The Pennsylvania State University

The Graduate School

Department of Computer Science and Engineering

ENABLING INTELLIGENT VISION SYSTEMS IN A CONFIGURABLE MULTI-ALGORITHM PIPELINE

A Dissertation in

Computer Science and Engineering

by

Matthew Joseph Cotter

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2015

The dissertation of Matthew Joseph Cotter was reviewed and approved* by the following:

Vijaykrishnan Narayanan
Professor in the Department of Computer Science and Engineering at The Pennsylvania State University
Dissertation Advisor
Chair of Committee

Mary Jane Irwin
Evan Pugh Professor in the Department of Computer Science and Engineering at The Pennsylvania State University

John Sampson
Professor in the Department of Computer Science and Engineering at The Pennsylvania State University

Mary Beth Rosson
Interim Dean of the Department of Information Sciences and Technology at The Pennsylvania State University

John P. Sustersic
Special Member
Research Associate at the Applied Research Laboratory, Intelligent Systems Department, Autonomous Control & Intelligent Systems (ACIS) Division

Steven P. Levitan
Special Member
John A. Jurenko Professor of Computer Engineering in the Department of Electrical and Computer Engineering at The University of Pittsburgh

Lee Coraor
Head of the Graduate Program in the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School

ABSTRACT

The machine vision community has expended tremendous effort in the research and development of algorithms in an effort to develop a system that is capable of seeing the world as humans do. These algorithms often focus on the accomplishment of specific tasks analogous to human vision such as scene awareness, object detection, object recognition, and object tracking. Joining forces with cognitive neuroscientists has steered much of this research towards the development of algorithms that not only accomplish the required tasks, but endeavor to do so in a biologically inspired fashion. Still, development and evaluation of these so-called neuromorphic algorithms is often done in isolation, with little regard given to the rest of the system necessary to make this human-like system a reality.

This dissertation provides a framework for the current and future development of complex and highly integrated multi-algorithm vision systems. This framework not only enables the composition of such systems, but enables seamless development and integration of improved algorithmic modules. In addition to this high-level system composition framework, the Cerebrum tool, targeted at development of hardware-accelerated architectures is detailed in this work. This tool enables the creation of such hardware-based accelerators by researchers and engineers without specific or detailed knowledge of the target hardware platform.

In addition to the framework and tools, this dissertation also details the analysis, development and evaluation of hardware accelerators for HMAX object recognition and AIM saliency detection. Armed with this intelligent framework and algorithmic accelerators, demonstrations of vision systems that leverage multiple algorithms are constructed and evaluated.

Hierarchical object classification, leveraging the benefits of Exemplar SVM and accelerated HMAX is shown to provide performance superior to either algorithm in isolation.

Furthermore, a more complex system, targeting the domain of personal retail assistance is composed and demonstrated for the benefit of visually impaired persons.

With an eye towards future systems, this dissertation also serves to evaluate and explore a number of technologies whose time is coming. New transistors, such as Tunnel FETs, and novel architectures, such as coupled oscillator arrays, are examined to identify benefits and concerns in their use for the development of future visual systems, both at the algorithmic and circuit/component level. This work also explores the potential for inclusion of additional data modalities, such as audio, for a more effective understanding of scene awareness.

The flexibility of the framework described here enables the inclusion of these emerging devices, architectures, and modalities alongside traditional software and hardware-accelerated implementations within a unified system in order to develop, evaluate, and deploy all of the components required for any given visual system.

TABLE OF CONTENTS

List of Figures vii

List of Tables ix

Acknowledgements x

Chapter 1 Introduction 1

Contributions of this Dissertation 2

Chapter 2 Vision Algorithms 5

2.1 Scene Context & Awareness 7
GIST 7
2.2 Attention & Visual Saliency 9
Attention based on Information Maximization 10
2.3 Object Recognition 12
HMAX 12
Exemplar SVM 15
2.4 Object Tracking 17
Consensus-based Matching and Tracking of Keypoints (CMT) 18

Chapter 3 Development of Intelligent Visual Systems 21

3.1 Intelligent Visual Pipeline & Framework 22
Communication 24
Modular Implementation 25
3.1 Algorithmic Accelerator Prototyping Tool (Cerebrum) 26
Cerebrum Front End 27
Cerebrum Back End 29

Chapter 4 Accelerated Architectures 31

4.1 HMAX Accelerators 31
Baseline Implementation 32
S2/C2 Accelerator 37
S1 Accelerator 41
System Evaluation 41
4.2 AIM Accelerator 46

Chapter 5 Case Study: Hierarchically Refined Object Recognition 50

HMAX+ESVM Pipeline 50
Evaluation 54

Chapter 6 Case Study: Object Detection and Tracking for Personal Assistance 58

Detection 60
Boosted Recognition 60
Tracking 61
Feedback 61

Chapter 7 Future Implications of Emerging Technologies and Non-Visual Features 63

7.1 Emerging Technologies 63
Tunnel FETs 65
Non-Boolean Architectures 69
7.2 Multiple Modalities 75
Audiovisual Scene Recognition 75
Lexicovisual Scene Recognition 79

Chapter 8 Summary 83

References 85

LIST OF FIGURES

Figure 2-1. Example of AIM attentional algorithm. 10

Figure 2-2. Computational layers and data flow of the HMAX algorithm 12

Figure 2-3. Example image pyramid generated in preprocessing of HMAX 13

Figure 2-4. Examples of oriented Gabor filters, as used in the S1 layer of HMAX 13

Figure 2-5. Illustration of cross scale pooling in C1 layer of HMAX 14

Figure 2-6. Illustration of several stages of CMT tracking 19

Figure 3-1. Example of Intelligent Visual Pipeline. 21

Figure 3-2. Example of Cerebrum Core definition 27

Figure 4-1. Execution time of HMAX as a function of number of threads 33

Figure 4-2. Speedup of S2 stage of HMAX as a function of multithreading 35

Figure 4-3. Power Consumption of HMAX due to multithreading 36

Figure 4-4. S2/C2 Stage Accelerator for HMAX 39

Figure 4-5. S1 Stage Accelerator for HMAX 41

Figure 4-6. Speedup of GPU- and FPGA-accelerated HMAX 43

Figure 4-7. Power Efficiency of GPU- and FPGA-accelerated HMAX 43

Figure 4-8. HMAX Classification Accuracy for Caltech256 dataset 44

Figure 4-9. HMAX Classification Accuracy for PASCAL VOC2007 dataset 45

Figure 4-10. Algorithmic data flow through AIM algorithm, including structural breakdown 47

Figure 4-11. Acceleration of fully custom AIM accelerator vs other platforms 48

Figure 4-12. Examples of salient regions identified by AIM 49

Figure 5-1. The runtime cost of Exemplar SVM scales with the number of exemplars per class 51

Figure 5-2. Difficulty of feature-based classification, such as HMAX, increases (to a point) with the number of candidate classes 51

Figure 5-3. Intelligent visual pipeline configured for HMAX+ESVM classification 53

Figure 5-4. Per-class recognition accuracy for HMAX on grocery dataset 54

Figure 5-5. Per-class recognition accuracy for ESVM on grocery dataset 55

Figure 5-6. Accuracy of HMAX recognition, as the number of candidate classifications for consideration is increased 56

Figure 5-7. Per-class recognition accuracy for HMAX+ESVM on grocery dataset 57

Figure 6-1. Example of Intelligent Visual Pipeline configured for Personal Assist 59

Figure 7-1. Limitations of Scaling of Vcc and Vt in Silicon MOSFETs 64

Figure 7-2. Transport and Drive Characteristics of n-MOSFET vs. n-TFET 65

Figure 7-3. Unidirectional conduction of n-TFET (Id-Vds) 66

Figure 7-4. Energy-Delay Plot of TFET-based 32-bit Sparse MCC Adder 67

Figure 7-5. D-Latch Based Flip-Flop Design Using FinFETs (left) and TFETs (right) 67

Figure 7-6. Threshold voltage drop (left) and switching overshoot (right) for FinFETs (top) and Tunnel FETs (bottom) 68

Figure 7-7. Effect of Enhanced Miller Capacitance on TFET SRAM Soft Errors 69

Figure 7-8. Effect of electrical masking on soft error propagation 69

Figure 7-9. Oscillator Inputs Derived for Edge Detection 72

Figure 7-10. Edge Detection using Oscillators **(left)** Original Image **(middle)** Canny Edges **(right)** Oscillator Edges 73

Figure 7-11. Oscillator Inputs Derived for Saliency Approximation 74

Figure 7-12. Results of Visual Saliency From left to right: Original Image, AIM Saliency, Full Oscillator Array, Separated Oscillator Arrays 74

Figure 7-13. Evaluation of scene recognition using both visual and audio features 78

Figure 7-14. Change in scene recognition accuracy between visual and audiovisual features 79

Figure 7-15. Exploration of impact of lexical prior weight on accuracy 81

Figure 7-16. Evaluation of scene recognition using both visual and audio features 82

LIST OF TABLES

Table 3-1. Examples of Intelligent Visual Pipeline Modules 22

Table 3-2. Examples of Intelligent Visual Pipeline Module Outputs 24

Table 4-1. HMAX Algorithm Parameters.....32

Table 4-2. HMAX Stage Execution time, as a percentage of total execution time 34

Table 4-3. AIM Accelerator Resource Utilization on Virtex6 FPGA 47

ACKNOWLEDGEMENTS

First and foremost, I would like to acknowledge the tireless guidance and encouragement of my dissertation advisor, Dr. Vijaykrishnan Narayanan. Without his willingness to take me on and guide me, this work, as well as my graduate and post-graduate career would no doubt have been impossible.

I also need to thank the rest of my doctoral committee for their input, guidance, and feedback as I pursued the various aspects of my research and for their confidence both in me and my work as I move forward beyond graduate school.

While all students, faculty, and researchers at the Microsystems Design Laboratory (MDL) at Penn State have been remarkably helpful during my time there, I do need to single out three individuals in particular: Michael Debole, Ahmed Al Maashri, and Kevin Irick. Without the day-to-day help and support from these amazing individuals, I am not sure that I could have made it as far as I have.

Last and certainly not least, I have to acknowledge the patience and sacrifices of my family throughout my academic career. Without their understanding, none of what I have done or will ever do would be possible.

Chapter 1

Introduction

The vision of building a system capable of understanding the world the way humans do has long been a goal of both the machine vision [**1**,**2**] and cognitive neuroscience communities [**3**,**4**,**5**]. The ultimate goal of such a system is to enable the system to autonomously answer questions such as: *"Where am I?", "What is going on?", "What is worth looking at?", "What do I see?", "What objects are present?", "Are things moving?",* and "*Where are they going?*" To humans, these questions are often too simple to consciously consider. However, endowing a computing system with these capabilities has proven challenging. For many years, both communities have attempted to tackle these problems separately. Computer scientists have focused on developing image processing algorithms rooted in disciplines such as information theory and machine learning to create systems to accomplish some of these goals. At the other end of the spectrum, cognitive scientists and neuroscientists have examined mammalian behavior and activity in the brain in order to establish an understanding of exactly how humans process information. It is this observation which drives the development of biologically inspired algorithms.

More recently, research has brought both vision communities together in an effort to create a system that is capable of seeing the world how humans see it. Much of this effort has focused on realizing biologically-inspired implementations of key aspects of the human visual process. Development of such neuromorphic algorithms has focused on a few select aspects of the human visual process such as scene and context recognition [**4**,**6**], attentional awareness [**7**,**8**] and object recognition [**3**,**9**,**10**,**11**]. Much of the work in developing these state of the art

algorithms has focused on their implementation and evaluation in isolation in order to solve the specific problem they address—answering only one, or possibly two, of these questions at a time.

In order to effectively model human perception, however, multiple algorithms must work in concert in order to quickly, efficiently, and accurately provide as much information as possible. Toward that end, this work discusses a unified multi-algorithm pipeline which is capable of enabling such intelligent visual perception systems. The goal of this pipeline is not to model the entire visual system exactly as in humans—not a fully neuromorphic system—but to provide a framework in which modules may be added, removed, configured and tuned to provide an effective answer to all of the questions necessary. Chapter 2 delves deeper into some of the algorithms that are used to enable an intelligent visual system. While there are a wide variety of algorithms which are capable of addressing machine vision problems such as awareness, attention, recognition and tracking, this chapter serves as an overview of the most prominent and widely used algorithms, many of which are further analyzed and employed throughout the remainder of this dissertation.

Contributions of this Dissertation

The framework for a flexible and configurable pipeline for visual systems is presented in Chapter 3. This framework contributes a foundation for the incremental and iterative development and evaluation of complex multi-algorithm visual systems. This software architecture enables implementation-agnostic support allowing individual modules to be developed and improved independently towards the ultimate realization of the desired system.

In addition to this high-level vision framework, a tool for the graphical development of FPGA-based hardware accelerators has been developed. This tool, Cerebrum [**12**], has not only been used to develop the hardware accelerator for HMAX, detailed within this dissertation, but

has also been made publicly available and used for the development of other hardware accelerators ~~as well.~~ While similar packages have been developed, such as the iLab Neuromorphic Vision C++ Toolkit [**13**], these tools focus specifically on the development of purely software based models of primarily neuromorphic vision algorithms. The Cerebrum tool appears to be the first of its kind to enable drag-and-drop synthesis of vision algorithms targeted towards an FPGA-based hardware architecture.

Furthermore, this work provides analysis of multiple vision algorithms, specifically HMAX and AIM, for object recognition and salient target detection, respectively. The analysis and profiling of these algorithms ultimately served as a basis for the exploration and development of hardware accelerators, detailed in Chapter 4, contributing towards the realization of real-time complex visual systems. In addition to the development of these accelerators, this work provides a foundational evaluation of the benefits of such hardware-accelerators in terms of power and performance, relative to other implementations on general purpose CPUs and graphics processing units (GPUs).

The effectiveness and utility of systems making use of these hardware accelerators and multi-algorithm intelligent vision pipelines are demonstrated within this work as well. The flexibility of the pipeline is shown through the use of cascaded object recognition algorithms. This hierarchical object recognition scheme, detailed in Chapter 5, leverages the speed of the hardware-accelerated HMAX algorithm as well higher as accuracies of Exemplar SVM-based classification in order to improve overall recognition accuracy in a reduced time frame.

This work further demonstrates the potential social contributions of such a complex pipeline through the demonstration of a real-world personal assistance system. The personal shopping assistance scenario, targeting visually-impaired, persons is described in Chapter 6. This system incorporates a number of vision-based algorithms to locate, identify, and track desired products within a grocery store environment. In addition to this information gathering, the

system also demonstrates how this information can be leveraged in order to provide feedback allowing a visually-impaired person to locate and pickup products without additional assistance.

Finally, this work provides an exploration of a variety of new and emerging technologies that may be leveraged in intelligent vision systems of the future. The evaluation of Tunnel FETs, as a candidate for the replacement of traditional MOSFET and FinFET devices, provides a foundation for the development and continued evaluation of these devices as they mature. This work contributes to the examination of novel architectures, such as coupled oscillators, by exploring their capability to effectively provide implementations of a variety of operations and algorithms used within the domain of machine vision and image processing. New sources of information may contribute to the understanding of a visual scene as well. This work explores the application of audio signals which may be present, but often ignored, in video systems in order to improve the capability for complex visual systems to achieve better context and scene awareness.

Chapter 2

Vision Algorithms

Towards the development of an intelligent vision system, many algorithms must work together. This chapter examines some of the questions that need to be answered by such a system, and provides a detailed background on algorithms capable of doing so.

Questions such as "*Where am I?*" and "*What is going on?*" are answered through the use of context recognition algorithms [**6**,**14**,**15**], attempting to capture what is commonly referred to as the "gist" of a scene, the name by which they are often called. These GIST-type algorithms employ a feature extraction methodology in order to construct a feature space for future classification using these algorithms. The features generated by these algorithms are typically useful for quickly identifying a generic location or scenario, such as a city street, meadow, or forest. In addition to recognition of such broad environments, these context awareness algorithms can also enable the identification of specific locations, such as a particular room or aisle in a grocery store.

Attentional algorithms such as AIM [**1**,**2**,**16**] and visual saliency [**7**,**8**] attempt to answer the simple question: *"What is worth looking at?"* While the question is simple, the approach is not. AIM attacks this problem from an information theoretic perspective to develop a measure of self-information or "surprise" found at each pixel in an image. These regions of high information indicate regions worth focusing attention on. Visual saliency approaches the problem from the cognitive science perspective, employing features that are demonstrated to produce attentional responses in humans, monkeys and other mammals. Both of these algorithms work to produce a heat map, indicating relative levels of interest which can be found at various locations in the image.

The most intensive research in visual processing algorithms has been focused towards the field of object recognition. These types of algorithms are designed to answer questions such as *"What do I see?"* and *"What objects are present?"* Many algorithms have been developed to address the problem of answering these questions. Algorithms such as HMAX [**3**,**9**,**17**,**5**,**18**], are modeled after the human ventral stream of visual processing. HMAX extracts features from a particular region of an image by identifying correlations to previously learned features. Others such as Exemplar SVM [**19**] and SURF [**20**] use combinations of feature extraction, brute force matching, and machine learning in order to match visible objects to previously seen templates or examples.

Machine vision has long studied tracking algorithms in order to answer the questions *"Are things moving?",* and "*Where are they going?*" This research has led to the development of such seminal algorithms as basic mean-shift [**21**] tracking and Lucas-Kanade optical flow [**22**,**23**]. Many current state-of-the-art tracking algorithms, such as OpenTLD [**24**,**25**] and CMT [**26**], are built upon these core algorithms, augmented with the inclusion feature detection, feature extraction, and machine learning algorithms.

All of these algorithms are useful in completing a piece of the puzzle on their own. However, the deep understanding that humans are capable of in their perception does not arise through analysis of all of these pieces in isolation. Upon seeing a scene, humans are capable of quickly—virtually simultaneously—identifying the context of a scene, identifying the locations of key objects in a scene. This is quickly followed by identification of what those objects are and how they (or other objects) may be moving within the scene. As understanding of a scene develops, new inferences will be made and conclusions drawn, modulated or reinforced by the previously observed data. For example, being able to quickly identify a scene as a city street can modulate the attention, focusing it towards the ground level areas, such as sidewalks and roads. Additionally, the knowledge of the current location can influence the likelihood of detecting

certain types of objects—lions are not often found on a city street and cereal boxes don't often appear in the vegetable aisle of a grocery store.

2.1 Scene Context & Awareness

Scene and context recognition algorithms are useful in the identification of a location depicted in an image. The specificity of the location may vary, identifying scenes and contexts ranging from a specific location to a general setting. Regardless, scene and context recognition algorithms enable additional inferences to be made not only about what is going on in the scene, but as to the types of objects that may be identified in the image as well [**14**,**15**,**27**].

GIST

A variety of descriptions can be applied to various scenes, much as humans are capable of describing a scene in multiple ways. Many of these descriptors define various visual aspects of an image, often in degrees. By combining these descriptions, a better understanding of the total content of a scene can be achieved. Oliva and Torralba, in [**28**], have presented a method for the holistic description of a scene using a spatial envelope encompassing these types of descriptors. This envelope attempts to quantify the degree of several image descriptors such as naturalness, openness, roughness, expansiveness, and ruggedness. These descriptors lead to features which are capable of describing an image in terms of many human-understandable qualities.

Naturalness captures the apparent synthetic nature of a scene, providing an indicator of whether the location appears man-made (urban) or natural (such as a field or forest). Providing a measure of how contained a scene is, openness describes whether the content of a scene is

bounded. For example, while an area such as a forest or grassy meadow has a high degree of openness, an indoor room such as an office appears to have clearly defined boundaries and a corresponding lack of apparent “freedom”. Roughness distinguishes the complexity of a scene, differentiating between scenes which contain a degree of regularity and smoothness, such as the exterior of an office building versus those which exhibit a high irregularity and lack of smoothness, such as a forest.

Due to perspective, parallel lines in a scene provide a measure of depth based on their level of convergence in a scene. Expansion quantifies this by determining the degree at which vertical and horizontal edges in a scene converge. Scenes with a high depth, and correspondingly high degree of expansiveness, contain lines that significantly converge towards their respective vanishing points. Ruggedness attempts to measure the contours of the scene of the ground level. Measured with respect to the horizon line of an image, this feature readily provides effective discrimination between man-made environments, typically with relatively smooth ground levels, such as floors and streets, and natural environments with highly uneven terrain, foliage, and rocky formations.

