch, and P. Perona. Is bottom-up attention useful for object recognition? In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–37, 2004.

[246] G. Sandini and V. Tagliasco. An anthropomorphic retina-like structure for scene analysis. *Computer Graphics and Image Processing*, 14(4):365–372, 1980.

[247] G. Sandini, P. Questa, D. Scheffer, B. Dierickx, and A. Mannucci. A retina-like CMOS sensor and its applications. In *Sensor Array and Multichannel Signal Processing Workshop. 2000. Proceedings of the 2000 IEEE*, pages 514–519, 2000.

[248] B. Scassellati. Eye finding via face detection for a foveated active vision system. In *Proceedings of the National Conference on Artificial Intelligence*, pages 969–976, 1998.

[249] B. Schauerte, J. Richarz, T. Pl\tz, C. Thurau, and G. A Fink. Multi-modal and multi-camera attention in smart environments. In *Proceedings of the 2009 international conference on Multimodal interfaces*, pages 261–268, 2009.

[250] B. Schauerte, J. Richarz, and G. A Fink. Saliency-based Identification and Recognition of Pointed-at Objects. In *Proc. Int. Conf. Intell. Robots Syst*, 2010.

[251] E. L Schwartz, D. N Greve, and G. Bonmassar. Space-variant active vision: Definition, overview and examples. *Neural Networks*, 8(7-8):1297–1308, 1995.

[252] M. Schwarz, R. Hauschild, B. J Hosticka, J. Huppertz, T. Kneip, S. Kolnsberg, L. Ewe, and H. K Trieu. Single-chip CMOS image sensors for a retina implant system. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 46(7):870–877, 1999.

[253] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In *Inter-*

national Conference on Learning Representations (ICLR 2014). CBLS, April 2014.

[254] H. Seyffarth and D. Denny-Brown. The grasp reflex and the instinctive grasp reaction. *Brain*, 71 (2):109, 1948.

[255] F. Shahbaz Khan, J. van de Weijer, and M. Vanrell. Top-down color attention for object recognition. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 979–986, 2009.

[256] Keyang Shi, Keze Wang, Jiangbo Lu, and Liang Lin. PISA: Pixelwise Image Saliency by Aggregating Complementary Appearance Contrast Measures with Spatial Priors. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[257] A. Shmuel and M. Werman. Active vision: 3d from an image sequence. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume 1, pages 48–54, 1990.

[258] Abhinav Shrivastava, Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Data-driven visual similarity for cross-domain image matching. In *ACM Transactions on Graphics (TOG)*, volume 30, page 154. ACM, 2011.

[259] C. Siagian and L. Itti. Biologically-inspired robotics vision monte-carlo localization in the outdoor environment. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1723–1730, 2007.

[260] R. Sim and N. Roy. Global a-optimal robot exploration in slam. In *IEEE International Conference on Robotics and Autonmation*, volume 1, page 661, 2005.

[261] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, September 2014. arXiv: 1409.1556.

[262] Parthipan Siva, Chris Russell, Tao Xiang, and Lourdes Agapito. Looking Beyond the Image: Unsupervised Learning for Object Saliency and Detection. *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[263] T. Smith, D. R. Thompson, D. S. Wettergreen, N. A. Cabrol, K. A. Warren-Rhodes, and S. J. Weinstein. Life in the Atacama: Science autonomy for improving data quality. *J. Geophysical. Research*, 2007.

[264] Trey Smith. *Probabilistic Planning for Robotic Exploration*. PhD thesis, Carnegie Mellon University, 2007.

[265] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *International Conference on Computer Graphics and Interactive Techniques*, pages 835–846. ACM New York, NY, USA, 2006.

[266] J. Sochman and J. Matas. Waldboost-learning for time constrained sequential detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 150–156, 2005.

[267] C. Spampinato, D. Giordano, R. Di Salvo, Y. H.J Chen-Burger, R. B Fisher, and G. Nadarajan. Automatic fish classification for underwater species behavior understanding. In *Proceedings of the first ACM international workshop on Analysis and retrieval of tracked events and motion in imagery streams*, pages 45–50, 2010.

[268] V. A Sujan and S. Dubowsky. Efficient information-based visual robotic mapping in unstructured environments. *The International Journal of Robotics Research*, 24(4):275, 2005.

[269] M. J Swain and M. A Stricker. Promising directions in active vision. *International Journal of Computer Vision*, 11(2):109–126, 1993.

[270] Christian Szegedy, Scott Reed, Dumitru Erhan, and Dragomir Anguelov. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*, 2014.

[271] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *CVPR 2015*, 2015.