These descriptors can all be formulated as a measure of the spatial information in an image, represented at varying spatial frequencies. The algorithm outlined in [**28**] details the application of such filters in order to compute these descriptors. Each descriptor is computed by applying the appropriate filters to the input image. The image is first converted into its corresponding representation in the spatial frequency domain. This representation has a two-fold impact on the computation. First, in the frequency domain, any influence of relative location of objects in the scene is obscured. Second, the application of the frequency-domain filters to the frequency domain image results in highly simplified computation. Rather than defining, or transforming, each descriptor filter in the spatial domain, convolution of each filter requires simple point-wise multiplication in the frequency domain.

Once all of the filters have been applied to the image, a summary of the responses must be extracted to use as features for recognition. In order to accomplish this, the response of each filter is transformed back into the spatial domain. A grid of NxM equal-sized subregions is defined across the image and the average response across each of these regions is computed. A finer grid corresponds to a feature vector more representative of the entire image, at the cost of a larger vector for subsequent processing and increased computation time.

2.2 Attention & Visual Saliency

The baseline models for visual attention essentially strive towards the ideal goal of identifying what region(s) of an image the eyes would be most drawn to look at first. A variety of attentional models have been developed. Bottom-up models [**1**,**2**,**16**,**29**] strive to identify regions of interest (ROIs) within a scene based on global image features. These models derive their measure of attention for any particular pixel or region based solely on statistics and information available within the image. While this approach has validity in identifying human fixations [**1**], there is evidence that human attention is driven by more than what is seen. [**30**] Recent attentional models have shown that human fixations are often modulated by a task-dependent factor. These task-driven, or top-down, attentional models have become increasingly attractive for systems in which the targets are known ahead of time. Both models have uses in intelligent vision systems, with bottom-up attention later influenced by top-down processing.

Attention based on Information Maximization

Saliency (Attention) based on Information Maximization [**2**], dubbed AIM, is a bottom-up attentional model that works to identify regions of interest in an image. These regions are identified using a statistical approach based on information theory. In essence, this algorithm endeavors to assign values to every pixel in an image corresponding to the level of Shannon self-information. These regions found to contain the most information within the image are identified as regions of interest. The kinds of regions identified by this algorithm are found to correspond to those that contain the most surprise, relative to the rest of the image. An example of the results of this algorithm can be seen in Figure 2-1.

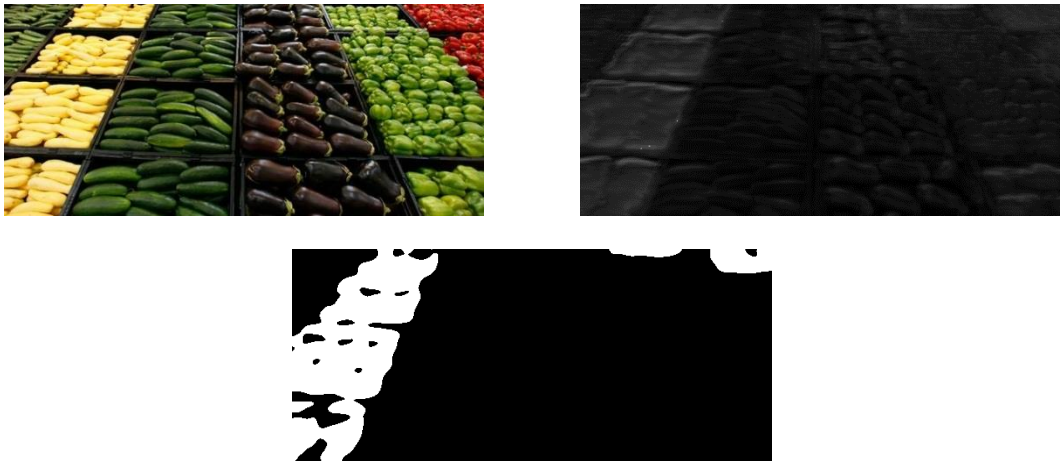


Figure 2-1. Example of AIM attentional algorithm.
(top-left) Input image, (top-right) Saliency map, (bottom) Thresholded ROI mask

The level of self-information contained throughout the image is extracted through the use of a series of convolutional operations. The kernels used for these convolutions are learned *a priori* and form the foundation of the computation of the algorithm. These kernels are derived through a multi-step process. Initially, a random sampling of image patches is taken from a large

set of images. Each of these patches is then evaluated in terms of its capability to extract information. These patches are then evaluated using Independent Component Analysis (ICA) in order to identify the set of patches which produces the most mutually exclusive information. These patches then become the basis functions on which the primary algorithm is based.

The runtime operation of the algorithm can be separated into three stages. In the first stage, each of learned basis kernels is convolved with the image. For multi-channel images, each channel is convolved with the basis kernels independently. After convolutions have completed, the response values are binned into histograms in order to compute pixel-response frequency counts. The pixel-responses are used along with these histograms to compute a probability and then a log-likelihood density estimation. This value estimates the probability of a particular pixel response, and therefore the probability of a given pixel. The summation of all of these likelihoods produces an estimate of the self-information at each pixel, resulting in an information map as seen in Figure 2-1 (top-right).

The exact process of identifying salient pixels from this map may vary. One simple approach is to identify a percentile threshold above which a pixel is considered to be salient and below which is considered irrelevant. Using this threshold, the likelihood value(s) which fall on or around this threshold are identified. This value is then used to binarize the information map into salient and non-salient pixels. An example of this binary map is shown in Figure 2-1 (bottom).

2.3 Object Recognition

HMAX

Object recognition in the human brain is processed in the ventral stream, between the V1 visual cortex and the inferior temporal (IT) cortex [**3**,**11**,**31**]. Based on this biological model, HMAX attempts to mirror the behavior and processing performed along the ventral stream in a biologically consistent manner. Working to match the scale-and rotationally-invariant capabilities of human process, this model builds a set of complex features, derived from simple features. This processing happens in a hierarchical, feed-forward fashion. Figure 2-2 [**32**] shows an example of the computational operations and dataflow of the HMAX algorithm.

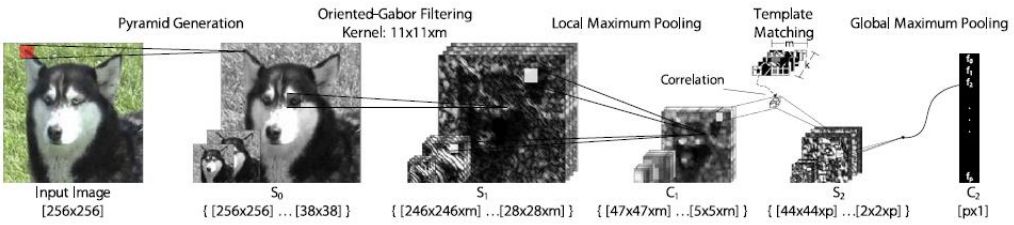


Figure 2-2. Computational layers and data flow of the HMAX algorithm

The first processing step performed in HMAX is a pre-processing step. In order to enable scale invariance, the first stage of the HMAX algorithm generates an image pyramid, representing the original input image at a variety of scales and resolutions, as seen in Figure 2-3. After composition of the image pyramid, the remainder of the HMAX algorithm is comprised of alternating layers of convolution operations and pooling operations, corresponding to the (S)imple and (C)omplex cells found within the visual cortex.

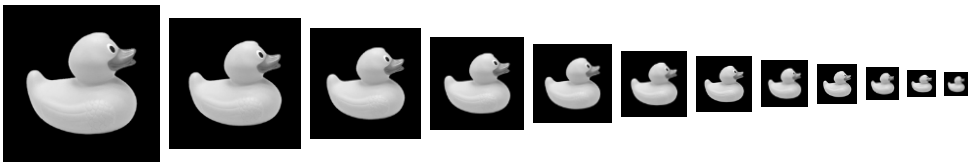


Figure 2-3. Example image pyramid generated in preprocessing of HMAX
The image of the duck is taken from the COIL-20 dataset [citation]

The first of these layers, the S1 layer, enables the rotational invariance of the HMAX algorithm. In the S1 layer, each scale of the image pyramid is convolved with a series of Gabor filters, a widely accepted model of the receptive fields in the visual cortex [**31**]. As in [**33**], this series of filters is designed to be 11x11 in size with dominant orientations equally spaced between 0 and π radians. The remaining parameters—wavelength (λ), effective filter width (σ), and aspect ratio (γ)—are set as described in [**18**] to have values of 5.6, 4.6, and 0.3 respectively.

$$G(x,y) = \exp -\left(\frac{(X^2 + \gamma^2 Y^2)}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda} X\right)$$

Equation 2-1. Gabor filter used in the S1 layer of HMAX processing

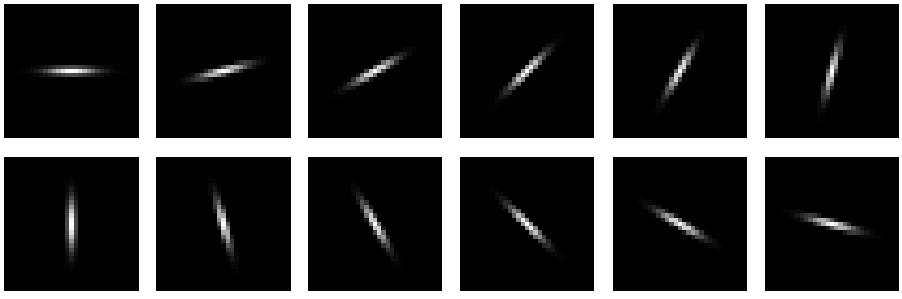


Figure 2-4. Examples of oriented Gabor filters, as used in the S1 layer of HMAX
Orientation angles spaced equally between 0 and π radians

Following the S1 layer, the C1 layer performs localized pooling across S1 output cells. This layer compresses some of the scale invariance introduced in the preprocessing stage by max-pooling values across adjacent scales. In addition to this cross-scale activity, pooling encompasses local regions of these adjacent scales. The resulting output of the C1 layer is a highly condensed set of images, retaining all orientation information generated by the S1 layer.

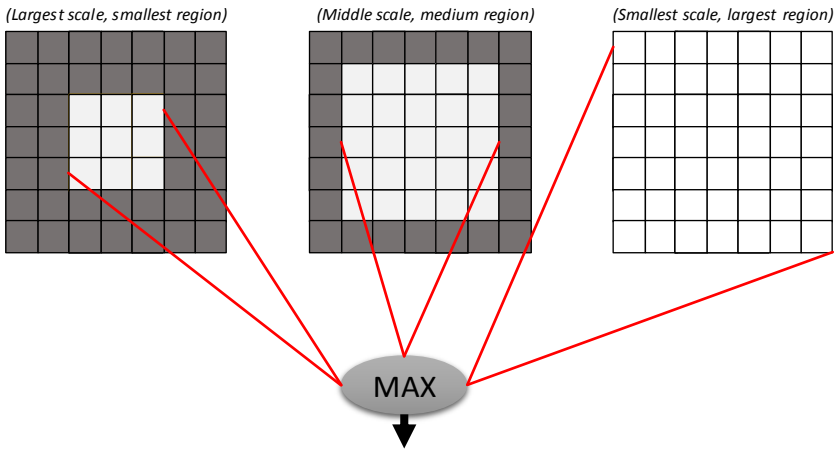


Figure 2-5. Illustration of cross scale pooling in C1 layer of HMAX

The most complex layer of HMAX processing, the S2 layer is modeled on the V4 (posterior IT) region of the ventral stream. In this layer, a large database of tuned features are matched across each scale of C1 output. This database consists of a number of prototype image patches, of varying sizes (4x4, 8x8, 12x12, and 16x16 as in [**10**]). Each of these patches consists of the target feature as seen at each orientation as generated by the S1. The matching function of the S2 may be varied, however, the normalized dot product, as shown in Equation 2-2, has been shown to be an effective matching function [**32**]. The processing of the prototypes through the S2 ensures that each prototype's orientation is matched against the corresponding orientation of C1

output. The output of the S2 layer results in a large amount of data, namely a response image for every dictionary template, correlated at every scale and orientation fed into the S2.

$$R(X,P) = \frac{X.P}{\sqrt{\sum x_i^2 - \frac{\left(\sum x_i\right)^2}{n^2 * m}}}$$

Equation 2-2. Equation describing the Normalized Dot-Product used in the S2 layer of HMAX

The final layer of HMAX processing, the C2 layer, enables global invariance by pooling across all responses for each prototype over a range of scales. By grouping large bands of scales together, this pooling effectively eliminates the impact of large variations in scale. Pooling across the entirety of each response image removes the impact of position within the frame. Likewise, pooling across orientations ensures that the best response for each prototype is found regardless of how the object is oriented in the initial input image. This global pooling results in a large and complex feature set that can be used for any number of machine learning systems for object recognition.

Exemplar SVM

The Exemplar SVM (ESVM) [**19**] model performs object detection based on the relatively simple concept of finding the best match of a particular object within a database of previously learned objects. In support of this, an SVM classifier is trained for each object, exemplar, in the learned database. While the database may contain any number of exemplars for a given class, a unique classifier is trained for each exemplar independently. These exemplars are each represented through the use of a rigid Histogram of Oriented Gradients (HOG) template.

HOG processing is based on gradient computation within the window. Each detected gradient is then employed in a nonlinear weighting operation, allowing each gradient to vote for a particular orientation, based on the magnitude response. These votes are then binned by orientation over small spatial windows within each cell. Contrast normalization is performed across all blocks within the cell, producing a normalized HOG cell response. The complete HOG descriptor is a combined vector of all components of the normalized cell responses from all of the blocks in the detection window [**34**].

Each specialized exemplar classifier is trained by first windowing the object ground truth in such a way as to produce approximately 100 cell windows. While the exact windowing function is different for each exemplar, the size of all cells for any given exemplar are the same. This results in the number of HOG features being different for each exemplar SVM classifier. The training process for each exemplar begins by this window selection, extracting the HOG features from each window. These features serve as the positive training example of the chosen exemplar, X_E. A number of negative training examples are then generated by extracting cells, of the same size as the training exemplar from database objects which do not represent the same class as that of the chosen exemplar. For each of the windows extracted from the non-chosen exemplars, HOG features are extracted. These features serve as the negative training examples of the chosen exemplar, N_E.

Finally, the SVM classifier is trained to learn the feature weights which provide maximal separation of the features represented in X_E from those found in N_E. For each positive exemplar, it is not uncommon to use a very large set of negative examples for training. However, the number of training vectors only impacts the training phase of the ESVM algorithm.

Once trained, an unknown object is processed through each of the trained ESVM classifiers. For each classifier, the cell windowing and HOG feature extraction is performed in

such a way as to match the cell topology of the positive example of the class. This ensures that the resulting HOG feature vector will have the same length as the ESVM classifier.

The detection and classification of ESVM has been shown to be highly effective [**19**]. However, as the classification process must be carried out for each candidate SVM, the runtime performance scales not with the number of classes as many classification schemes, but rather with the number of exemplars. Thus, the ensemble of exemplars can become quite large. The ability to classify a wide number of classes would require processing of the entire ensemble, requiring significant expenditure of time and resources.

2.4 Object Tracking

Machine vision has developed a number of algorithms that can be useful for tracking moving objects. The most basic of these algorithms is mean shift tracking. This algorithm suffers from a number of deficits which later algorithms have tried to address. Deformations and rotations in the object easily create confusion and can cause tracking to fail. Algorithms such as Lucas-Kanade optical flow [**22**] attempt to improve on this by attempting to first identify key components of a tracked object. These components are tracked individually by identifying local regions in subsequent frames which exhibit strong correlation to the component in the previous frame. The net motion of the object can be computed through analysis of the apparent movement of its components. However, this type of tracking may succumb to errors when dealing with objects that are capable of deformation, effectively changing the relative locations between key components while in motion.

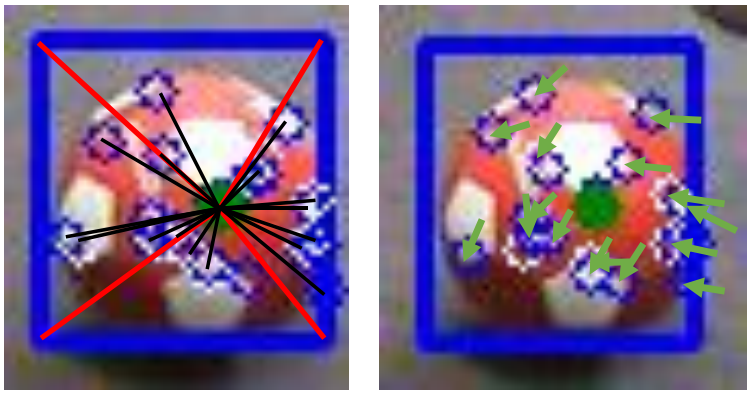
More advanced algorithms have attempted to address this issue through several means. OpenTLD [**25**] employs an online learning model during tracking. This enables a periodic update allowing the algorithm to adapt to subtle changes in the object over time, effectively

compensating for deformations. However, rapid movements or changes in the object without corresponding updates to the model can cause tracking to fail. Once this happens, it becomes difficult for TLD to re-engage tracking with the target.

Another algorithm, Consensus-Based Matching and Tracking of Keypoints (CMT), sacrifices some of this flexibility in order to enable simplified target reacquisition.

Consensus-based Matching and Tracking of Keypoints (CMT)

The CMT algorithm performs object tracking by combining a persistent object model based on advanced keypoints and optical flow. Tracking is initialized by providing the algorithm with the current frame and the region of that frame which contains the object. The entire input frame is processed using the advanced keypoint detection algorithm, BRISK in the default implementation. The set of active keypoints is constructed by identifying all of the keypoints which fall inside the region defining the target object. These keypoints, and their associated descriptors comprise the defined object model. This model, in addition to the keypoints, contains information relating their relative locations and angles, called springs, with respect to the center of the defined object region. This additional information is used for object center, scale, and rotation estimation during subsequent frames, as seen in Figure 2-6a. All remaining keypoints, and their descriptors, are defined to constitute the background model of the scene.



(a) (b)

(c) (d)

Figure 2-6. Illustration of several stages of CMT tracking

(a) Keypoints and springs identified during initialization. **(b)** Optical flow estimation. **(c)** Scale and Rotation estimation using relationships between keypoints. **(d)** Center point estimation using transformed springs

Subsequent frames are processed by first performing keypoint based Lucas-Kanade optical flow (Figure 2-6b) of the active model keypoints into the current frame. Both forward and reverse optical flow are computed to define a degree of error in the tracking of each keypoint. Any keypoint which is not able to be tracked within a reasonable error bound is ignored in the tracking as untracked.

The set of keypoints that are successfully tracked are then used to estimate the relative scale and rotation of the object (Figure 2-6c). This is done through analysis of the relative change in distance and locations between keypoints. After estimation of scale and rotation, these keypoints are then used to estimate the center of the object (Figure 2-6). By using the springs for

each of these keypoints, an agglomerative hierarchical clustering algorithm is employed to allow each keypoint to vote for the location it considers the most likely center. The center of the primary cluster is taken as the estimate for the new object center, after any votes which fall too far outside this cluster are discarded.

After tracking, the algorithm attempts to reconstitute any keypoints which were not successfully tracked by performing another round of keypoint detection and matching against the initial model. It is this step that enables CMT to reacquire an object after it has been obscured, left view, or temporarily changed its form or appearance enough to be lost. For each keypoint in the image, it is matched to the model developed in the tracking initialization through the use of a brute force matching algorithm. While the matcher identifies the best match between keypoints, the algorithm enforces a minimum threshold the degree of this match. Additionally, another constraint is imposed, requiring a high degree of relative confidence in the match, by comparing it to the second-best match as well.

With tracking and matching complete, the set of active keypoints is updated and a new bounding box is computed. In its current form, CMT does not adapt the model of the object over time, so one drawback is that it loses tracking of objects that have a significant change in view over time, such as an object with differing front and rear sides that rotates.

Chapter 3

Development of Intelligent Visual Systems

This chapter details the framework for the development of an intelligent visual system based on a software-architected pipeline. This pipeline presents a unified software architecture for the inclusion, development and utilization of a wide variety of detection, recognition, and tracking modules, among others, necessary to construct such an intelligent system. A visualization of a complex intelligent vision pipeline is shown in Figure 3-1. Despite its grounding in a software implementation, the components of the pipeline are entirely self-contained, rendering the internals of their implementations irrelevant.

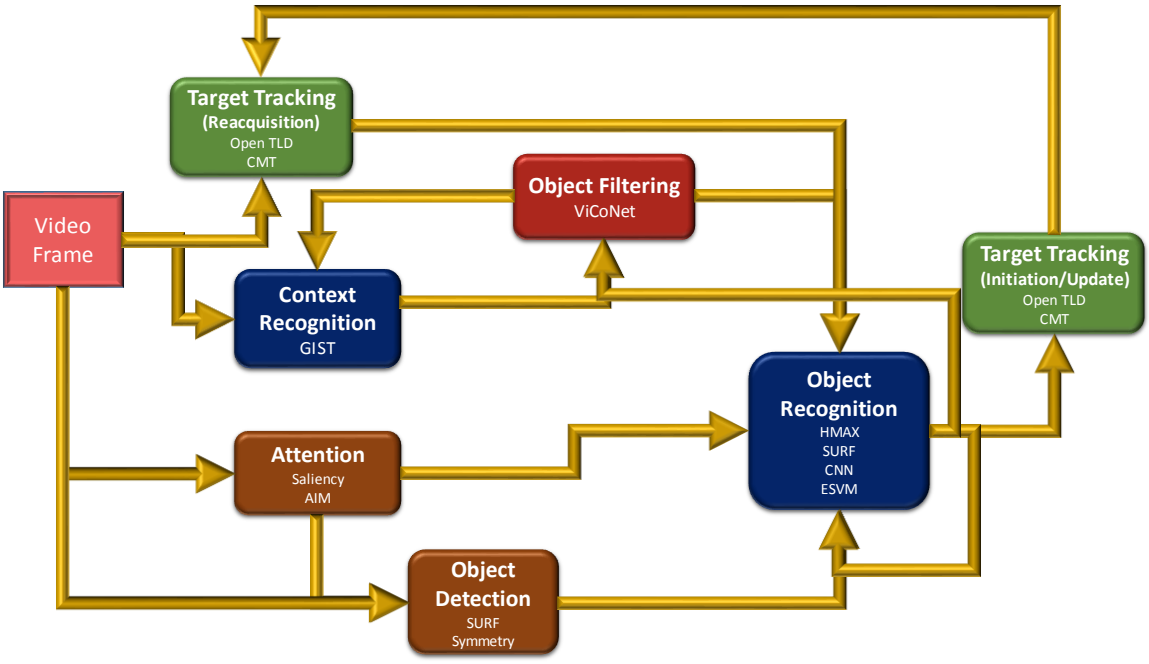


Figure 3-1. Example of Intelligent Visual Pipeline.

A software tool for the prototyping and development of FPGA-based hardware accelerators is also presented in this chapter. This tool presents a hardware-agnostic interface for

research scientists to develop algorithmic accelerators targeting a variety of FPGA hardware platforms. The interfaces to these accelerators can be easily mapped into the intelligent vision pipeline described here, regardless of the exact nature of their implementation.

3.1 Intelligent Visual Pipeline & Framework

The intelligent visual pipeline discussed here is created with several goals in mind. The first of these goals is that the pipeline configuration is highly flexible. The only requirement of components in the system is that they conform to one of several interfaces indicating their general purpose: input, output, detection, feature extraction, classification, filtering, or tracking. Often used together, feature extraction and classification modules may be grouped together into a general class of modules explicitly targeted at self-contained recognition. Examples of the types of components and algorithms that would fall into these categories are listed in Table 3-1. These interfaces are not exhaustive. The dynamic nature of the pipeline facilitates the development and integration of interfaces which have not yet been defined. This enables future support of algorithms and information modules that are not yet known, or known to be necessary.

Table 3-1. Examples of Intelligent Visual Pipeline Modules

Input	**Output**	**Detection**	**Recognition**		**Filtering**	**Tracking**
			Feature Extraction	**Classification**		
Image Set	Display Visualizer	Saliency	SURF	Linear (RLS)	Visual Co-occurrence Network (ViCoNet)	OpenTLD
Video File	Audio Feedback	AIM	SIFT	Brute Force Matching	Top-K Response	CMT
GigE Camera	Haptic Feedback	SURF	HMAX	SVM		OpenTLM
USB Webcam		SIFT	Histogram Of Gradients (HoG)	ESVM		
IP Camera		Symmetry				
Web Stream						

The modular flexibility provided by this framework enables hierarchical solutions to vision-related problems by cascading multiple different algorithms together. Visual GIST-type algorithms, by providing context and awareness, can greatly improve the recognition of objects in a scene. However, the types and configuration of objects found in a scene may be just as telling. For example, if an object recognition module strongly recognizes several objects such as cars, buses, and signs, the system may be able to infer that the location is a city regardless of the result of the context recognition algorithm. In turn, this improved context awareness may enable further refinement of other, less easily recognized objects in the scene.

The cascaded arrangement of modules may also include similar algorithms in order to better solve or reinforce results, such as in the case of object recognition. This is the case explored by the hierarchical recognition pipeline described in Chapter 5. This pipeline employs multiple feature extraction and classification modules alongside a classification filtering module in order to exploit the accuracy and performance characteristics of different recognition algorithms.

With time being a key limiting factor in a real time vision system, it may be often the case where there is insufficient time to sufficiently classify or process all detected objects within a scene. The inclusion of tracking-enabled modules in this intelligent pipeline opens a new means of improving performance by exploiting the temporal dimension. Detected objects can be tracked through frames over time, regardless of whether or not they are given a classification. Over time, as the number of new detections decreases, the slack time may be given to classify these previously unclassified objects. In the event that no objects are found to be unclassified, classification can still provide benefits in the presence of tracking capabilities, however. Objects with weak classifications may be reclassified, possibly with greater effort, in order to improve or reinforce the assigned classification.

The flexibility of this pipeline only requires that those components necessary for the target application be included. The hierarchical recognition described in Chapter 5 presents a simple pipeline, while a more complex pipeline, as described in Chapter 6, utilizes a wider range of components.

Communication

Each module must conform to a particular interface, indicative of the type of operation that it performs or purpose it serves. The distinguishing factors of these interfaces, therefore, lies in the type of data that it produces. A summary of the types of outputs each module produces is given in Table 3-2. Input modules produce raw input data such as frames from an IP camera or video file. These output data streams are tagged with the ID and function of the module that generated it in order to facilitate advanced data flows.

Table 3-2. Examples of Intelligent Visual Pipeline Module Outputs

Input	Output	Detection	Recognition		Filtering	Tracking
			Feature Extraction	Classification		
Raw (image, audio, etc.) data	Visual, auditory, physical feedback	Regions of Interest (ROIs)	Feature Vectors	Classification Scores/Labels	Classification Scores/Labels	Regions of Interest (ROIs)

An event/subscription style model is used by the pipeline to propagate data from one module to the next. This allows each module to register for the type of data stream that is requires. The pipeline architecture is then able to connect each available data stream to any modules which have requested it. In addition to specifying the type of data requested, the module may specify additional constraints or restrictions on the source of the data. For example, a

classification module registered to receive feature vectors and trained on HMAX features should not be sent feature vectors generated by a HoG module. So, the classification module may specify that the type of feature vector must come from an HMAX module.

The type of data a module may request isn't strictly limited by its interface. This allows, for example, an object recognition module to request input streams from other object recognition modules. This, naturally, may result in an infinite loop, so the pipeline is crafted so that a module may not receive its own output directly. Such a loop is still possible if routed through additional intermediate modules, so there is some responsibility on the developer of each module to properly ensure that repetitive input data is eventually discarded without producing a response.

Modular Implementation

The second goal of this intelligent pipeline is to ensure that its operation is wholly agnostic with respect to the underlying implementation of each module. Conceptually, this means that a software implementation of HMAX is no different from a hardware-accelerated implementation of HMAX—they are both feature extractors in the eyes of the pipeline. All classifier modules take in data and produce classification results.

However, it may be of use to know the underlying implementation when constructing a pipeline implementation. Thus, each module may be annotated with additional version or implementation tags that allow intelligent assembly of the pipeline. If a pipeline requires an implementation of the AIM ROI detector, for example, functionally it does not matter whether the module is software-based, or hardware-accelerated. However, the architect may specify that, if available, a hardware-accelerated version be used rather than a software implementation. Ultimately this allows the description of the pipeline to be boiled down to its essence, which is a

series of modules and the connections between them, allowing the system to assemble the most effective version possible given the available components.

3.1 Algorithmic Accelerator Prototyping Tool (Cerebrum)

A number of factors make it difficult for researchers in fields such as neuroscience to quickly and efficiently construct high performance, hardware-accelerated systems to evaluate the algorithms they develop. Firstly, often times researchers in these fields lack the skill set to develop the required hardware accelerators. Additionally, even armed with the necessary accelerators, synthesis of the components into a usable system presents an equally challenging task without additional assistance. Without the tools able to leverage on-chip communication infrastructures, such as networks-on-chip and a lack of standardization across FPGA systems, use of these platforms as a prototyping and development system is seen as far too daunting a task. In an effort to remedy this situation, and bridge the usability gap between computational researchers and engineers, the Cerebrum software tool was developed.

Cerebrum defines and leverages a standardized abstraction across both FPGAs and hardware modules (IP cores) available for them. This enables a clear framework for the creation and modification of an algorithmic dataflow requiring minimal engineering effort. This tool is composed of a front end graphical user interface, exposing the available components and platforms to the user. The back end of the tool automates the processing of several engineering tasks necessary to take the user-defined algorithmic dataflow and map it onto the desired FPGA platform.

Cerebrum Front End

The front end GUI of Cerebrum presents the users with a drag-and-drop style of interface for the development of hardware accelerated algorithms. For the purposes of designing the algorithm, a library of available IP cores are displayed to the user. The visibility and availability of IP cores in the library may be limited by the target hardware platform, thus preventing the user from spending time developing towards a system which ultimately cannot be synthesized due to IP incompatibility. The library of modules is defined by a structured XML-based architecture which enables the hardware developer to expose hardware parameters and control how components and IP cores are connected. The snippet of XML shown in Figure 3-2. Example of Cerebrum Core definition, demonstrates a Cerebrum IP core specification. This example Cerebrum core defines a simple I/O interface core.

```
<Software>
  <DesignDisplay>
     <Category Name=”System Interfaces” />
     <Ports>
        <Port Type="INITIATOR" Name="rx” Interface="bit:128" />
        <Port Type="TARGET" Name="tx" Interface="bit:128" />
     </Ports/>
  </DesignDisplay>
</Software>
<Hardware>
  <Interface Type="SAP" PE="True">sys_iface</>
  <PCores>
    <PCore Type="sys_rx_if" Version="1.01.a" … />
    <PCore Type="sys_tx_if" Version="1.01.a" … />
  </PCores>
  <Clocks>
    <Name=”100MHz Oscillator"… Frequency="100MHz"… />
  </Clocks>
</Hardware>
```

Figure 3-2. Example of Cerebrum Core definition

The `<Software>` section describes the appearance and interface of the core with respect to the front end GUI. This section defines where the core appears in the library of components, and what visible ports it exposes to the user, among other visual properties. The

types of the ports defined here indicate how the core may be interfaced with other cores in the design. Cerebrum defines four types of ports, applicable to different types of cores. Streaming cores utilize ports designated as either `INPUT` or `OUTPUT`. Computational cores expose `INITIATOR` and `TARGET` ports. `OUTPUT` ports may only direct data to `INPUT` ports (of other streaming cores) or `TARGET` ports (of computational cores). Likewise, `INITIATOR` ports may only direct data to an `INPUT` of streaming cores or a `TARGET` of computational cores.

The <Hardware> section defines the details of the hardware components which comprise the Cerebrum Core. In addition to defining the type of the Cerebrum Core (SOP: Streaming OPerator) or SAP (Switch-Attached Processor), this section defines the specific IP cores and any required clocks produced by or required by the component. Other aspects of the hardware covered by this section include importing predefined component interconnects, declaring a limited set of supported platforms and architectures, as well as the estimated resource requirements of the components contained in the Cerebrum core. Armed with the cores available in the library, construction of a system becomes a simple matter of dragging and dropping the necessary components into the workspace and creating the dataflow connections between them. This specification wrapper hides all of the internal details from the algorithm developer.

Once the user has drawn a system on the workspace, the back end of the Cerebrum tool takes over to perform the task of translating the displayed system onto the target platform. In order to achieve this, the tool requires three sets of specifications. The first specification is the design specification, created by the user in the Cerebrum front end. The second specification is that of the target hardware platform. The platform specification defines several low-level details about the reconfigurable platform. This includes the number and types of FPGAs available, how they are connected, both logically and physically, the resources available on each, and any FPGA-specific pin constraints or signals. The final required specification is the Cerebrum project.

This file defines all of the options associated with how and where the EDA tools should be run. Armed with complete specifications for the project, the design, and the platform, the tool may proceed to back end synthesis, translating the provided information into a hardware definition for the target FPGA platform.

Cerebrum Back End

Using these specifications, the Cerebrum back end performs a series of steps in order to realize the desired system on the target hardware. The first step in this process is the mapping of the necessary components to the available FPGAs in the system. A customized accelerator mapping algorithm was developed in order to optimize multiple constraints including minimization of data flow and inter-FPGA communication while maximizing resource utilization. The mapping algorithm proceeds through several phases in order to work towards an optimal solution.

A feasibility check is first performed. This step is designed to short-circuit the algorithm in the event that the system requires more resources than are available on the entire FPGA platform. If, at least superficially, there are sufficient resources to map the required components, mapping proceeds to the component grouping phase.

Advanced users may annotate the design with requirements that some components be co-located on the same FPGA, without regard to the specific FPGA in the platform. Sets of component groups that are so annotated are coalesced into a Component Group. For the purposes of mapping, this group is treated as a single “black box” component requiring the resources of all contained components as well as data connections to all components with connections to the components internal to the group. Once all component groups have been formed, mapping proceeds to I/O distance computation.

Each component, or component group, is annotated with its distance from the system input and output connections. This distance is calculated by finding the average number of hops, or edges, in the available paths between the component and the specific connection. These distances are used to optimize placement of components with shorter I/O on FPGAs closer to the corresponding system I/O port.

Once again, users with advanced knowledge of the target platform may influence the mapping algorithm. During the design process, if desired, each component may be annotated for placement on a specific FPGA. In the event that a component is assigned to an FPGA and assigned to be grouped with another component, the entire component group is then assigned to the FPGA. If the user attempts to assign groups components to different FPGAs an error is generated, advising the user to correct the discrepancy. With a valid pre-mapping, each such component is assigned to its target FPGA, virtually allocating the required resources.

Finally, the optimization of component placement begins. A greedy algorithm is used to find the best target FPGAs for each component. In order to avoid a lack of resources at later iterations of the algorithm, the components and groups with require the most resources are placed first. Optimal placement is determined by finding the FPGA which has the most resources remaining and also minimizes the data traffic between FPGAs. This traffic is measured by finding the number of inter-FPGA hops required for data to flow through the component, if it were placed on each FPGA. This approach continues until either all components are successfully placed, or a component is encountered for which no available FPGA has sufficient remaining resources.

With all components mapped to a target FPGA, the Cerebrum back end generates the files necessary for the proprietary back end tool flow. A system file is generated for each FPGA based on the components that were assigned to it during by the mapping algorithm. Following this, the entire back end tool flow is automated; proceeding through the steps of synthesis, technology mapping, placement and routing, and finally bit stream generation.

Chapter 4

Accelerated Architectures

This chapter provides details on the development of novel accelerators for two visual algorithms. The first algorithmic accelerator described here is targeted for the HMAX feature extraction algorithm. This includes a detailed analysis and profiling of the algorithm, both in parts and as a whole, enabling highly effective and targeted optimizations in the overall architecture. This architecture is composed of multiple accelerators which enable high performance with minimal degradation in accuracy. The HMAX accelerator described here not only represents a novel architecture for the algorithm, but was also established as the foundation for ongoing research as part of the DARPA Neovision2 [**35**] project.

Further extending the family of vision-oriented accelerators, this section also details the development of an attentional model accelerator based on the AIM algorithm. The AIM accelerator described here was developed for incorporation into the visual processing pipeline of the Supervised Autonomous Fires Technology (SAF-T) [**36**] program under the Office of Naval Research.

4.1 HMAX Accelerators

The HMAX algorithm, designed for function in an effort to mirror the biological responses in the brain, displays several inefficiencies when executed on a general purpose processor. These inefficiencies are a result of the computational structure of the algorithm and result in less than optimal runtime performance. In order to effectively utilize HMAX as a feature extraction engine for object recognition in a real time intelligent visual system, a detailed study of

the algorithm is necessary. This study, detailed here, identifies many of these inefficiencies within the general purpose CPU implementation of HMAX, thus enabling the development of effective and efficient hardware accelerators enabling dramatically improved performance of the HMAX algorithm.

As detailed in Chapter 2, the implementation of HMAX used in this dissertation is based on the variant HMIN [**33**]. This implementation defines a number of parameters for this particular implementation of the HMAX algorithm, which are summarized in Table 4-1. In order to establish firm baselines for performance purely in software, the C++ implementation was first augmented to support multiple threads, enabling performance gains through thread-level parallelism.

Table 4-1. HMAX Algorithm Parameters

Parameter	**Value (Range)**
Pyramid Scales	8 to 12
S1 Orientations	4 or 12
C2 Pooling Overlap	1 Scale
S2 Dictionary Templates	5120 templates
S2 Dictionary Template Sizes	4x4, 8x8, 12x12, 16x16
C2 Pooling Range	6 Scales [11-6, 6-1]

Baseline Implementation

The evaluation platform for the multi-threaded software implementation of HMAX consisted of an Intel-based system equipped with dual 2.4GHz Quad-Core Xeon-class processors with a total of 12 GB of main system memory. Each of the four cores of the Xeon processor were HyperThreading-capable, providing 2 logical CPUs per core, resulting in a total of 16 processing units. The software was compiled with maximum optimizations and the SSE2 SIMD instruction

set enabled for further increased performance. Figure 4-1 demonstrates the impact of multithreading on the runtime performance of the software.

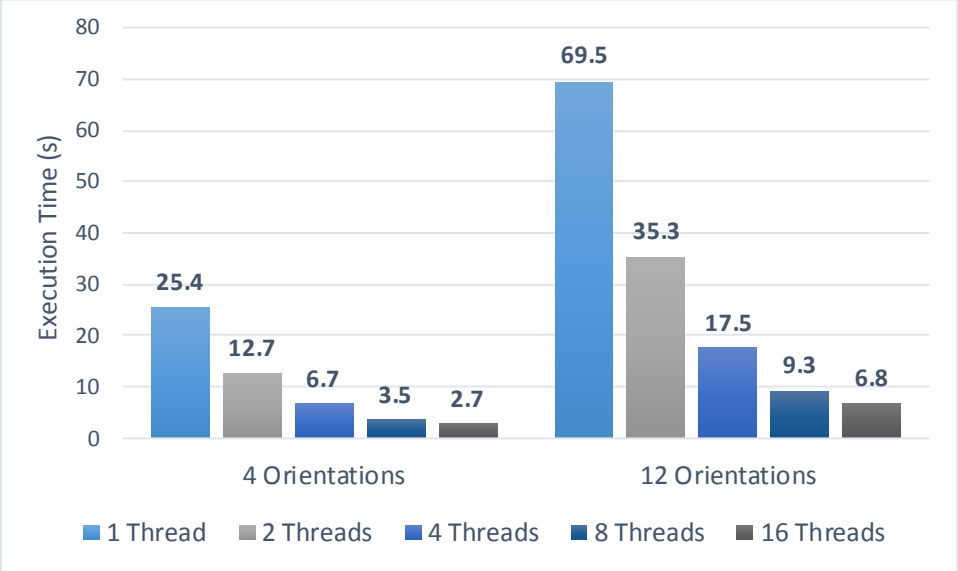


Figure 4-1. Execution time of HMAX as a function of number of threads

In addition to the number of available threads, the runtime performance was also found to depend, nearly linearly, on the number of orientations to be processed. Reducing the number of orientations by a factor of 3, from 12 down to 4, the execution time of the algorithm was reduced by a factor of 2.5 to 2.8, depending on the degree of multithreading enabled. The impact of thread-level parallelism is clearly evident in the observed execution times, wherein a 2X increase in the number of threads often results in a nearly 2X increase in performance. However, increasing the number of threads from 8 to 16 produces only a 1.3X improvement in execution time.