[272] Raphael Sznitman, Carlos Becker, Fran\ccois Fleuret, and Pascal Fua. Fast Object Detection with Entropy-Driven Evaluation. *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[273] N. Tamayo and V. J Traver. Entropy-based saliency computation in log-polar images. In *Intl. Conf. on Computer Vision Theory and Applications*, volume 1, pages 501–506, 2008.

[274] K. A Tarabanis, P. K Allen, and R. Y Tsai. A survey of sensor planning in computer vision. *Robotics and Automation, IEEE Transactions on*, 11(1):86–104, 1995.

[275] D. R Thompson and R. Castano. A performance comparison of rock detection algorithms for autonomous planetary geology. In *Proceedings of the 2007 IEEE Aerospace Conference*, 2007.

[276] D. R. Thompson and D. Wettergreen. Intelligent maps for autonomous kilometer-scale science survey. In *Proc. i-SAIRAS*, 2008.

[277] David R Thompson. *Intelligent Mapping for Autonomous Robotic Survey*. Ph.D Thesis, Robotics Institute, Carnegie Mellon University, August 2008.

[278] S Thrun. A Bayesian Approach to Landmark Discoverty and Active Perception in Mobile Robot Navigation. Technical Report CMU-CS-96-122, Carnegie Mellon University, May 1996.

[279] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[280] M. Tistarelli and G. Sandini. On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 401–410, 1993.

[281] F. Tong and Z. N Li. Reciprocal-wedge transform for space-variant sensing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(5):500–511, 1995.

[282] A. Torralba, A. Oliva, M. S Castelhano, and J. M Henderson. Contextual guidance of eye movements and attention in real-world scenes: The role of global features in object search. *Psychological review*, 113(4):766, 2006.

[283] B. Tovar, L. Mu\ noz Gmez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, and S. Hutchinson. Planning exploration strategies for simultaneous localization and mapping. *Robotics and Autonomous Systems*, 54(4):314–331, 2006.

[284] P. E Trahanias, S. Velissaris, and T. Garavelos. Visual landmark extraction and recognition for autonomous robot navigation. In *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*, volume 2, pages 1036–1043, 2002.

[285] J. K Tsotsos. Analyzing vision at the complexity level. *Behavioral and Brain Sciences*, 13(3):423–469, 1990.

[286] J. K Tsotsos. On the relative complexity of active vs. passive visual search. *International journal of computer vision*, 7(2):127–141, 1992.

[287] A. Ude, C. G Atkeson, and G. Cheng. Combining peripheral and foveal humanoid vision to detect, pursue, recognize and act. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2173–2178, 2003.

[288] A. Ude, C. Gaskett, and G. Cheng. Foveated vision systems with two cameras per eye. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3457–3462, 2006.

[289] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[290] R. Valenti, N. Sebe, and T. Gevers. Image saliency by isocentric curvedness and color. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2185–2192, 2009.

[291] K. E.A Van De Sande, T. Gevers, and C. G.M Snoek. Evaluating color descriptors for object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1582–

1596, 2010.

[292] J. Van De Weijer, T. Gevers, and A. D Bagdanov. Boosting color saliency in image feature detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(1):150–156, 2006.

[293] I. Van Der Linde, U. Rajashekar, A. C Bovik, and L. K Cormack. DOVES: A database of visual eye movements. *Spatial Vision*, 22(2):161–177, 2009.

[294] J. Van der Spiegel, G. Kreider, C. Claeys, I. Debusschere, G. Sandini, P. Dario, F. Fantini, P. Bellutti, and G. Soncini. A foveated retina-like sensor using CCD technology. *Analog VLSI implementation of neural systems*, pages 189–211, 1989.

[295] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple Kernels for Object Detection. 2009.

[296] T. Vidal-Calleja, A. J. Davison, J. Andrade-Cetto, and D. W Murray. Active control for single camera SLAM. In *Proc. of ICRA*, 2006.

[297] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Advances in Neural Information Processing System*, 14, 2001.

[298] P. Viola and M. Jones. 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, pages I–511, 2001.

[299] P. Viola and M. J Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[300] L. Von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64, 2006.

[301] A. Wald. *Sequential analysis*. Dover Publications, 2004.

[302] D. Walther and C. Koch. Modeling attention to salient proto-objects. *Neural Networks*, 19(9): 1395–1407, 2006.

[303] D. Walther, U. Rutishauser, C. Koch, and P. Perona. Selective visual attention enables learning and recognition of multiple objects in cluttered scenes. *Computer Vision and Image Understanding*, 100 (1):41–63, 2005.

[304] H. Wang and P. Chu. Voice source localization for automatic camera pointing system in videoconferencing. In *Applications of Signal Processing to Audio and Acoustics, 1997. 1997 IEEE ASSP Workshop on*, pages 4–pp, 1997.

[305] M. Wang, J. Konrad, P. Ishwar, K. Jing, and H. Rowley. Image saliency: From intrinsic to extrinsic context. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 417–424, 2011.

[306] Xiaoyu Wang, Ming Yang, Shenghuo Zhu, and Yuanqing Lin. Regionlets for generic object detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 17–24. IEEE, 2013.

[307] Z. Wang and B. Li. A two-stage approach to saliency detection in images. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 965–968, 2008.

[308] D. Wettergreen, N. Cabrol, V. Baskaran, F. Caldern, S. Heys, D. Jonak, R. A. Luders, D. Pane, T. Smith, J. Teza, et al. Second experiments in the robotic investigation of life in the atacama desert of chile. In *Proceedings of iSAIRAS*, 2005.

[309] D. Wettergreen, N. Cabrol, J. Teza, P. Tompkins, C. Urmson, V. Verma, M. Wagner, and W. Whittaker. First experiments in the robotic investigation of life in the atacama desert of chile. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 873–878, 2005.

[310] P. Whaite and F. P Ferrie. From uncertainty to visual exploration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1038–1049, 1991.

[311] P. Whaite and F. P Ferrie. Autonomous exploration: Driven by uncertainty. *Pattern Analysis and*

Machine Intelligence, IEEE Transactions on, 19(3):193–205, 1997.

[312] D. J. White, C. Svellingen, and N. J. C. Strachan. Automated measurement of species and length of fish by computer vision. *Fisheries Research*, 80(2-3):203–210, 2006.

[313] R. Wodnicki, G. W Roberts, and M. D Levine. A foveated image sensor in standard CMOS technology. In *Custom Integrated Circuits Conference, 1995., Proceedings of the IEEE 1995*, pages 357–360, 1995.

[314] G. Wolberg and S. Zokai. Robust image registration using log-polar transform. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 1, pages 493–496, 2000.

[315] J. M Wolfe. Guided search 2.0 A revised model of visual search. *Psychonomic bulletin & review*, 1 (2):202–238, 1994.

[316] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg. Fast asymmetric learning for cascade face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(3):369–382, 2008.

[317] Lan Wu. *Multi-view Hockey Tracking with Trajectory Smoothing and Camera Selection.* PhD thesis, University of British Columbia, 2008.

[318] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *arXiv:1502.03044 [cs]*, February 2015. arXiv: 1502.03044.

[319] H. Yamamoto, Y. Yeshurun, and M. D Levine. An active foveated vision system: Attentional mechanisms and scan path covergence measures. *Computer Vision and Image Understanding*, 63(1): 50–65, 1996.

[320] B. Yamauchi, A. Schultz, and W. Adams. Integrating exploration and localization for mobile robots. *Adaptive Behavior*, 7(2):217, 1999.

[321] Daniel L Yamins, Ha Hong, Charles Cadieu, and James J DiCarlo. Hierarchical Modular Optimization of Convolutional Networks Achieves Representations Similar to Macaque IT and Human Ventral Stream. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3093–3101. Curran Associates, Inc., 2013.

[322] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical Saliency Detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[323] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency Detection via Graph-Based Manifold Ranking. *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[324] A. L Yarbus. Eye movements during perception of complex objects. *Eye movements and vision*, 7: 171–196, 1967.

[325] M. Yeasin and R. Sharma. Foveated Vision Sensor and Image ProcessingA Review. *Machine Learning and Robot Perception*, pages 57–98, 2005.

[326] Y. Yeshurun and M. Carrasco. Attention improves or impairs visual performance by enhancing spatial resolution. *Nature*, 396(6706):72–75, 1998.

[327] S.S. Young, P.D. Scott, and C. Bandera. Foveal automatic target recognition using a multiresolution neural network. *Image Processing, IEEE Transactions on*, 7(8):1122–1135, 1998.

[328] A.L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). *Advances in Neural Information Processing Systems*, 2:1033–1040, 2002.

[329] A. Zaharescu, A. L Rothenstein, and J. K Tsotsos. Towards a biologically plausible active visual search model. *Attention and Performance in Computational Vision*, pages 133–147, 2005.

[330] Y. Zhai and M. Shah. Visual attention detection in video sequences using spatiotemporal cues. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 815–824, 2006.

[331] L. Zhang, M. H Tong, T. K Marks, H. Shan, and G. W Cottrell. SUN: A Bayesian framework for saliency using natural statistics. *Journal of Vision*, 8(7), 2008.

[332] W. Zhang, G. Zelinsky, and D. Samaras. Real-time accurate object detection using multiple resolutions. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007.

[333] M. Zhu. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2004.

[334] Q. Zhu, M.C. Yeh, K.T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491–1498, 2006.

[335] C. Lawrence Zitnick and Piotr Dollr. Edge boxes: Locating object proposals from edges. In *Computer VisionECCV 2014*, pages 391–405. Springer, 2014.