This reduced performance benefit is explained by analyzing the impact of multithreading on each stage of the HMAX algorithm. As shown in Table 4-2, the S2 stage of processing thoroughly dominates the runtime of the algorithm. In fact, the S2 computation averages over 96% of the total execution time with virtually no regard to the number of available threads or

orientations being processed. This indicates that the vast majority of the performance gains found due to multithreading can be found within the S2 stage.

Table 4-2. HMAX Stage Execution time, as a percentage of total execution time

# Threads	# Orientations	S1	C1	S2	C2
1	4	1.62	0.19	97.61	0.53
	12	1.77	0.21	97.82	0.19
2	4	1.69	0.22	97.45	0.56
	12	1.79	0.22	97.78	0.19
4	4	1.89	0.30	97.15	0.50
	12	1.90	0.26	97.60	0.17
8	4	3.54	0.58	94.97	0.60
	12	2.96	0.40	96.27	0.25
16	4	4.71	0.87	93.26	0.76
	12	3.08	0.56	95.92	0.29

Further profiling of the S2 stage specifically reveals the reason for relatively reduced performance gains when increasing the number of parallel threads from 8 to 16. As the number of threads increases beyond the number of processed orientations, the ability to exploit the available parallelism dramatically decreases. As the critical path of the S2 is comprised of correlation of input images with prototypes in the S2 dictionary, exploitation of parallelism at this level is difficult. The largest performance gains come from the thread-level parallelism at the granularity of processing one orientation/scale being processed per thread. Maximal orientation-level parallelism is achieved at the point in which the number of threads equals the number of orientations, 4 and 12 respectively. Further increases in available threads are able to exploit scale-level parallelism in the processing of the S2, but ultimately the reaches the point of diminishing returns, as seen in Figure 4-2.

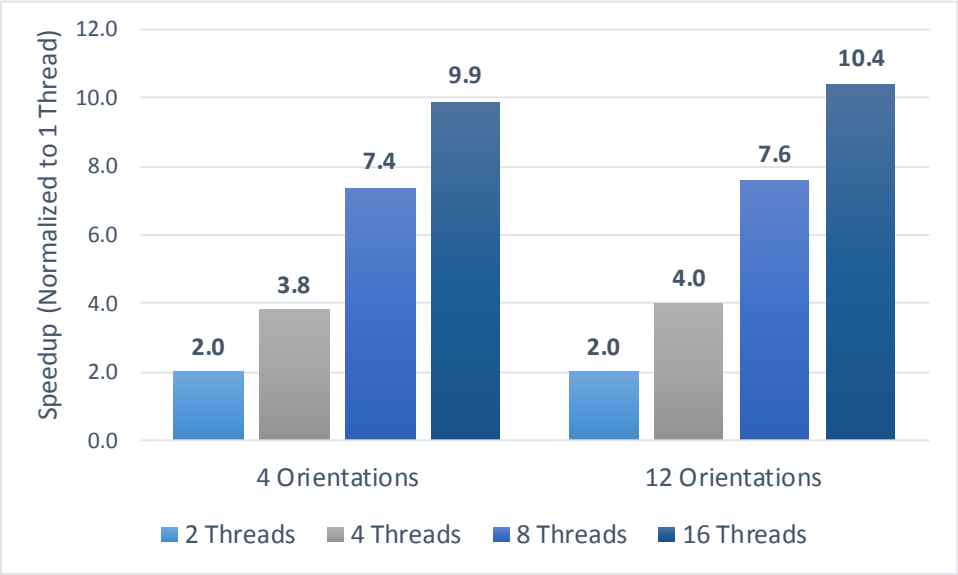


Figure 4-2. Speedup of S2 stage of HMAX as a function of multithreading

As performance is not the only concern in many systems, such as embedded systems, power consumption must be taken into consideration as well. For each of the algorithmic and multi-threading configurations described here, the runtime power consumption of the system is measured as well and shown in Figure 4-3. For this evaluation, power was measured by using a power meter, which provides both continuous and instantaneous power measurements during runtime.

The results here clearly demonstrate the benefits of such parallelism. However, further performance improvements will require application specific acceleration. Specifically, aspects of the algorithm shown to be least influenced by the enabled parallelism, such as convolution operations, require a targeted approach. Due to its enormous influence on the overall execution time of the algorithm as well as the prevalence of convolution operations within, the S2 represents the stage which can gain the most benefit from targeted acceleration.

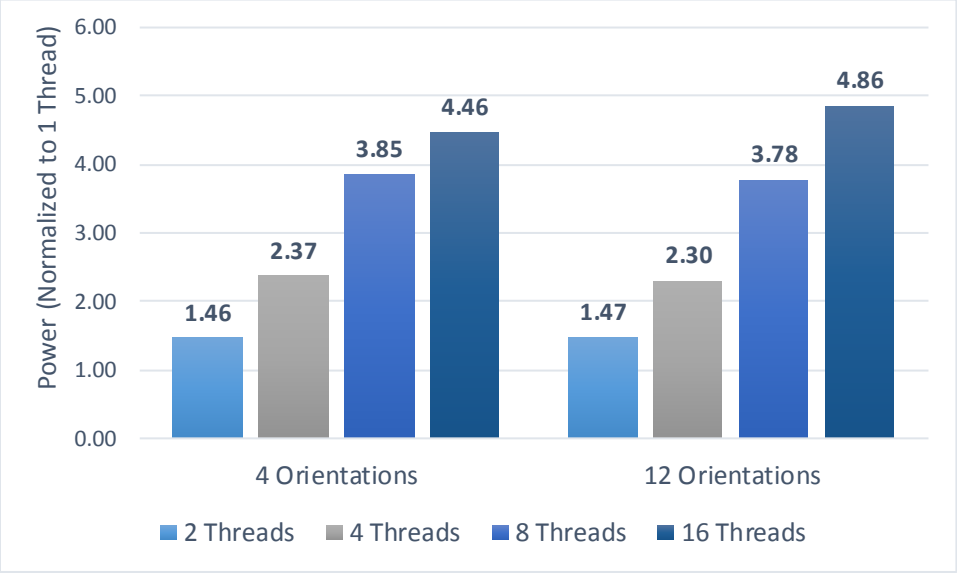


Figure 4-3. Power Consumption of HMAX due to multithreading

The development and evaluation of the hardware modules used for accelerating HMAX was done on a multi-FPGA platform. This platform consists of four Xilinx Virtex5 SX240T FPGAs connected together using Low-Voltage Differential Signaling (LVDS) interconnect. The four FPGAs, are in turn connected to the development system through the Front Side Bus (FSB) in a processor socket. The host system is the same Quad-Core 2.4GHz Xeon processor that the software implementations of HMAX were developed and evaluated on. In order to facilitate communication between accelerators, a packet-switched, high-bandwidth Network-on-Chip (NoC) architecture is utilized [**32**]. This framework enables runtime reconfigurability of the network as well as stream- or compute-based accelerators. The dataflow of this network is highly fluid as well, allowing the construction and use of virtual data paths, or flows, enabling accelerators to pass data through a pre-planned series of accelerators to enable a variety of algorithms using common modules.

S2/C2 Accelerator

The core function of the S2 stage is a correlation operation, akin to template matching, computed between the outputs of the C1 stage and each of the stored prototypes in the S2 dictionary. This correlation operation has to be performed between every scale of the C1 and every prototype in the dictionary and every scale being processed. This requires an enormous amount of computation and produces a very large amount of data. Examination of the data flow between S2 and C2 shows that the massive amount of data generated by the S2 is quickly and radically pooled into a much smaller feature space. Detailed analysis of exactly how this happens leads to the realization that several benefits can be realized by combining the S2 and C2 stages into a single unified module.

The C2 stage finds the maximal response of a particular prototype regardless of orientation and within a limited range of scales. The order these values are generated is irrelevant—only the maximal value matters. This means that the max-pooling operation can be performed on-the-fly as each new prototype response is generated, by comparing the new value and the existing pooled value. In addition to the consolidation and optimization of the pooling operation performed by the C2, this has the added benefit of dramatically reducing the amount of data that must be transferred over the NoC between the S2 and C2 modules. The degree of this reduction in data transfer is quantified by Equation 4-1. In this equation, the numerator corresponds to the data values that would be sent over the network to the C2 without this optimization. S corresponds to the index of the S2 scale being processed, X_S and Y_S indicate the size of the S2 output at scale S. $N_{prototypes}$ indicates the total number of prototypes in the S2 dictionary, and $N_{prototypes}[X_S,Y_S]$ indicates the number of prototypes that can be successfully correlated with the template. This indicator is required to account for the fact that the larger prototypes are too large to correlate with the smaller scales, resulting in no values to pool under

these conditions. The denominator of this equation represents the amount of data that is sent over the network from the combined S2/C2 module with this optimization. It boils down to the number of prototypes in the dictionary—1 max value per prototype—times the number of scale bands over which pooling is done, in this case 2.

$$\frac{\sum_{S=0}^{S-1}\{(X_S \times Y_S) \times N_{prototypes}[X_S, Y_S]\}}{N_{prototypes} \times 2}$$

Equation 4-1. The degree of data transfer reduction between S2 and C2 of HMAX by combining the two modules into a single accelerator

Using the default HMIN configuration parameters, consisting of 12 scales and 12 orientations, along with an S2 prototype dictionary of 5120 prototypes, this equation indicates a data transfer reduction by a factor of 4,154X.

Further reduction in network traffic and data transfer is possible through clever management of the input data to the S2. The correlation of inputs with prototypes requires frequent accesses to the input data for each orientation as it is required for processing of each prototype's corresponding orientation. Rather than requesting this data repeatedly from an off-module memory, all orientations for a given scale are loaded into local memory once and then fetched, as needed, for processing. This optimization results in a 5,005X reduction in data transferred, using the same configuration based on the calculation shown in Equation 4-2. The symbols in this equation are the same as those in Equation 4-1, with $N_{orientations}$ representing the number of orientations being processed.

$$\frac{\sum_{S=0}^{S-1}\{(X_S \times Y_S) \times N_{orientations} \times N_{prototypes}[X_S, Y_S]\}}{\sum_{S=0}^{S-1}\{(X_S \times Y_S) \times N_{orientations}\}}$$

Equation 4-2. The degree of data transfer reduction to S2 through the use of input scale buffering

The architecture of this combined and optimized S2/C2 module is shown in Figure 4-4 [**37**]. Each input scale, including all applicable orientations, is buffered in the image memory. The dictionary of prototype patches are preloaded into an attached on-chip SRAM memory. This memory structure is necessary due to the large size (24MB) of the prototype dictionary in use.

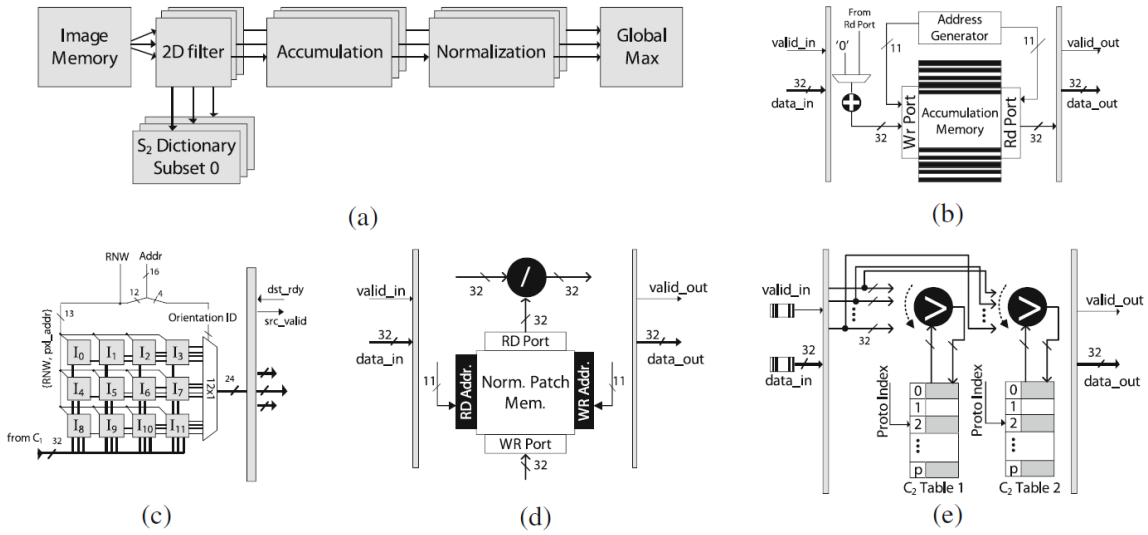


Figure 4-4. S2/C2 Stage Accelerator for HMAX

(a) High-level view of S2/C2 Accelerator Pipeline. **(b)** Pixel-wise accumulation across correlation responses within each scale. **(c)** C1 outputs are buffered in a local image memory, referenced by orientation. **(d)** Accumulated outputs are normalized after computation. **(e)** Global max-pooling is performed on a per-pixel basis as outputs are generated.

An array of systolic 2D convolution units are supplied inputs from this image memory. The coefficients of the convolution units are loaded from the local prototype dictionary for each prototype being processed. The size of operation for each convolution unit is configurable to support the various prototype sizes in the HMAX implementation. The size of the active operation is changed by the control unit as new prototypes are loaded from the dictionary memory. The access latency of the dictionary memory is obscured by overlapping the computation of the convolution operations with the loading of prototypes.

A temporary memory structure is used to store the accumulated output of each orientation's convolution. Upon initiation of each new scale, this memory is reset. As each value is generated from the convolutional units, the corresponding location in the accumulation memory is read, updated, and rewritten. Once processing of the particular convolution is complete the resulting values are forwarded ahead to the normalization.

The normalization unit contains a small memory buffer that is preloaded with coefficients used to normalize the convolution of each orientation's template. These values are used to compute the normalized pixel-wise output for each prototype which is forwarded to the pooling unit.

As the outputs of each scale are generated for each prototype, they are fed into the global pooling unit. This unit identifies the maximum response from among all of the inputs. This max value is then compared to the existing values in the corresponding entry of the C2 data table. If the new value is greater, the C2 table entry is replaced with the new value.

Once all prototypes have been processed, a read request to the S2/C2 accelerator returns the feature vector contained in this C2 table.

S1 Accelerator

The S1 stage is responsible for processing the oriented Gabor filters over the input image scales. The accelerator for this stage is designed to compute these convolution operations in a streaming fashion, producing output pixels as soon as they are ready. The architecture of this accelerator is composed of FIFO which converts the serial inputs to a parallel stream for convolution. A 2D convolution engine that follows takes the parallel stream and Gabor coefficients from a local memory to produce a series of outputs that are fed into an adder tree structure. The serial outputs of this adder tree are the streaming outputs of the S1 accelerator. Figure 4-5 [**37**] shows an overview of this S1 accelerator architecture.

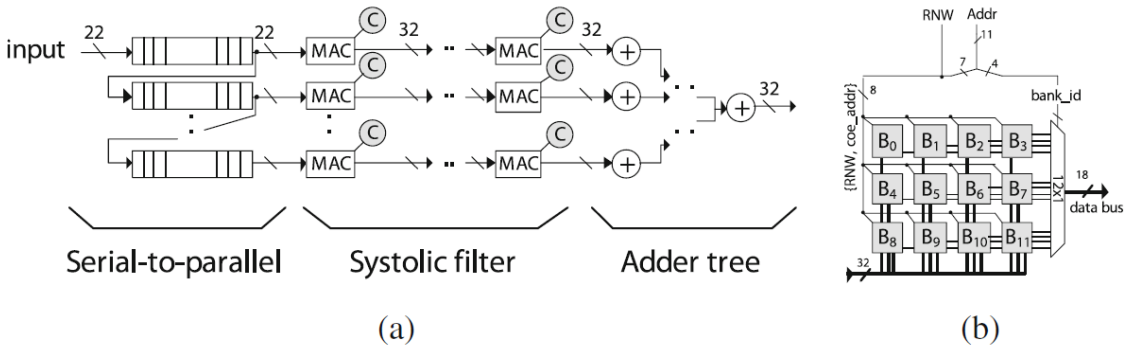


Figure 4-5. S1 Stage Accelerator for HMAX

System Evaluation

Graphics Processing Units (GPUs) are another popular architecture for high-performance systems. These devices enable high levels of acceleration by exploiting parallelism through the utilization of a massive number relatively simple, but high performance computational resources. In order to establish a well-rounded comparison of this FPGA-accelerated architecture, a GPU-based implementation of HMAX [**38**] was evaluated in tandem. The GPU evaluation was

performed on an NVIDIA Tesla M2090 [**39**], which houses a 1.3GHz Tesla T20A GPU with 1.3GB of memory. The GPU is hosted on a system which has 49GB of system memory and a 12-core 3GHz Xeon-class Intel processor.

In addition to runtime performance, power is a critical aspect in the evaluation of these platforms. For the CPU and FPGA power measurements, both idle and active power consumption was measured using a power meter. The GPU power consumption was measured using tools provided by NVIDIA to query power measurement sensors located on the GPU. For these platforms, the idle power is removed, with only the power due to computation being considered.

Figure 4-6 shows the relative speedups achieved by both the GPU- and FPGA-accelerated implementations of HMAX in terms of frames per second (fps). For both the configurations using both 4 and 12 orientations, both of these platforms easily outpace the performance of the software implementation. The FPGA architecture provides significant gains over the CPU implementations of 7.3X and 8.0X for 4 and 12 orientations, while exhibiting only modest gains—1.1X and 1.2X respectively—over the GPU implementation. From these results, it appears that the GPU is almost as good as the FPGA.