Appendix A

Reefbot

A.1 Introduction

This appendix is included in the thesis as it provides a good example of a complex, high level task where excess imagery can degrade task performance. In this setting, a visual utility process would not only reduce the computational cost, but also be able to improve the task performance.

Reefbot was an underwater robotic exhibit at the PPG Aquarium in Pittsburgh, USA that was operational for over 1300 hours in between 2010-2012. Kids drove the robot through the large exhibit tank and took pictures of fish, which could then be classified to display more information about the species. As they interact with the exhibit, they learn about underwater robotics, fish species and marine conservation.

While Reefbot is used as an educational exhibit, it was simultaneously acting as a development platform for automated visual marine species identification. An underwater robotic system that could reliably classify fish species would be invaluable for monitoring and exploring coral reefs that are rapidly disappearing throughout the worlds oceans. We are using the natural setting and large quantity of visual data collected from Reefbot in order to develop algorithms for visual identification of the these marine species. This chapter presents a dataset for realistic marine species identification along with the initial results from our classification system.

Though object classification has been a central topic in computer vision for many years, it has not been explored for classifying fish in underwater natural scenes from moving platforms. This medium presents some unique challenges. Underwater, light reacts differently than in air. As light travels through the water, red light is scattered, causing objects to appear bluer the further they are from the light source

Figure A.1: The Reefbot robot. Image ©Paul A. Selvaggio

and the camera. Furthermore, underwater scenes are illuminated from above, often through a turbulent surface, causing the light to shimmer. Complicating the task, fish species can be difficult to distinguish, even for a human. Some fish only differ by a slightly different shape, or different skin markings. Also, in a natural setting, fish deform as they swim and can appear in any arbitrary pose, not just the canonical profile view. As a result of these factors, there can often be a larger visual difference within a class than between classes. Finally, it can be difficult to segment fish from their natural background as many species are colored specifically to camouflage with their background and 3D sensors like the Kinect are not effective underwater (the Kinect can operate underwater, but only to 1.5 m).

This chapter is broken into four major sections. In appendix A.2, we describe the related work to the problem we address and the techniques we use. In appendix A.3, we detail the robotic platform. Appendix A.4 describes the dataset created and marine species classification system. Finally, appendix A.5 presents the analysis from some initial experiments.

A.2 Related Work

Automated fish classification has been previously demonstrated in structured settings. In these instances the setup is engineered to enable a vision system to easily segment the fish from the background. This enables contour-based methods to classify the silhouette of the fish to be used, which has proven to be effective. Examples of this approach include a fishing boat that uses a backlight on a conveyor belt to monitor the catch [312], spillways that use infrared diodes to count fish migrating past dams [39], classification of dead bodies placed on consistent backgrounds for use in catalogs [7] and a video sequence of a fish in front of a blue wall [26].

A couple of studies have tried to identify fish in the natural settings that were interested in for this study [75, 267]. Both use static cameras anchored to the sea floor. Then, the fish are segmented from the background by calculating the average background frame and looking for outliers. Once again, this enables countour-based methods to be used. In our case, Reefbot is a moving camera and thus, the background cannot be calculated as easily. As a result, a tight segmentation is not possible and countour-based methods cannot be used. Furthermore, the two studies use small datasets for testing, which could artificially improve performance, especially since these datasets are extracted from video. In particular, [75] only classify two different kinds of fish, while [267] only test 100 images spread across 10 different fish species.

When classifying objects in natural contexts, much more work has been done in non-marine settings. A large body of this work aims to distinguish between various categories of objects like bicycles vs. beer bottles. The best classifiers use a combination of features that describe shape, texture, color and context [112, 295]. However, differences between different species of fish are more subtle that the differences between broad categories and thus, the same techniques are not necessarily viable.

More applicable to this task in this chapter are studies in object instance recognition. This body of work develops techniques to identify images of the same object in multiple scenes. One example is face recognition where a classifier identifies individuals from their face [10]. In this work a face detector [299] is used to localize specific points on the face. Then, features calculated at those points are classified. This technique works best when the face is viewed straight on and thus, would be the equivalent of only classifying full profiles of fish, which is unrealistic in a dynamic setting.

Another class of work in instance detection is image search. These techniques extract robust features like SIFT [190] at interest points for a corpus of images to build an index. Then, at query time, the same features are extracted from the query image and matched to those in the index in order to retrieve matching examples. Results can also be ranked by the geometric consistency of the points found [51]. This technique could work very well for the species identification task because special markings on the individuals would be extracted into an index. Furthermore, the technique does not require a tight segmentation of the instance. However, sometimes species have very similar structure and only vary by color.

Figure A.2: The Reefbot exhibit console. Image ©Paul A. Selvaggio

Most analogous to our problem is the work by Nilsback et al. in classifying different species of flowers [214]. In this study, the authors deliberately select a large collection of flowers that have similar color and/or shapes. Images are taken in natural light with a variety of backgrounds. Though the flower classification problem exhibits many of the same issues as marine species classification, underwater photography presents some unique challenges. Water heavily distorts colors of objects and the incident light is often variable because of turbulence at the water's surface. Furthermore, in the flower study, all of the pictures show the complete flower roughly in the center of the image, whereas this study uses more unconstrained data where sometimes, only portions of the fish are visible or the fish is in a random pose.

A.3 Robotic Platform

Reefbot is a physically modified VideoRay® Pro 4 ROV with stock firmware. The stock video camera has been removed and replaced with an Arecont AV10005 HD surveillance camera with a fixed focus lens that operates on an Ethernet network. In order to accommodate the larger video camera, the body of the vehicle was extended 10cm and a new shell was developed (fig. A.1). Power for the camera is supplied using an onboard relay that can be controlled using the main firmware. Video from the camera is transmitted using two twisted pairs in the tether and routed to a 100 base-T network link on the surface.

Control for the main vehicle and camera are provided by a PC integrated into the console of the exhibit and running custom software built on top of ROS (Robot Operating System). The console itself contains two stations: one for a driver and one for a fish specialist. The driver controls the robot around the tank and takes pictures, while the fish specialist helps to identify the species of fish found in the tank. Once an individual fish is identified, the patrons are presented with more information about the fish species. Patrons who are not directly at the console interact with the exhibit by watching the robot move through the two story tank and by monitoring the progress of the active participants on external monitors (fig. A.2).

A.4 Marine Species Classification

A two stage process is used to visually classify marine species. In the first stage, candidate regions are extracted from the background by identifying those areas that are moving relative to the predominant motion in the scene. Unlike previous work in fish classification, Reefbot moves in the scene and fish appear in front of complex backgrounds. As a result, the fish cannot be as easily segmeneted from the background. In the second stage, features are extracted for each candidate region and classified to determine the most likely match. As we cannot perform a tight segmentation, we cannot use contour-based features and instead rely on distinctive textures and colors. This section will present the details of the two stages of this process and describe the dataset used for evaluation in this study.

Figure A.3: An example frame from Reefbot. The scene can have many fish, they can appear in arbitrary orientations and the lighting makes them difficult to identify.

A.4.1 Dataset

The dataset used for evaluation in this study is extracted from video taken by Reefbot during normal operations. Example frames were taken at 1Hz over a 5 hour period through the middle of the day. The tank is lit from above through turbulent water with lights of different temperatures. Furthermore, ambient daylight enters the tank through the viewing panels on the side. The resulting lighting conditions are very challenging for an automated vision algorithm. The light shimmers and changes color from one area of the tank to another. Furthermore, because water absorbs red light, there is a noticeable blue shift to colors as object move further away from the camera (fig. A.3).

Using the example frames from the video, the automated fish extraction process described in the following section is applied. This process identifies portions of the frame that move relative to the motion of the entire frame. The frame is ignored if the gross motion of the frame cannot be determined. For comparison, the fish are also extracted from the same frames using Amazon's Mechanical Turk. Workers are asked to draw bounding boxes around all fish in the scene. Once the fish are extracted, they are labeled manually by an expert. The resulting dataset is summarized in fig. A.4 and will be released publically at: http://www.andrew.cmu.edu/user/mdesnoye/reefbot. The set is heavily biased by species ranging from 15 to 1766 instances of a particular species. This occurred because some species are more common in the tank (e.g. there are 15 Golden Trevally and only one Bonnethead shark) and different species have different activity levels. Also, as we are sampling from a video, successive frames can be quite similar, making the visual recognition task easier. Thus, for all experiments, the dataset is split so that the test set consists of a contiguous 10 minute segment of video. The results from multiple runs with different held out sets are then averaged together for the final results.

A.4.2 Fish Extraction

Reefbot moves through its environment and fish appear in front of complex backgrounds, making accurate background segmentation nigh impossible. Therefore, we take an approach that aims to roughly identify the location of the fish and use a classification technique that does not require a tight segmentation.