However, execution time is not the only metric considered. Figure 4-7 adds the impact of power consumption to the equation. This figure shows the relative performances of these platforms in terms of frames per second per Watt (fps/W). This metric incorporates power by estimating the power required to achieve a particular level of performance. Due to the high power consumption of the GPU, the FPGA implementation easily pulls away in terms of power efficiency. The FPGA outperforms the CPU for 4 orientations in this metric by a factor of 10.9X and by 13.8X for 12 orientations. Similarly, the FPGA beats the GPU by factors of 2.3X and 2.6X for 4 and 12 orientations, respectively,

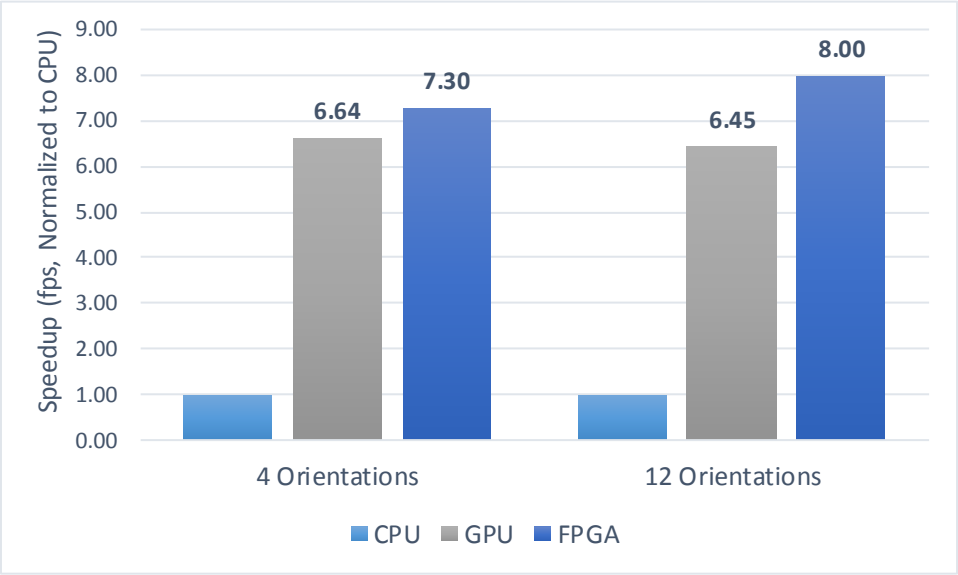


Figure 4-6. Speedup of GPU- and FPGA-accelerated HMAX

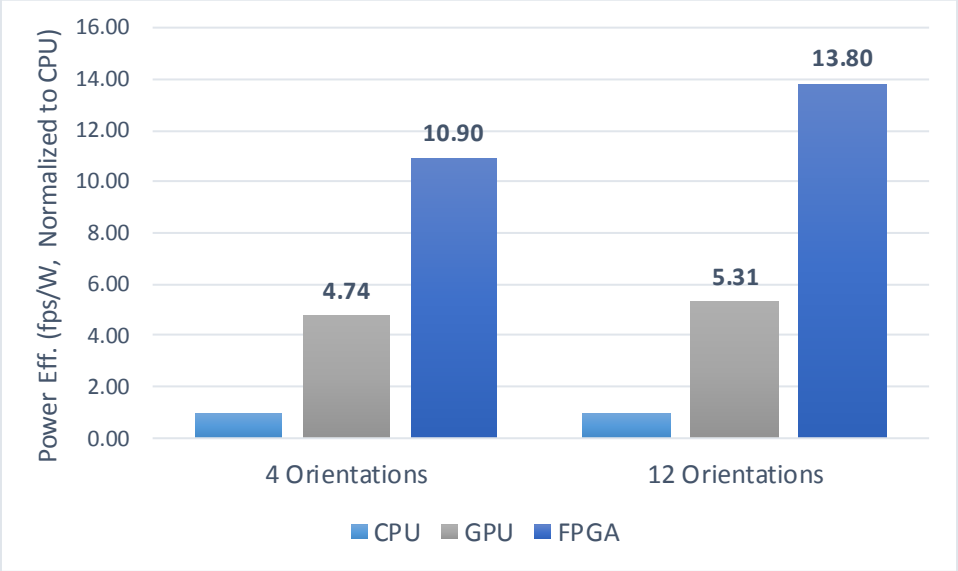


Figure 4-7. Power Efficiency of GPU- and FPGA-accelerated HMAX

Furthermore, the impact of this architecture on the algorithm's performance in terms of accuracy must be considered. Evaluations of the classification accuracy of the accelerated HMAX

were performed using both the Caltech256 [**40**] and PASCAL VOC2007 [**41**] datasets. For the evaluation of the Caltech dataset, the number of scales was fixed to 12. However, the evaluation was performed using both 4 and 12 orientations. The number of training images for each category was varied from 5 to 40, while the number of test images per category remained constant and non-overlapping with the training set in each case. The results of these evaluations are shown in Figure 4-8 [**37**]. Using 40 training images, HMAX is capable of achieving accuracies of 23% and 25% for 4 and 12 orientations. These results fall within 15% of the best reported accuracies at [**40**] on this highly difficult and varied dataset.

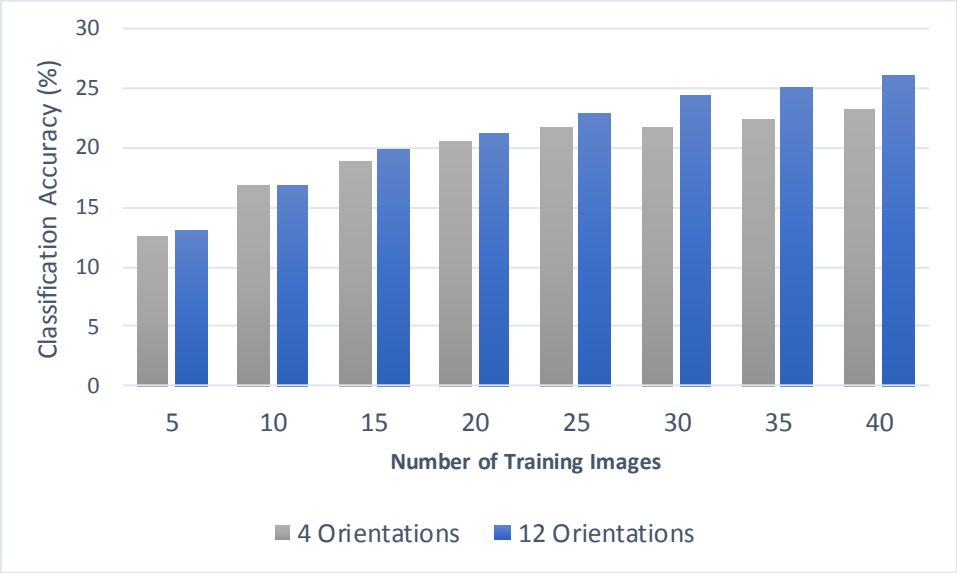


Figure 4-8. HMAX Classification Accuracy for Caltech256 dataset

The PASCAL VOC2007 dataset was evaluated using k-fold cross validation. Due to the large size of the dataset, consisting of over 15000 object instances in 5011 training images, an independent multi-class voting scheme was used utilizing 10 independently trained classifiers. The training data for each object class was independently divided into 10 non-overlapping training sets. After training, the evaluation was performed on the test data, by allowing each

classifier to vote for its most likely candidates, and synthesizing a probability for each class given the voting scheme. These classifications and probabilities were then used to generate the average precision data shown in Figure 4-9 [**37**].

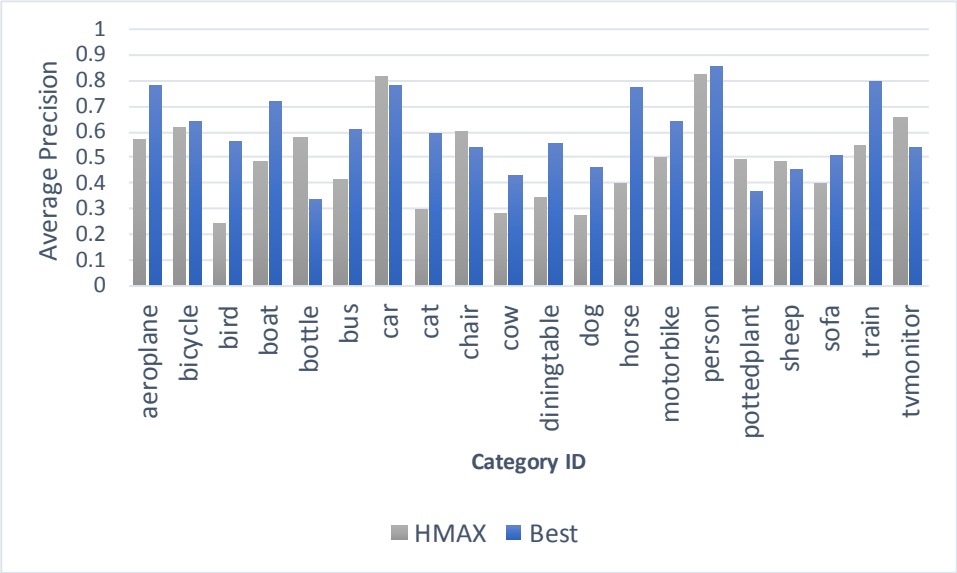


Figure 4-9. HMAX Classification Accuracy for PASCAL VOC2007 dataset

The PASCAL dataset was again used to evaluate the impact of fixed-point hardware precision on the relative classification accuracy. The HMAX algorithm was used to generate features in both the floating-point software implementation and the fixed-point FPGA implementation. Independent classifiers were trained on both of these datasets and evaluated using features similarly extracted from the test datasets. The truncation of floating-point data to fixed-point precision was found to cause less than 2% degradation in the accuracy of classification.

4.2 AIM Accelerator

While, the details of the AIM algorithm are described in Chapter 2, the core computations of the algorithm can be broken down into three phases. The first of these involves the convolution of the input image with the defined basis filter functions. For multi-channel imagery, each channel must be convolved with the basis functions separately. The second stage of the algorithm is the computation of the probability density estimation. This estimation is performed by first binning the pixel responses across basis functions into a histogram. These histogram frequency counts are then used to convert each pixel response into a log-likelihood through the use of a logarithm operation. Finally, the log-likelihoods are aggregated on a per-pixel basis to generate the information map. Additional processing may be used to translate this information map into a binarized version indicating which pixels are considered sufficiently salient. This algorithm flow and breakdown is given in Figure 4-10.

The AIM accelerator was developed initially on a Xilinx Virtex6 SX475 series FPGA. Several customized architectural modules were developed for this accelerator: a convolver for the basis filter operations, a histogramming module, a logarithmic computation module and an output aggregator. The accelerator module returns the information map, while the threshold computation and pixel masking is currently left to the host system, if desired. The fully customized accelerator incorporates multiple pipelines, each consisting of one instance of each module, except for the aggregator which is only instantiated a single time. The overall resource usage of the Virtex6 FPGA for a 4-pipeline implementation of this customized accelerator is shown in Table 4-3.

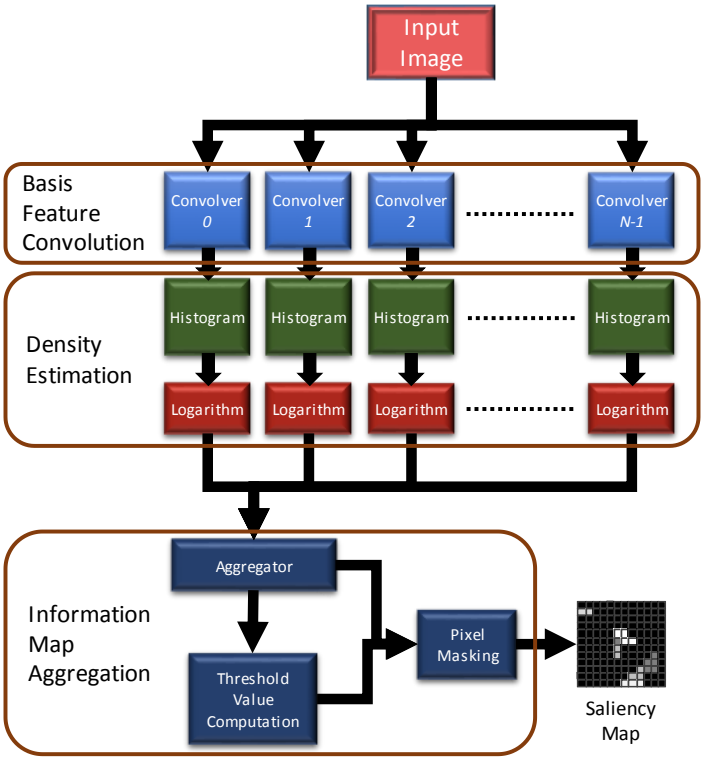


Figure 4-10. Algorithmic data flow through AIM algorithm, including structural breakdown

Table 4-3. AIM Accelerator Resource Utilization on Virtex6 FPGA

	Resources Required	**Resources Available**	**Percentage Utilization**
Logic Elements	43,099	297,600	6%
Flip-Flops	37,105	595,200	14%
36kb Block RAMs	139	1064	13%
Digital Signal Processors (DSPs)	446	2016	22%

The performance of the custom AIM accelerator was evaluated against a number of computing platforms. The results, shown in Figure 4-11, demonstrate that this accelerator easily outpaces the performance of even other hardware-based accelerators implemented without algorithm-specific customization. The fully customized AIM accelerator demonstrates a 6X to 10X speedup when compared to non-custom microaccelerator architectures at the same

frequency, and an 8X to 19X improvement over a high-performance multicore Xeon processor. When compared to the low-power ARM core commonly found on embedded platforms, the performance improvement is even more dramatic—330X to 630X.

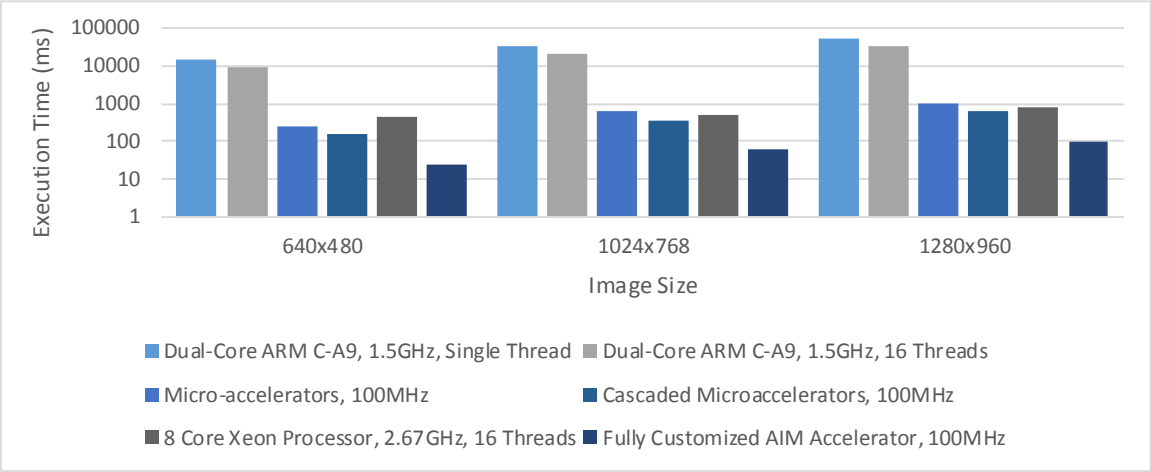


Figure 4-11. Acceleration of fully custom AIM accelerator vs other platforms

In further development of this accelerator, it was retargeted for acceleration on a Virtex7 VX690T FPGA. The FPGA was connected via PCI Express in a host system equipped with **a** 3.2GHz Intel Xeon CPU running Ubuntu Linux. This accelerator was evaluated in comparison with an optimized software implementation on the same system. In addition to the performance metrics, an evaluation of the power consumption of this system was performed as well. For power measurement, only the active runtime power was considered. This was calculated by subtracting the power consumed during runtime from the power measured while the system was idle. The accelerated implementation was found to run 16.7X faster than the software, while consuming 31.5% less power. Salient regions identified by the AIM accelerator are shown in Figure 4-12.

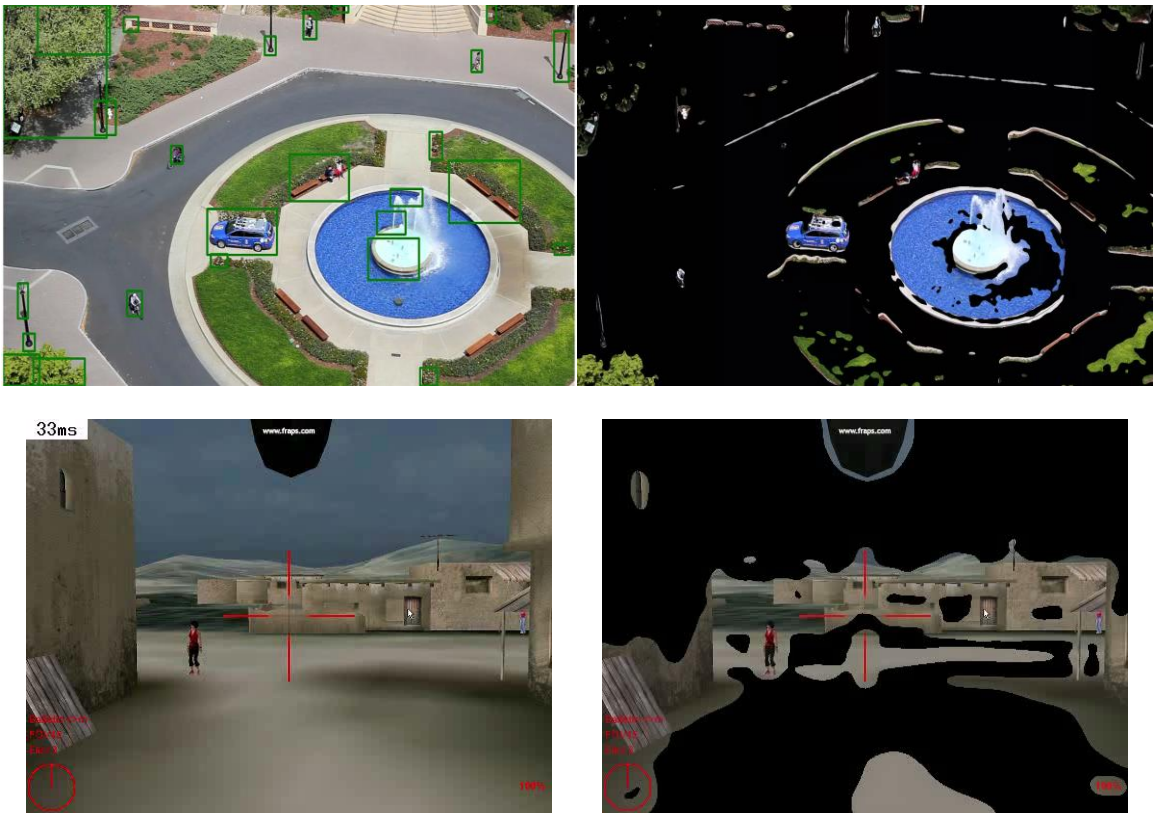


Figure 4-12. Examples of salient regions identified by AIM

The images on the left are the input frames, while the images on the right at the AIM-indicated salient regions. The top images are from the Stanford tower dataset [**42**], the bottom images come from data supplied by ONR for the SAF-T [**36**] program.

Chapter 5

Case Study: Hierarchically Refined Object Recognition

HMAX+ESVM Pipeline

Object recognition in real world systems must strive to meet two criteria: high performance and high accuracy. Given the enormous number of potential objects to be found in the world, this is no small task. Algorithms such as Exemplar SVM are capable of achieving remarkably high accuracy in the face of a large number of candidate classes due the nature of the computation performed. ESVM works to identify not only an object class, but the closest matching learned exemplar of that class. This accuracy comes at the cost of evaluating a series of classifiers whose number grows not necessarily with the number of classes, but with the number of exemplars per class. Figure 5-1 [**43**] show this growth in the detection time of the ESVM as the number of exemplars per class increases.

By contrast, object recognition based on an algorithm such as HMAX is effectively independent of the number of classes, as the feature extraction dominates the runtime relative to the actual classification. Shown in Figure 5-2 [**43**], HMAX features perform very well for a limited number of classes. With less than 10 candidate classes, the accuracy of object recognition can easily reach 90% or higher with just HMAX features. However, the discriminative power of these features quickly diminishes as the number of candidate classes increases. Even stabilizing around 50% accuracy for a large number of classes leaves much to be desired.