Our automatic fish extraction is performed using a probabilistic framework that assumes that those objects in the frame that move relative to the background must be fish. In particular, we assume that the dominant affine motion of the frame is roughly equal to the relative motion of the robot to the background. This assumption will break in four cases. First, when a fish swims close to the camera, obscuring most of the view, the fish becomes the "background". Second, when observing static objects at significantly different distances, parallax will be relevant and the close object will appear to move relative to the background. Third, if the robot moves quickly in three dimensions relative to the frame rate, the motion will no longer appear affine. Finally, our assumption breaks when the robot looks outside of the tank and

Species	Human Dataset	Machine Dataset
Not a fish	57	976
Zebra Shark	467	1766
Black Tip Reef Shark	20	48
Bonnethead Shark	90	205
Golden Trevally	206	435
Big Nose Tang	355	729
Unicorn Tang	159	392
Orbiculate Bat Fish	249	508
Half Moon Angelfish	20	41
Passer Angelfish	15	42
Speckled Blue Grouper	62	139
Blueline Snapper	303	659
Guineafowl Puffer	21	42
Sailfin Tang	268	499
Pink Tail Triggerfish	52	106
Emperor Red Snapper	53	180
Blue Tang	71	115
Total	**2468**	**6819**

Figure A.4: Summary of the number of instances for each species in the human and machine extracted datasets

sees another type of moving animal: humans.

In order to identify the affine motion between two successive frames, we use the iterative process from [237]. This process uses a subsampled version of the image for speed and then refines its estimate of the transformation using optical flow equations. Once the two frames are aligned, the abnormal motion is calculated by measuring the distance, in LAB color space, between the two frames. LAB color space is used in order to minimize the impact of lighting changes. We then calculate the probability of a given location containing a fish using a smooth sigmoid threshold (eq. (A.1)). The threshold is found by fitting a Pareto function to the distance image and selecting the point where 3% of the pixels would be included. This thresholding compensates for scenes of significantly different brightness.

$$p(F_{t_2}|I_{t_1}, I_{t_2}) = (1 + e^{T_{pareto} - dist(I_{t_2}, I_{t_1})})^{-1} \tag{A.1}$$

The approach described thus far is sensitive to transient changes or spurious detections, which occur frequently due to the shimmering light in the scene. Thus, we also include a prior to encode our belief that fish will be close to their prior location. The prior is described by eq. (A.2):

$$p(F_{t_2}|p(F_{t_1})) = \alpha + (1 - \alpha)(p(F_{t_1}) * N(0, \sigma(t_2 - t_1))) \tag{A.2}$$

α is a regularization term to ensure that over time, the prior does not converge to a single point while the previous estimate of a fish's location is convolved with a gaussian. The width of the Gaussian is determined by the amount of time between frames and the typical speeds of fish in the tank.

The final probability of finding a fish in a given pixel is thus defined as:

$$p(F_{t_2}) = \lambda * p(F_{t_2}|I_{t_1}, I_{t_2}) + (1 - \lambda)p(F_{t_2}|p(F_{t_1})) \tag{A.3}$$

Once the probability is calculated for the entire frame, the fish are extracted by thresholding the probability and filtering out those blobs that are too small.

A.4.3 Fish Classification

Once patches of the image are identified as potential fish, they are evaluated by a classifier. The interactive nature of the exhibit necessitates a classification solution that has a fast query time. Furthermore, a given species can appear in drastically different lighting, orientations and shapes, creating instances that will be spread throughout a feature space. Therefore, in our classification experiments, we use the image search framework from [51]. This approach has very fast query times and the capability to segment the feature space arbitrarily.

In this framework, a dictionary is created by extracting features from a large video sequence and clustering the those features into a set of visual words. Then, a document index is built using labeled image patches. For each patch, features are extracted and assigned to their nearest visual word in the dictionary. The words are then collected into a histogram and represented as a vector. This is the same representation used by in text search and data mining. Thus, all the tools developed in those communities can be used. At query time, the incoming image is converted to its vector word form and then used search the index. The candidate matches are scored using two different methods. The first approach is the traditional tf-idf approach with a cosine similarity metric.

The second scoring approach is detailed in [51] and adds an additional geometric reranking after the initial ranking. In this case, our visual words contain geometric semantics if the features are interest point descriptors associated with a specific location in the image. Interest point descriptors like SIFT can be matched across frames. Therefore, when scoring two image documents, we can also count the number of matching pairs that are geometrically consistent. To do this, we take the top 20 returns from the initial search and rerank the results based on the maximum number of pairs of points that are consistent with a Homography found using RANSAC.

Once all the documents are scored, if the top result is over a confidence threshold, the species has been identified and more information about it is shown to the user. If the top result is not over a confidence threshold, than the up to 5 distinct species labels are returned to the user who, is asked to identify which fish he/she took a picture of.

A.4.3.1 Features

Many factors differentiate fish species. Analogous to the flowers in [214], different fish species can have similar colors, similar markings or similar shapes. Furthermore, some species significantly change in different lighting conditions or even as they age. As a result, along some dimensions, there can be more variation between instances within a species than between different species. Thus, multiple classes of features are necessary to differentiate between species. In this work, we examine two classes of features: color histograms and shape based interest point descriptors. Unlike previous work, we cannot use contour-based methods because we are not accurately segment the entire fish from the background. All of the features are assigned to visual words to be used in the classification framework described above.

HSV Color Features Color is difficult in underwater scenes. Water scatters red light, so objects further from the camera or deeper from the surface will appear bluer (fig. A.3). However, some fish are very different colors, so even with the color distortion, it is possible to differentiate some species by looking at their color. To do this, we use a histogram of representative colors in the HSV color space. HSV is used because distances in HSV are less sensitive to variations in illumination.

Greyscale Shape Descriptors Distinctive shapes, like markings on the skin or the shape of a fin, can differentiate fish species. Therefore, we identify these shapes using SURF [23] descriptors evaluated at points identified by a Hessian interest point detector.

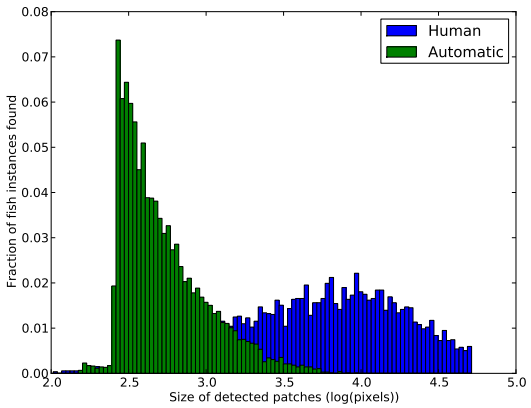


Figure A.5: Histogram of the sizes of the two datasets. The automatically extracted fish patches are significantly smaller than the human labeling because it identifies less of the fish.

Color Shape Descriptors Interest point descriptors are traditionally evaluated in greyscale. However, color is a very distinctive element of a fish species and so we analyze the effect of incorporating color into a SURF detector. In a previous study [291], SIFT features evaluated in different color spaces were analyzed for their stability in the face of lighting intensity and color changes, which are very important in our setting. Promising results were shown with both CSIFT and OpponentSIFT. We use their SURF equivalents in this study for speed.

A.5 Evaluation

A.5.1 Quality of Segmentation

An underwater setting can be very complex and many fish have evolved camouflage for their natural environment. This creates challenges for an automated segmentation algorithm. In this experiment, we examine the performance of our approach, which only uses the apparent motion relative to the scene. The automated extraction is compared to the bounding boxes identified by human subjects. As motion is measured by image differencing, only small regions at the front and back of a uniform region will be easily identified. This approach will not create complete segmentations of the fish, but it will be able to identify smaller regions that are likely to be part of the fish. Therefore, in our analysis, we consider any overlap between the region found and the human bounding box to be a positive identification. Figure A.5 shows the histogram of detected fish sizes for for the two extraction techniques. Though the automated technique identifies less of the fish, it can still be effective as it is more likely to identify visually distinctive portions of the individual, which are needed for classification.

Figure A.6 shows the precision recall curve of our approach generated by changing the confidence threshold. Even though we are only using motion to extract the fish, we are still able to do so precisely as long as we do not need to extract all the fish in the scene. We still are able to achieve over 70% precision while extracting 20% of the fish.

A.5.2 Impact of Segmentation Quality

Using an inaccurate segmentation can add noise to the classification system and hurt performance. If the region is too large, then extra features will be added to the document vector, increasing the noise. If

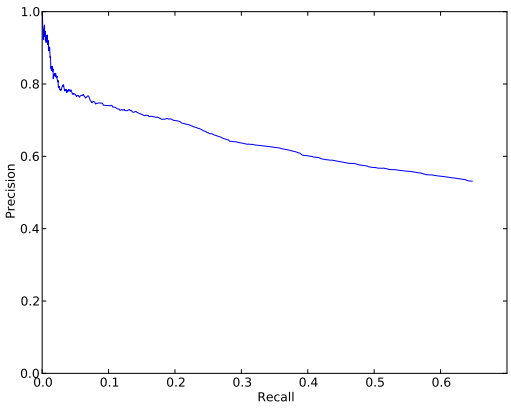


Figure A.6: Precision Recall curve of the fish extractor. Even though we are only using motion, we are still able to achieve over 70% precision while still being able to extract 20% of the fish.