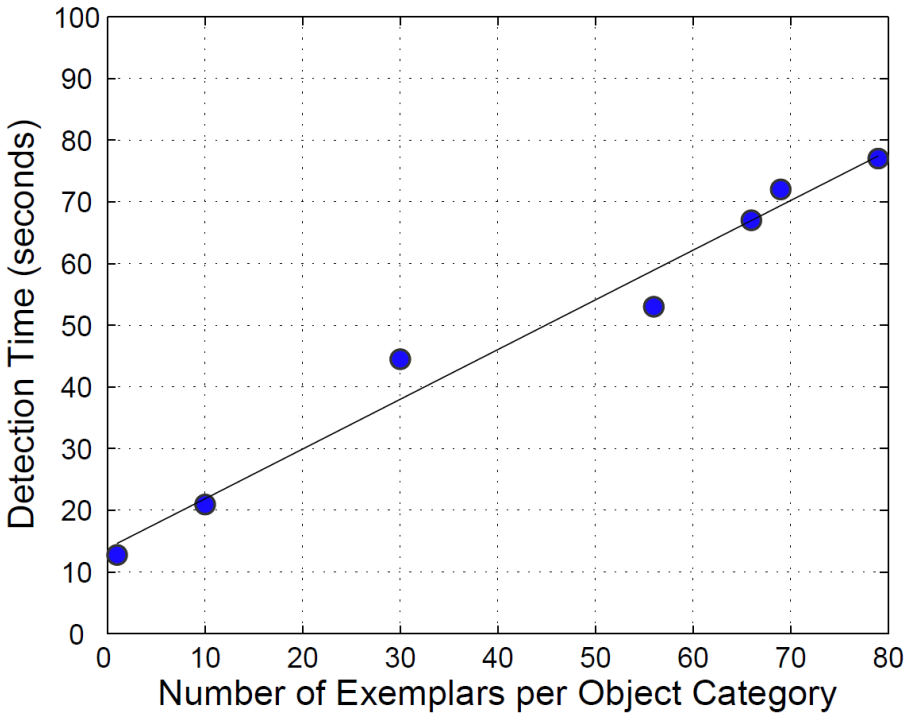


Figure 5-1. The runtime cost of Exemplar SVM scales with the number of exemplars per class

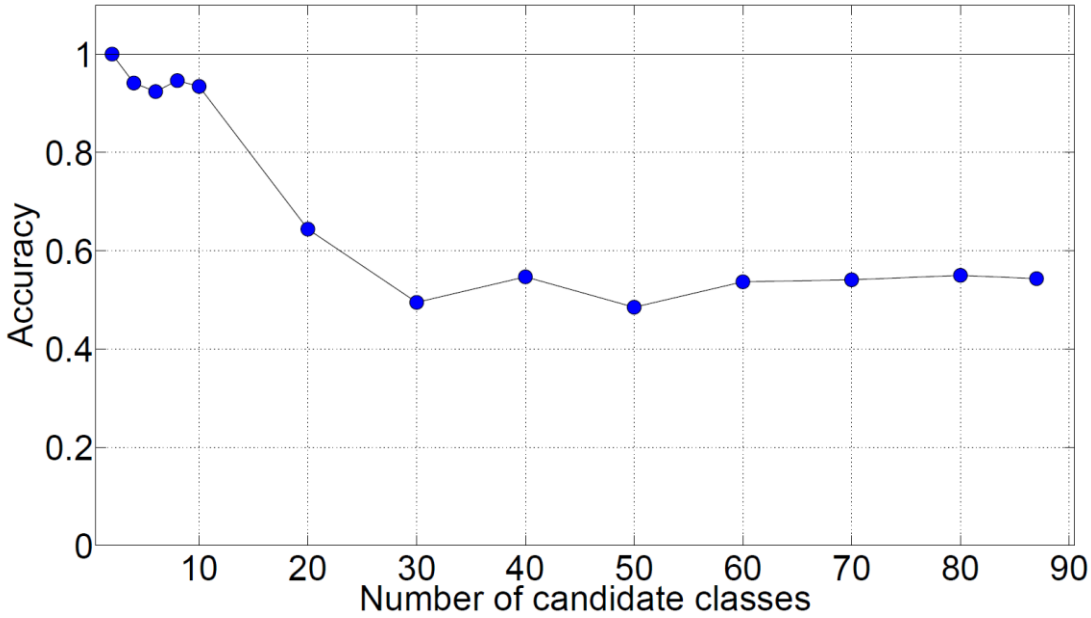


Figure 5-2. Difficulty of feature-based classification, such as HMAX, increases (to a point) with the number of candidate classes

The ideal system would combine the high accuracy and discriminative power of ESVM with the relative speed and performance of HMAX. Aiming towards just such a system, this chapter describes the results of doing exactly that—combining these algorithms in a hierarchical fashion to improve ESVM performance and accuracy by using HMAX as a filter to dramatically reduce the number of exemplars that must be evaluated.

The evaluation of this hierarchical recognition system was performed in the context of a grocery store environment. The training and testing datasets were independently generated by sampling from items that would be found in typical grocery store aisle. Eighty seven products were identified, characterized under eight general categories of packaging shapes. Using this list, the set of training images was synthesized by using the Microsoft Bing Search API to download a large number of images corresponding to each of these categories. The images for each category were manually pruned to ensure that no cross-product contamination was present in the dataset. This pruning also guaranteed that all images in the training set accurately represented the type or instance of the product that would be seen on grocery store shelves. Examples of images pruned from this dataset would be images that contained two brands of soda in the same image or images that contained a brand logo without any packaging. After the required pruning was complete, the training dataset consisted of just under 7800 images spanning the 87 categories. These images were used to independently train an HMAX-based classifier and Exemplar SVM system.

In the context of personal assistance in a grocery store environment, the object regions of interest may be detected by a number of algorithms such as saliency [**6**,**29**,**44**], objectness [**45**], or symmetry [**46**]. Figure 5-3 shows an example of an intelligent visual pipeline configured for such a recognition system.

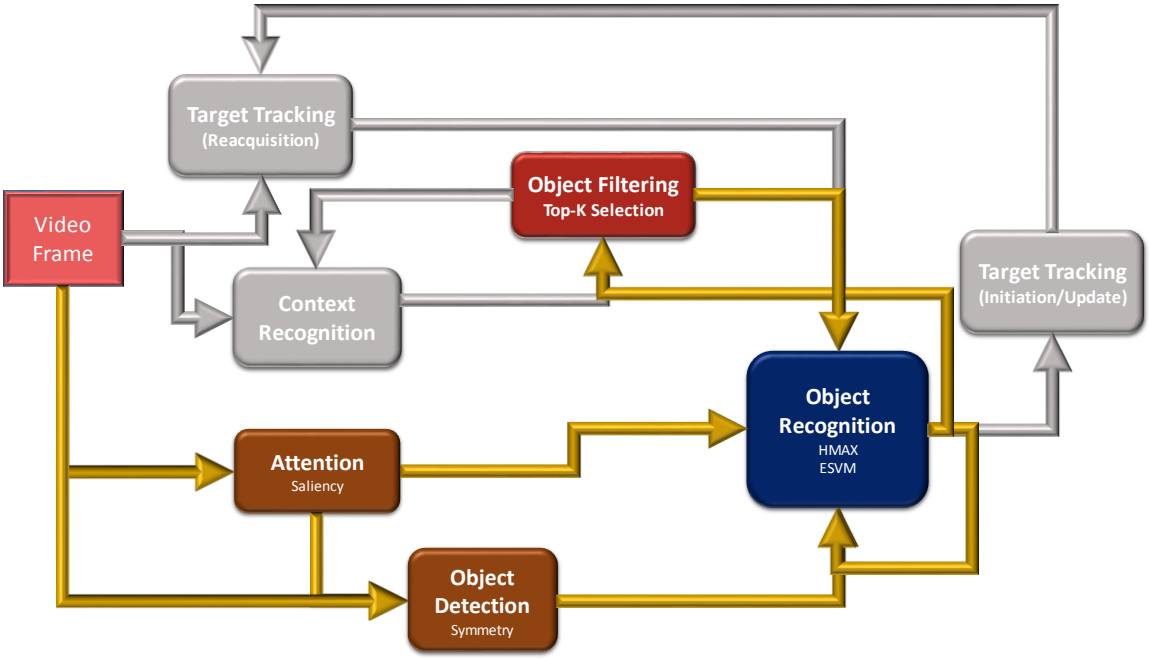


Figure 5-3. Intelligent visual pipeline configured for HMAX+ESVM classification

Once an ROI is selected, it is first dispatched to the HMAX classifier for processing. In this hierarchical system, the results of the HMAX classification, rather than being taken as an absolute winner-take-all classification, are passed on to an object filtering function. This prunes the number of candidate classes to a limited subset of the HMAX results. The classes included in this filtered set of classes are the most likely classes as returned by HMAX, taken as the *K* classes which have the highest classifier scores, in decreasing order of score (and estimated probability). By narrowing the set of candidate classes down to a few, the number of exemplars that must be processed through the ESVM is not fixed and greatly reduced by the number of classes returned by the filter.

Evaluation

In the evaluation of the merits of the hierarchical classification, a dataset of test images was created by manual annotation of a variety of images taken in the aisles of a local grocery store. All instances of each of the 87 training categories that appeared in these images were annotated using the LabelMe annotation tool [**47**]. After annotation, approximately 800 images were extracted spanning many of the training categories of products.

Each of the test images was classified using features extracted from HMAX. The per-class recognition accuracy for this evaluation of HMAX alone is shown in Figure 5-4 [**43**]. In this case, the average accuracy of HMAX across the test images was found to be 47.36%. A similar evaluation was carried out for the Exemplar SVM classification. These results, shown in Figure 5-5, demonstrate accuracy superior to that of HMAX, at 55.29%. However, with nearly 7800 training exemplars to be evaluated, the runtime cost of this system is on the order of hours per classification.

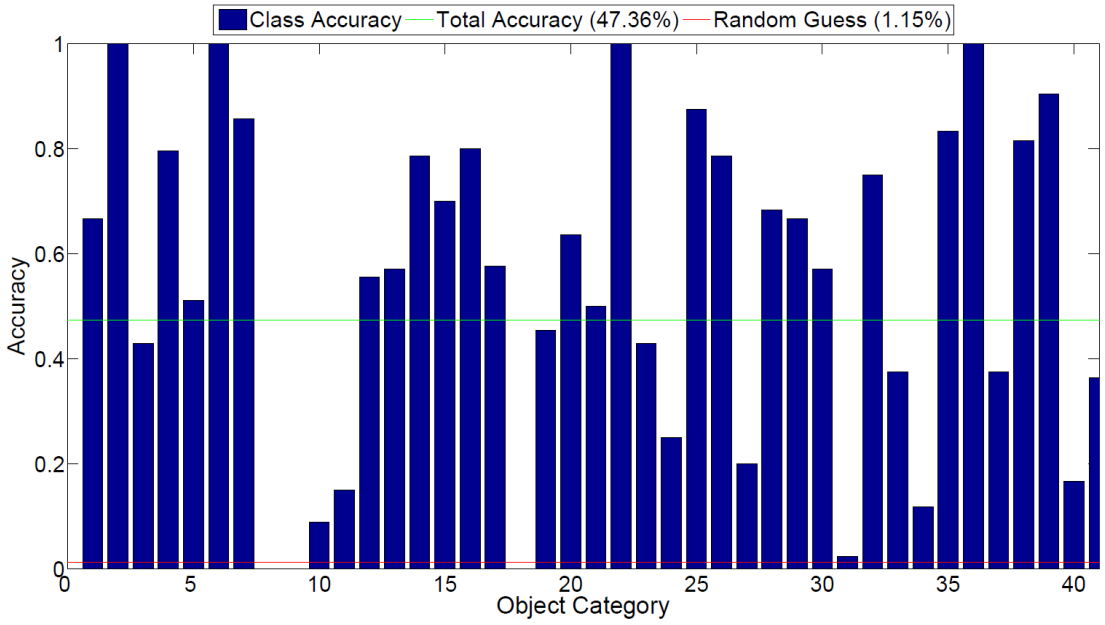


Figure 5-4. Per-class recognition accuracy for HMAX on grocery dataset

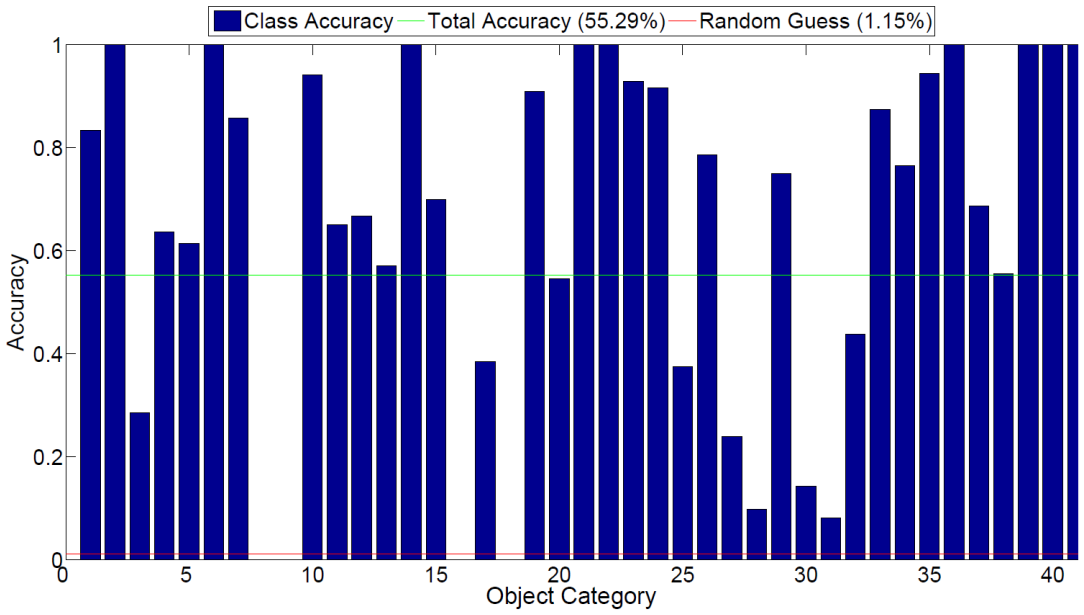


Figure 5-5. Per-class recognition accuracy for ESVM on grocery dataset

Selection of the parameter, *K*, indicating the number of HMAX classes to be returned is an important consideration. By selecting a value of *K* too small, the reduced accuracy of HMAX will filter out the correct exemplars reducing classification accuracy. Extremely large values of *K* would retain too many exemplars, negating much of the impact of the filtering on runtime performance. Figure 5-6 [**43**] shows a plot of accuracy versus *K* in HMAX classification. For the purposes of this plot, an object is considered to be classified as correct if the true class is contained within the top *K* HMAX result classes.

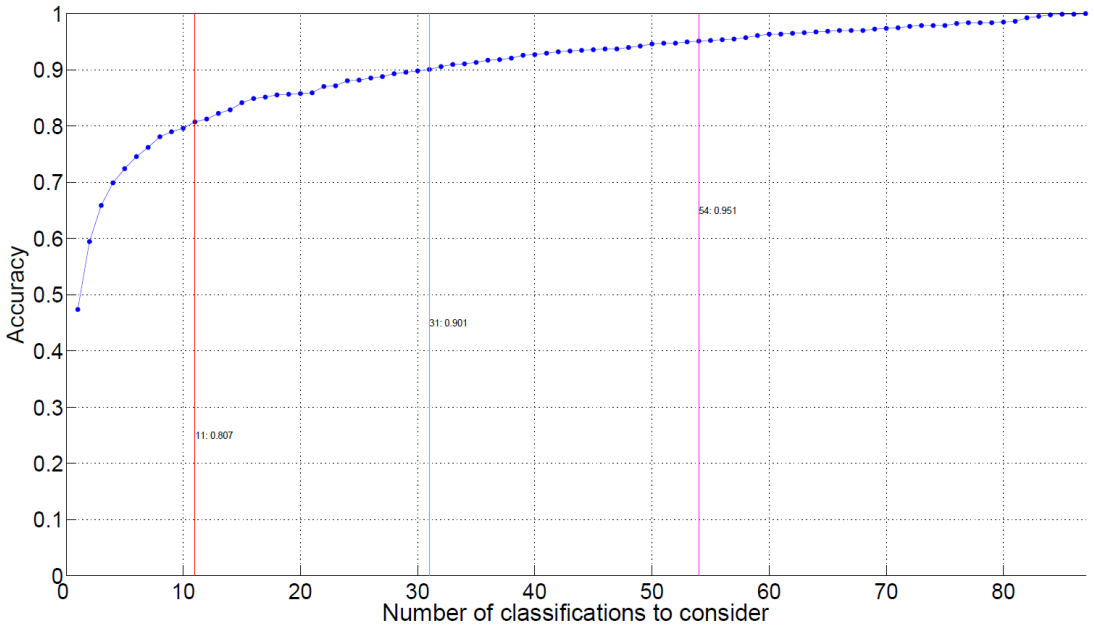


Figure 5-6. Accuracy of HMAX recognition, as the number of candidate classifications for consideration is increased

For the evaluation given in Figure 5-7 [**43**], the value of *K* was selected to be 11. This corresponds to the minimum value of *K* that was found to produce an estimated accuracy of 80% or greater in Figure 5-6 [**43**]. As this value of K corresponds directly to the number of ESVM classes that need to be evaluated, this reduces the computational load of the ESVM by a factor of 11/87 = 7.9X (12% of the whole) on average, though this reduction will fluctuate with the number of exemplars actually contained within the target 11 classes. The accuracy of the combined HMAX+ESVM system was found to be nearly 59.1%, corresponding to 14% and 4% improvements over HMAX and ESVM alone. While the improved accuracy of the ESVM appears to be marginal, it should be noted this increase comes with the added benefit of drastically reduced computational load. Further improvements to the accuracy of the ESVM portion of the system may be had by increasing the number of exemplars used in each class for

classification. Due to the pre-filtering provided by the HMAX front-end, this increase in the exemplar count would pose a modest increase to the overall system execution time.

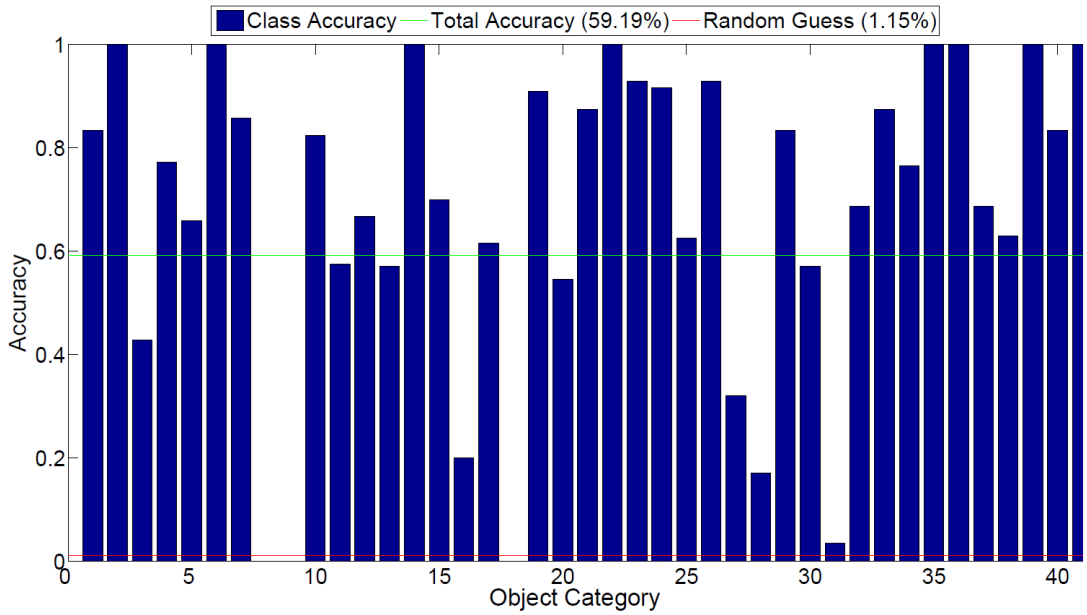


Figure 5-7. Per-class recognition accuracy for HMAX+ESVM on grocery dataset

Chapter 6

Case Study: Object Detection and Tracking for Personal Assistance

As wearable technology and computing devices equipped with imaging capabilities have gained prominence, the domain of visual systems for active personal assistance has seen a significant uptick in exploration. As an example, in the domain of assistance for visually impaired persons, such a system may enable the user to perform tasks many individuals take for granted, such as shopping in a grocery store.

With the ultimate goal of guiding a visually impaired person to reach out and grab a desired item off a grocery shelf, this chapter identifies and explores several critical aspects that must be addressed in order to achieve a successful interaction for the user. First and foremost, the system must be able to detect objects on the shelf. If the desired object is known ahead of time, as in the case of locating an item on a shopping list, several algorithms are available to accomplish this, such as SURF [**20**], symmetry [**46**], or template matching. In the more abstract case, where the object may not be known ahead of time, a more general approach such as attention or top-down saliency [**30**] may be applicable.

In some cases, the detection algorithms may be able to immediately identify not only a location, but a specific object as well. However, in cluttered environments, a lack of confidence may require additional processing capability to ensure the correct object is selected. Using the object locations identified by the detection algorithm, additional recognition processing may be performed. An algorithm such as HMAX, accelerated in hardware can provide high-speed and high-quality recognition in these types of situations.

Once a target is firmly established, the tracking algorithm must take over and guide the user towards the object. By employing a hand-mounted camera and glove, the tracking algorithm, in concert with additional analysis, can be used to estimate the camera movements necessary in

order to bring the desired object in line with the center of the view frame—the view as seen from the user's camera.