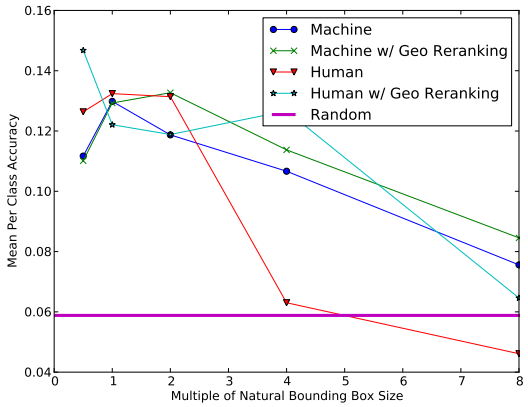
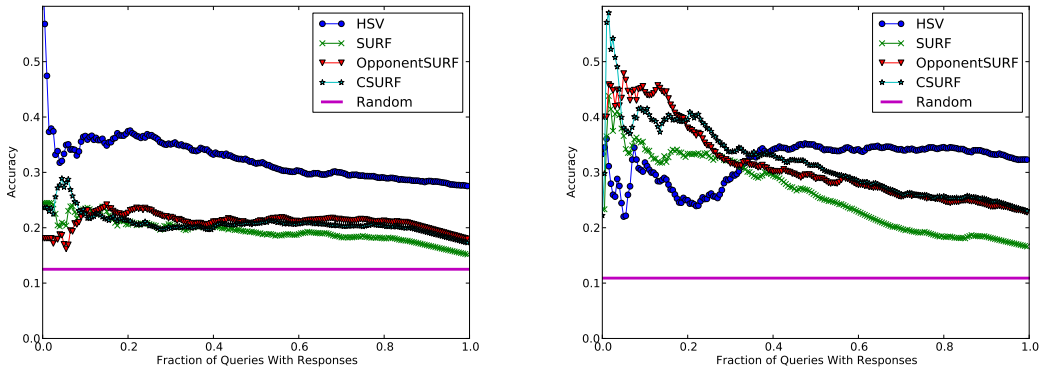


Figure A.7: Histogram of the sizes of the two datasets. The automatically extracted fish patches are significantly smaller than the human labeling because it identifies less of the fish.

the region is too small, then there may not be enough features extracted to identify the fish. To test the impact of segmentation quality on the performance of the classification, we use both the human based segmentation and bounding boxes around the automated segmentation. The sizes of the boxes are then varied to see how performance changes away from the natural bounding box. Performance is measured as the mean class accuracy to account for the varying number of examples in the dataset. The results for this experiment, when using OpponentSURF features, are shown in fig. A.7.

In all cases, we see that as the bounding box increases in size, performance degrades as would be expected with more noise in the system. This effect can be partially mitigated by geometric reranking because that will filter out features that are not geometrically consistent, which happens more in background regions.

For some of the human segmented images, there is higher performance when the bounding box is smaller than the region identified. Since fish are sinuous, a bounding box around their entire area will include a lot of background. Thus, using a smaller box can decreases the number of features from the background and improve the signal to noise ratio. We see the opposite effect in the automated segmentation because those boxes are already very small. As they shrink even more, less features are extracted from the fish,



(a) Machine

(b) Human

Figure A.8: The accuracy of the classifier for different fractions of queries returning confident results. Results are shown for both the machine and human datasets

lowering the signal and hurting performance. These results indicates that it is very important to have a good segmentation of the fish. We would also expect improved performance with a tight segmentation instead of a simple bounding box.

A.5.3 Feature Types

In order to determine the effectiveness of different types of features, experiments were conducted using the features described in appendix A.4.3.1. The results are shown in fig. A.8. All of the features show performance significantly better than random, while the best feature set is the simplest: a histogram of HSV colors. Fish are very colorful and thus, it is not surprising that color alone has some discriminative power. The interest point detectors performed worse and is most likely due to the small number of interest points that can be found in the region of a fish. Typically, only around 15 are found during the automatic segmentation and 60 are found using the human segmentation. There is also a slight improvement to using color based SURF descriptors, with OpponentSURF and CSURF showing very similar results. This result, along with the reasonable performance of an HSV color histogram, indicate that color is a crucial component to any automated fish classification system.

A.6 Conclusions

We have built Reefbot, a robotic underwater vehicle, to act as an ambassador of the reefs to patrons at the PPG aquarium. This platform is also being leveraged to develop algorithms for automatically identifying fish species, which has proven to be a very formidable problem. We have presented a challenging dataset along with initial results from a segmentation and classification system that shows promise for classifying fish species in natural environments from moving platforms. Through our analysis, we have demonstrated that a good segmentation and the use of color information is crucial for any algorithm to reliably identify fish in a natural setting using cameras.

A.7 Acknowledgements

The work in this chapter was conducted with the help of the Reefbot team: Michael Furlong, Scott Moreland, Ashley Kidd, John Thornton, Justine Kaznica, Bob Snowden and David Wettergreen. We would also like to thank the Spark Fund, VideoRay® and the Pittsburgh Zoo and PPG Aquarium for their support in creating the Reefbot exhibit. We would also like to acknowledge the help in labeling the images provided by the anonymous workers on Mechanical Turk.