Finally, with a firm target being tracked, it is necessary that the system be capable of providing adequate feedback to the user in order to enable them to successfully locate and reach the object in the real world. This interface, targeting visually impaired persons, must naturally be non-visual. The obvious choices therefore are either audio cues, directing the user move their hand (and camera) in a specific direction, and haptic feedback, gently guiding the user towards the target location. An effective system may be constructed by combining the two cues—constant haptic feedback with a periodic audio cue. This chapter details the realization of such a system.

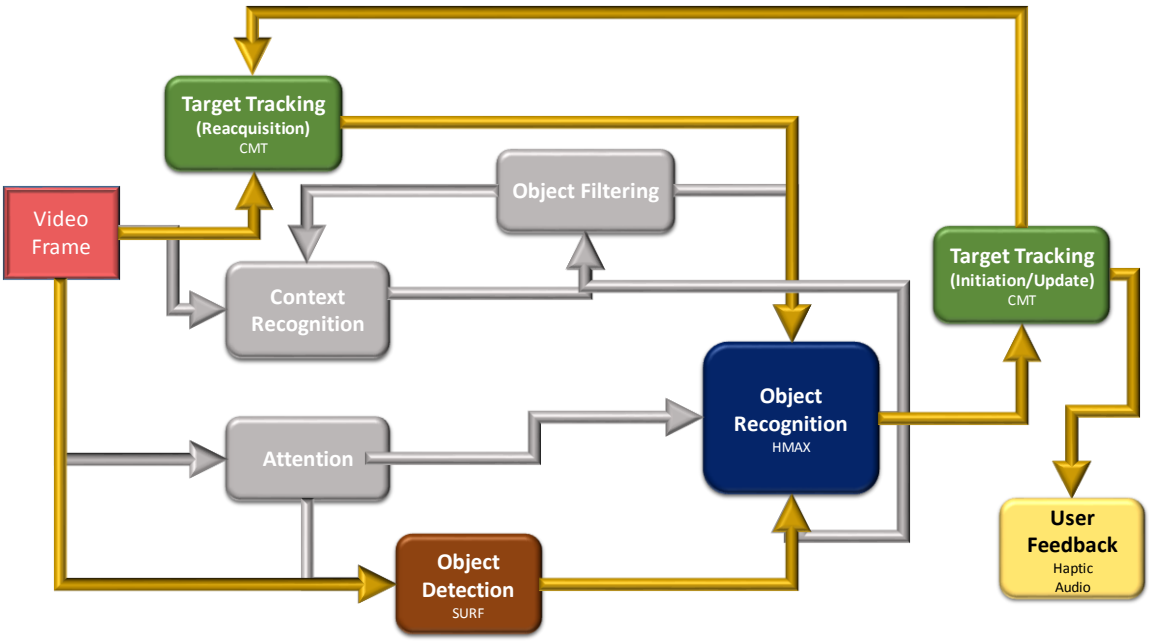


Figure 6-1. Example of Intelligent Visual Pipeline configured for Personal Assist
Modules in gray are not necessary for this function.

Detection

For generic scenarios, visual saliency algorithms such as AIM, are useful for locating regions and objects of interest within a visual scene. However, in the context of shopper assist, a more targeted approach is possible. Users working to fulfill a shopping list know ahead of time the products that are desired. With foreknowledge of the desired objects, a more specific algorithm geared towards a combination of both detection and recognition may be useful.

When the shopping list is generated, a dictionary of SURF models is generated with it. In the deployment scenario, in the store aisle, these models are used to detect candidate regions where the desired objects may be found. This detection not only enables detection of regions of interest, but provides a degree of object classification as well, since the product associated with matched model is already known. However, despite a detection, if the degree of match is not sufficiently strong, a secondary classification scheme may be used to boost confidence in the detection. These ROIs are sent to the recognition module for evaluation and boosting, if necessary.

Boosted Recognition

The first task of this module is to evaluate the incoming recognition, in order to determine whether additional verification is required. In the event that the initial match is sufficiently confident, no further processing is needed and the ROI is immediately forwarded to the tracking module. However, in the event that a detected keypoint match is not sufficiently confident, the ROI is processed through HMAX for additional feature extraction and classification to provide an additional degree of confidence or rejection of the initial classification. If the initial classification is confirmed, the target location is forwarded to the

tracking module. If the classification fails to verify the initial classification, the ROI is discarded without being sent to the tracker.

Tracking

Once an ROI is sent to the tracking module, it is registered with the provided classification tag and tracking commences based on the current view of the ROI in the visual scene. After initialization, information about the current location of the ROI in the scene is forwarded to the user feedback module for further analysis. As the user moves the hand-mounted camera through the visual scene, the tracking module continually updates the relative location of the registered ROI, passing this information to the user feedback module until such time as the tracked object is lost or removed.

Feedback

The user feedback module is responsible for providing cues to the user, directing them how to move their hand in order to reach the item on the shelf. This is determined by computing the distance and direction vector between the center of the field of view (where the user's hand-mounted camera is pointing) and the center of the target object (the relative location of the target). The critical component of this computation is the angle, which is computed by finding the arctangent of the relative pixel displacement along the x-axis and y-axis. Using this angle, the direction of camera movement necessary to place the target object in the center of the frame is determined. Regardless of the magnitude of the angle, if the displacement is sufficiently small, the camera is aimed towards the target well enough that the user is directed to move forward

This direction comes through two non-visual avenues of communication. The most persistent is haptic feedback, enabled through vibration motors attached to the glove. Multiple motors, scattered across the glove are activated separately, with each indicating a particular direction—up, down, left, right, or forward—that the user should move their hand in order to reach the target.

Additional feedback is given through occasional audio cues, reaffirming the persistent response provided by the haptic vibrations. In order to mitigate the impact of audio cues, playing too late or too often, audio feedback is only generated after sufficiently large displacements are detected. This also mitigates the effect of "shaky cam syndrome", where in the camera oscillates around the target rapidly due to shaky hand movements.

Finally, once the scale of the object has increased to the point that is no longer visible, the user is directed to move forward slowly, signified by both an audio prompt and decreased frequency in the haptic vibrations. This enables the user to cautiously locate the item and pick it up, without wildly thrusting their hand into the object, possibly knocking it or other items off the shelf. Once picked up, the user can notify the system, removing the object from the tracking system, and allowing the next item on the list to be detected, recognized and tracked.

Chapter 7

Future Implications of Emerging Technologies and Non-Visual Features

This chapter provides an exploration of emerging technologies and architectures that may serve to further enhance the development of intelligent visual systems. This exploration includes the evaluation of emerging device technologies which stand ready to replace conventional MOSFET architectures. Abandoning traditional architectures entirely, non-Boolean architectures based on coupled oscillators are evaluated for their potential to contribute to future vision systems. Finally, improvements to vision systems stemming from entirely non-visual modalities are identified and analyzed to identify their impact when incorporated into visual recognition systems.

7.1 Emerging Technologies

Modern digital systems are largely composed of circuitry which is architected using planar silicon MOSFET transistors. Through advancements in fabrication technology, these devices have scaled down over the years from 350nm and 250nm devices of the mid-1990s to the 32nm and 22nm devices of the current generation. Along with this scaling, came increased device density, reduced chip areas, lower supply voltages, and improved performance. However, in addition to these benefits, additional hurdles have been introduced as these small nano-scale devices have hit what has become known as the "power wall."

At larger feature sizes, dynamic power is the dominant component of power, while the large channel length effectively minimizes the static leakage power consumed by each device. With scaling, any increases in dynamic power due to frequency were offset by reductions in operating voltage, made possible by the smaller device size. However, as the channel length is

shortened, significant increases in static leakage power emerge--again some of which is offset by supply voltage scaling.

In order to combat the leakage current introduced by small feature devices, FinFETs have emerged as a leading contender to continue the desirable scaling trends previously enjoyed by planar MOSFET devices. These FinFETs employ a three-dimensional structure in which the gate wraps around three sides of the channel, allowing much tighter control over current flow. This has the effect of reducing leakage current while increasing the switching speed of the device.

However, in order to maintain distinct "on" and "off" states, any reduction in supply voltage must be met with a reduction of the switching threshold voltage of the device. Due to this fundamental characteristic of MOSFETs, supply scaling has become infeasible due to stagnated threshold voltage scaling, as seen in Figure 7-1. The limitation on scaling of the threshold voltage is a limitation imposed to the carrier transport mechanism employed in MOSFET-based devices. As such, even the three-dimensional FinFET is subject to the limitations on scaling of the threshold voltage. In order to circumvent this limitation, a new transport mechanism is necessary.

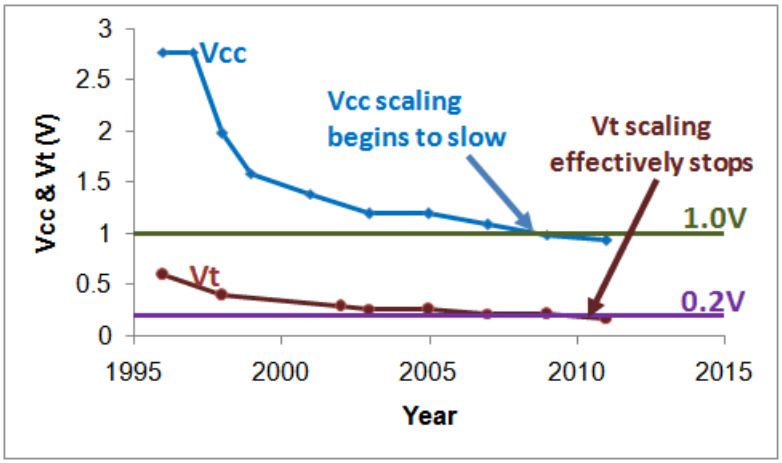


Figure 7-1. Limitations of Scaling of Vcc and Vt in Silicon MOSFETs

Tunnel FETs

Traditional MOSFETs and FinFETs use silicon as the semiconductor material, with the physics and behavior of the channel adjusted through the use of dopants to control electrical carrier concentrations in the drain, channel, and source regions of the device. Carrier transport occurs due to emission over the energy barrier in the channel after it has been sufficiently moved, as determined by the threshold voltage, due to an applied gate voltage. Silicon MOSFETs also suffer from a shallow sub-threshold slope of at least 60mV/decade. This means that each 10X increase in drain current requires at least 60mV increase in gate voltage. Such MOSFETs operating at low supply voltages suffer from an additional limitation on drain current, potentially limiting the device's potential to reach saturation.

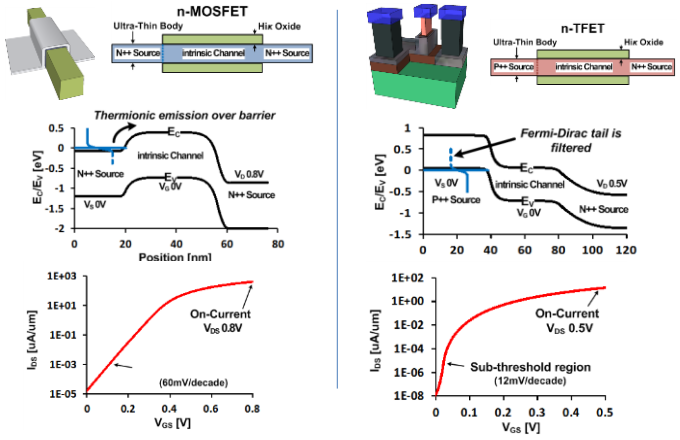


Figure 7-2. Transport and Drive Characteristics of n-MOSFET vs. n-TFET

Tunnel FET devices, rather than rely on thermionic emission of carriers for transport, derive their name from *tunneling*--the primary transport mechanism in such devices. Using different materials, along with a *gated p-i-n* structure, an abrupt energy barrier is created to effectively suppress current flow. However, the band structure is such that a small change in the intrinsic region opens a "thin" barrier between energy bands allowing carrier transport via

interband tunneling. This small change that is necessary to switch the device "on" translates to a small threshold voltage, often typically 100mV or less. Additionally, the band structure of the TFET leads to a very small "steep-slope" over 12mV, which further enables operation at a much lower voltage.

Due to the structure of the energy bands in a TFET device, reverse conduction is virtually impossible within normal operating parameters. However, a significant forward bias will cause an avalanche current resulting in a significant reverse conduction, entirely independent of gate voltage, as seen in Figure 7-3 [**48**]. Due to this behavior, it is clear that TFET devices cannot be a direct replacement for MOSFET devices in all circuits. While neither static nor dynamic logic circuit designs would be functionally impacted by a direct replacement with TFETs, any design that includes pass transistors must consider the asymmetric current conduction characteristics.

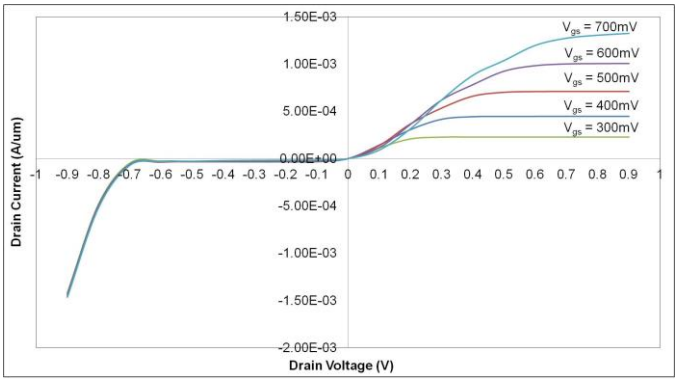


Figure 7-3. Unidirectional conduction of n-TFET (Id-Vds)

As a single device, TFETs appear as very attractive alternatives to silicon FinFETs, already demonstrated to be superior to MOSFETs, particularly at low voltages. However, a deeper analysis is necessary, considering the impact of these design considerations on the overall performance. Energy-Delay characteristics have been evaluated for a variety of simple circuits. These components were then evaluated together, within the context of a 32-bit sparse Manchester

Carry Chain adder. The Energy-Delay characteristics of this TFET-based adder can be seen in Figure 7-4 [**48**,**49**]

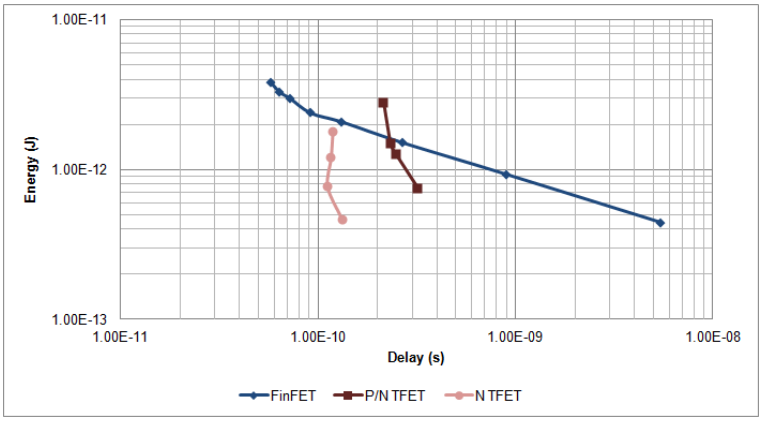


Figure 7-4. Energy-Delay Plot of TFET-based 32-bit Sparse MCC Adder

Storage elements such as flip-flops and latches are widely used through modern computer architectures and constitute a significant aspect of those designs. Therefore, it is prudent to consider those circuit designs when evaluating Tunnel FETs as replacements for FinFET-based architectures. Many were found to be functionally unaffected by direct device replacement. Flip-flops based on differential sense amplifiers were improved with the addition of a single NTFET to restore a lost discharge path. However, the most interesting case was found to be the "D Flip-flop" (DFF), composed using two D-latches with staggered clocks.

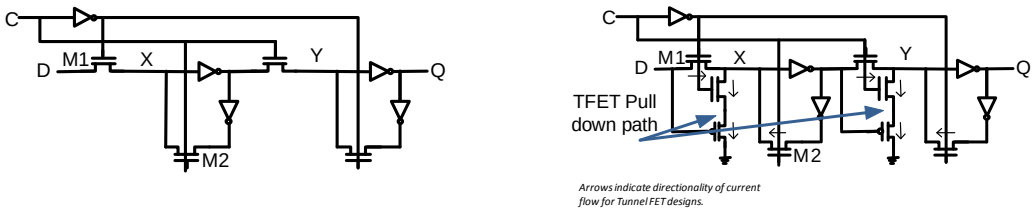


Figure 7-5. D-Latch Based Flip-Flop Design Using FinFETs (left) and TFETs (right)

This TFET-based DFF was found to demonstrate superior performance in virtually all flip-flop characteristics. However, the TFET design consumed more dynamic power than its FinFET counterpart during data transitions. Further analysis of the circuit found this to be attributable to two TFET characteristics: the low threshold voltage drop, and the enhanced Miller Effect (Figure 7-6).

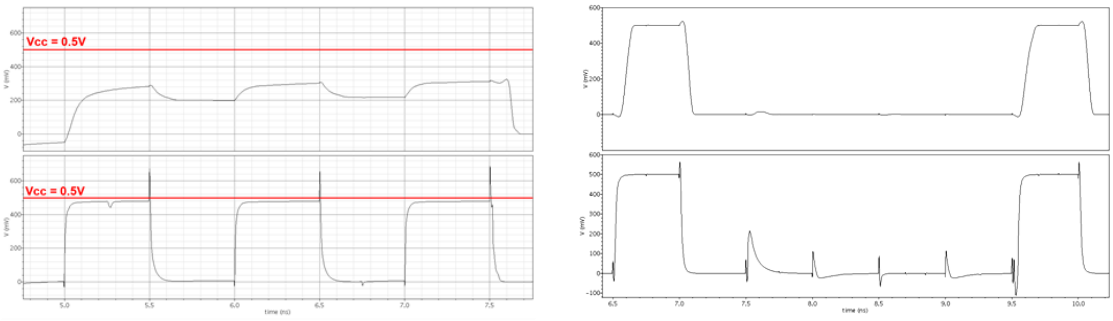


Figure 7-6. Threshold voltage drop (left) and switching overshoot (right) for FinFETs (top) and Tunnel FETs (bottom)

At or near sub-threshold supply voltages, the reliability of CMOS devices becomes highly questionable, with the most vulnerable circuits being SRAM memories.

The resilience of TFETs to soft errors were evaluated and compared to the resilience of FinFETs. This evaluation was performed using circuit-level simulations including simulation of current-injection caused by particle strikes of varying strength. The TFET SRAM design is found to require particle strikes which generate 4-5X more charge at the impact node in order to generate a bit-flip. This increased resilience has been traced to the enhanced Miller Effect found in TFETs [**50**], as seen in Figure 7-7. When a particle strike attempts to induce a bit flip, the Miller Effect from the opposing inverter provides a counteracting charge, effectively assisting in the recovery of the node, preventing the bit flip from taking hold.

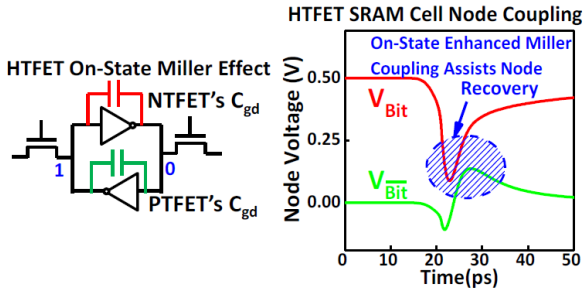


Figure 7-7. Effect of Enhanced Miller Capacitance on TFET SRAM Soft Errors

Electrical masking effects also inhibit the infiltration of soft errors into a system. These effects were evaluated by determining the number of stages through which a strike-generated pulse is able to propagate before dissipating due to electrical damping. Once again, TFETs were found to exhibit far superior electrical masking effects when compared to Si FinFETs at the same supply voltage.

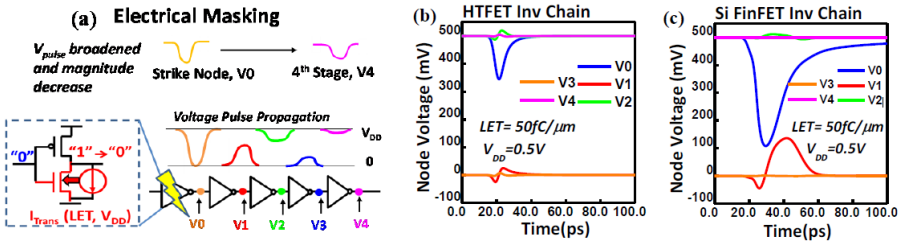


Figure 7-8. Effect of electrical masking on soft error propagation

Non-Boolean Architectures

Exploration of emerging device technologies as a replacement for MOSFETs has also triggered a renewed interest in arrays of coupled oscillators. These devices are well known, and have been well studied for a long time. However, advances in technology have been made to the point where devices of this nature may be useful in performing computations in computer

systems. The essential concept of the coupled oscillator array is that despite a variety of initial states and conditions on each oscillator, feedback and coupling effects between the oscillators will cause all oscillators in the array to synchronize. The exact nature of the synchronization is a function of the physical device type, the distribution of the initial inputs, as well as the coupling topology.

Due to the variety of materials, types, and operational ranges of coupled oscillator technologies [**51**,**52**,**53**], synchronization may take many forms such as frequency locking or phase locking, either over time or as a binary state--locked or unlocked after a specified period.

The details of the I/O characteristics of the oscillators vary with the technology, but the general commonality is that inputs, be they voltages or currents, set up the initial conditions for each oscillator in the array. After a period of time, due to the various coupling effects, the output of the oscillators will be in some state of synchronization. The circuitry to detect this synchronization is still an active area of study. A method for detecting frequency locking has been proposed in [**54**], making use of an integrate-and-fire mechanism in order to measure the time with which the oscillator array converges to a common frequency. Those vectors which are the most similar are most closely synchronized initially, and therefore require the least amount of time to fully synchronize.

Coupled oscillators also provide an interesting platform for the composition and computation of vision algorithms. The essential operation performed by coupled oscillator arrays is that of a degree-of-similarity estimation between two vectors. The degree of similarity and comparison may be adjusted by modulating the coupling effects between specific subsets of oscillators in the array. The work presented here represents an early study of the feasibility for image processing using coupled oscillators such as those expressed by the Kuramoto model [**55**,**56**,**57**,**58**,**59**], which assumes a uniform, weak, all-to-all coupling between oscillators in the array. The inputs to the oscillator array are derived from input image pixels. The output of the

array processing is taken as a measure of the time required for the oscillator array to synchronize to a common phase for the given inputs.

Edge Detection and Bounding Boxes:

Edge detection is a commonly used algorithm to outline objects in a scene and as a potential first stage in performing segmentation on images. Detection of these edges also can provide reasonable bounds for identifying a bounding box, or rectangle, surrounding any given object.

Edges, by their definition, represent pixels in an image that are starkly different from those around it, confined to a small area. The Canny Edge detection algorithm finds these pixels by computing the location of the maximal local gradients in the X- and Y- directions, as well as the combined XY directions. While oscillator arrays may be used to derive an approximation of the gradient functions in these directions, the overall computation would at a minimum require multiple passes over the image to compute the separate directional gradients, in addition to digital computations of the sum of the gradients and finding the local maxima. Despite the effort, the results of the gradients would only be approximations due to the nature of the oscillators.

Examining the intent of the algorithm--detecting the edge pixels--allows for another interpretation which is much more oscillator-friendly. Edge pixels are those which are different from their neighbors; the pixels which are most dissimilar within a small neighborhood. In effect, this is what the oscillators provide--requiring the longest synchronization times for those vectors which are the most dissimilar.

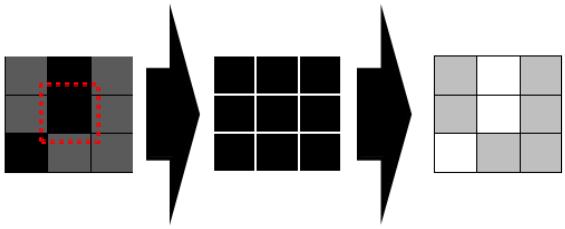


Figure 7-9. Oscillator Inputs Derived for Edge Detection

In order to find a measure of (dis)similarity of a pixel with its neighbors, the inputs to the oscillator array are generated as the difference between the central pixel (the test pixel) of a 3x3 image window and all 8 of its neighbors, as shown in Figure 7-9. The magnitude of the output pixel is then taken as the ratio of two values. The first value, numerator of the ratio, is the time required for the array to converge for the given inputs. The denominator is the time required for the array to converge given the maximally different inputs; the scenario in which the center pixel is maximally different from all 8 neighboring pixels (i.e. a single black pixel with all 8 neighboring pixels being white).

The computation of the pixel values for the entire image is then performed by sliding this window across the image, computing the differenced input and synchronization time for the array at each central pixel over the image. It is worth noting that each output pixel is independent of all other output pixels, and so for performance improvements, any number of these windows may be computed in parallel.

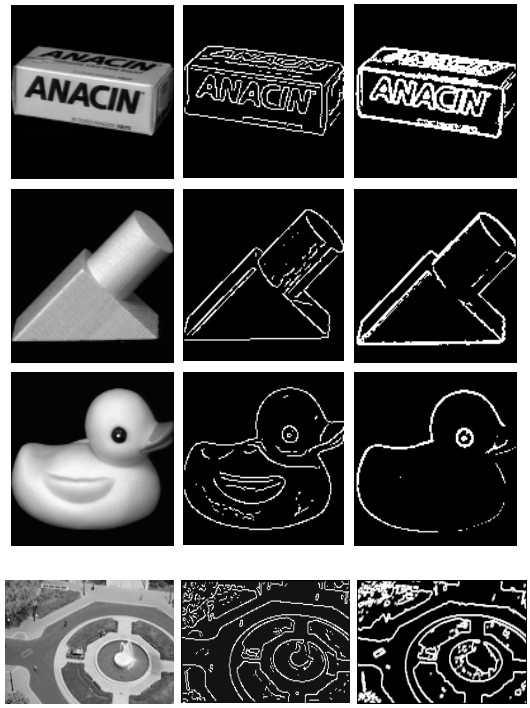


Figure 7-10. Edge Detection using Oscillators
(left) Original Image **(middle)** Canny Edges **(right)** Oscillator Edges

Saliency Approximation

Basic visual saliency algorithms work towards identifying those pixels which grab the viewer's attention; i.e. identifying those pixels which stand out. This operation is very similar to the proposed edge detection using oscillators, however, it must be performed over a larger area.

To that end, a larger 19x19 image window is used and processed using a similarly larger oscillator array. The inputs are generated in exactly the same manner as in the edge detection experiment, however, the differencing is performed over a larger area. A 7x7 example of input generation for saliency is shown in Figure 7-11. The size of the array in this experiment is arbitrary; a smaller or larger array may be used. However, as the size of the array increases, the resolution of detection decreases. This results in something that amounts to edge detection when

using small (3x3) arrays and begins to approach something resembling visual saliency as the array becomes larger.

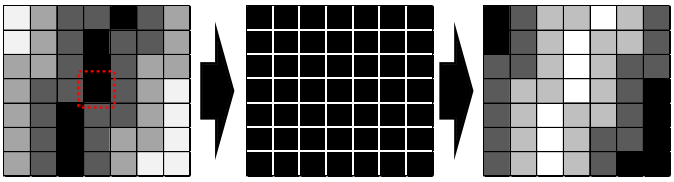


Figure 7-11. Oscillator Inputs Derived for Saliency Approximation

In this oscillator-based saliency, it is worth noting that in images/regions of weak contrast, the large oscillator array is not effective at detecting objects, as seen in the teddy bear image, shown in Figure 7-12. However, separating the array allows the oscillators to mark enough of the bear as salient to warrant detection. Another interesting example is the duck, in which a significant portion of the grassy area is marked as salient by the AIM-Saliency algorithm. The full oscillator array detects boundaries around the duck that are much more defined while still detecting enough inside the duck's boundary to be considered salient.

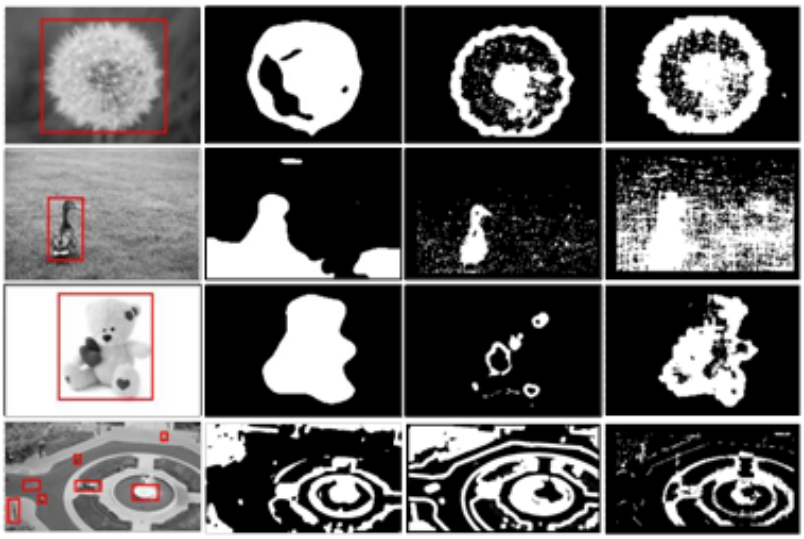


Figure 7-12. Results of Visual Saliency
From left to right: Original Image, AIM Saliency, Full Oscillator Array, Separated Oscillator Arrays

7.2 Multiple Modalities

In addition to device-level technologies, focused on improving the performance of recognition systems in terms of power and speed, there is room for improvement in terms of accuracy as well. It is known and understood that machine learning based classifiers suffer in terms of accuracy as the number of potential candidate classes increases. One method of combating this problem would be use cascades of classifiers ranging from broad to specific classifications, in effect narrowing the field of candidate classes at each stage. This approach is demonstrated through the use of hierarchical classification in Chapter 5. However, such a solution still solely relies on visual features. This not only ignores a large potentially helpful feature space, but is, by definition, incapable of capturing features which are not visible.

By fusing additional data modalities, feature classifiers can learn from a much larger feature space in order to more accurately identify scene context and better inform object recognition algorithms. The research described here focuses on the augmentation of vision-based scene and context recognition through the inclusion of features extracted from audio data.

Audiovisual Scene Recognition

While humans often rely on vision to make judgments about their surroundings, identify objects, and take actions, sight is not the only sense which enables this behavior. In fact, even those with impaired vision are capable of making many of the same identifications as those with perfect vision. The sounds of the environment often shape our understanding of what is happening and the identification of objects around us.

Here, the impact of audio on context awareness, or scene recognition, is evaluated. This evaluation uses video clips as the evaluation data. Spanning fifteen classes of sporting events, the

dataset is split equally into training videos and testing videos, with each video broken into visual frames and audio streams. Visual features are extracted using the visual GIST [**28**] algorithm presented by Oliva and Torralba. A linear classifier is trained using features from the training set and evaluated on features from the testing set.

Extraction of audio features proves to be a more challenging matter. There are a number of audio features and signatures known, and a variety of ways to learn them, ranging from simple frequency responses as in Fast Fourier Transforms (FFT), Mel-Frequency Cepstral Coefficients (MFCC), or One-Zero Gamma Tone (OZGT) filter responses. Many of these methods are independent of the length of the audio clip, however, in order to support a real time audio recognition scheme, some concept of an audio "frame" must be clearly defined. Classification is then performed at the frame-level, which need not necessarily be in time-sync with the corresponding video frames.

Many of these features were investigated in order to find their discriminative capability with respect to audio scene awareness, however, the sporadic nature of these responses over time often created issues. Attempts at classification using these raw features produced results amounting to noise, just above random chance at around 7%. In attempts to learn a more useful feature space, these raw features were processed through a sparse auto-encoding neural network (SAENN). Using this structure, a key subset of frequency-based features were learned and classification was once again performed. While slightly better, around 9%, the audio-only classification of this dataset was found to be ineffective, even with learned features.

These classification schemes appeared to exhibit extreme sensitivity to temporal effects as well as sharp discontinuities in frequency responses between audio frames, even within the same scene. Borrowing some aspects from the HMAX algorithm, which in the visual domain mitigates the effects of scale and spatial discrepancies, a conceptually similar feature extraction algorithm was developed for audio frames.

In order to mitigate the impact of temporal effects on the feature extraction, a series (or pyramid) of progressively smaller (temporal scales), but overlapping windows are created, modulated by a windowing function to reduce the impact of edging effects on the windows. This stage is analogous to the pyramid generation preprocessing performed in HMAX.

For each of these windows, raw features are extracted based on FFTs, MFCCs, OZGT, and similar frequency responses. These feature responses are then pooled both across temporal scales and across windows within a scale for each feature independently. These processes correspond to the S1 and C1 stages of the HMAX algorithm.

As done for the S2 stage of HMAX, the pooled responses are randomly sampled in order to generate a dictionary of prototype frequency signatures used for future feature extraction. The size and extent of this dictionary is variable. In evaluation, each extracted feature is correlated with each corresponding feature in the dictionary to produce a similarity value. These values are pooled across large bands of temporal scales, as in HMAX C2, to produce the final feature vector.

Using a linear classifier, trained using the features extracted from this aural HMAX, the test data was once again evaluated for scene recognition accuracy. While, the results are not yet on par with the recognition achieved by visual GIST, the results are promising. The improved accuracies demonstrate the effectiveness of mitigating temporal and frequency effects in audio feature extraction.

The combined audiovisual classification is performed by fusing the response of the visual scene classifier with that of the audio scene classifier. Each vector of responses is normalized to the range (0,1) through the use of a log-sigmoid function applied to each response value. A weighted averaging function is applied to the two vectors to produce a fused response, with the highest being taken as the most probable. The results of the visual-only, audio-only, and audiovisual scene recognitions are shown in Figure 7-13.

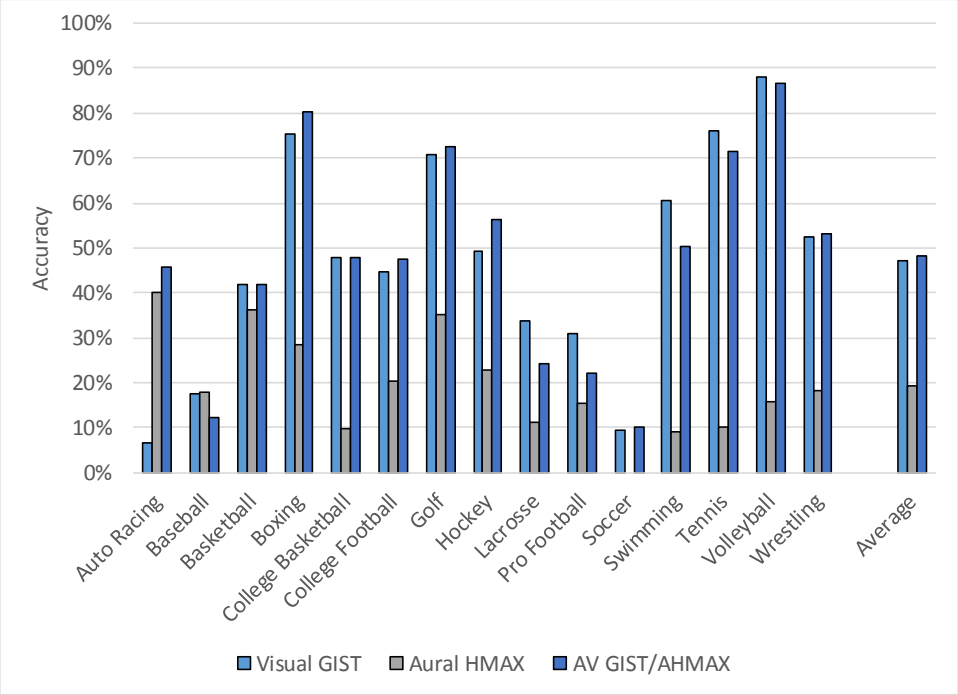


Figure 7-13. Evaluation of scene recognition using both visual and audio features

The average accuracy of scene recognition using only visual GIST features was found to be 46.99% for the fifteen sports classes. The extracted audio features, on their own, produce an accuracy of only 19.35%, which while relatively low, is a marked improvement over raw features and features learned through an SAENN. By combining the two feature sets, however, the accuracy is increased, although marginally (1.11%), to 48.09%. Further inspection of the accuracy of individual classes within the dataset shows an interesting case for the audio features. Several classes show reasonable improvements in accuracy. These results show that there is a case for the inclusion of audio features as some classes exhibit remarkably distinctive audio signatures, as shown in Figure 7-14.

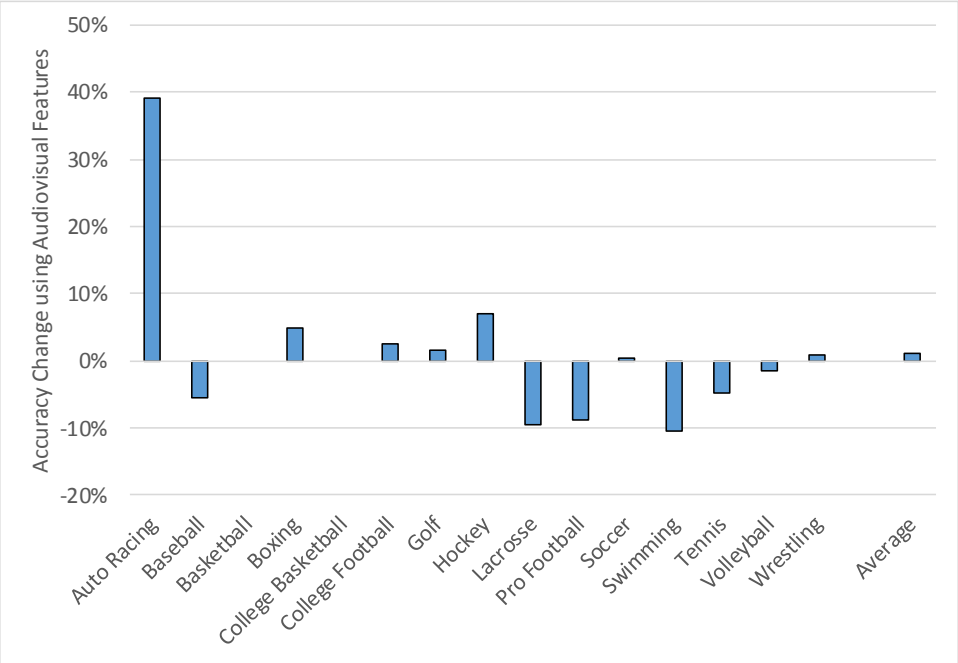


Figure 7-14. Change in scene recognition accuracy between visual and audiovisual features

Lexicovisual Scene Recognition

The previous section focused on the inclusion of general audio as a modality for scene recognition. However, not all audio signals are general, ambient, or environmental. The spoken word provides another key audio feature capable of further enabling scene and context discrimination. Using the same audio and video streams, a transcript of the words spoken in each scene were created. Based on this, a probabilistic training model was generated associating each word with a probability of a particular scene. This probability is simplistically generated by building a word frequency histogram, ignoring many of the most common words, such as “the”, “an”, “a”, “and”, “with”, etc. Classification of a scene after each spoken word is produced by indexing the histogram using the recognized word.

Simple word-by-word application of such a system would generate a very spiky behavior in the speech classification as new words are recognized. In order to combat this behavior, a Naive Bayesian model is employed. This lexical scene estimation model bases the current estimate of the scene context not only on the probability of the new word, but rather on a weighted average estimate factoring in the most recent previous scene estimates.

Visual scene probabilities are generated by applying logarithmic-sigmoid function to the output of the linear classifier. This normalizes the visual responses into a range compatible with the probabilities produced by the lexical scene recognition.

Similar to the audiovisual fusion, a weighted average is used to fuse the visual and lexical responses into a single response vector. The lexical responses are only updated as new words are recognized, and therefore the initial estimates assume a uniform probability distribution. This yields classification to the visual scene recognition until spoken words are detected.

A similar approach is applied in the fusion of visual and lexical cues, applying a weighted average of the two. The initial probabilities assume a uniform distribution among all available classes, effectively yielding full control to the visual classifier, until speech is heard. In the event that a word is recognized, but is not known from the training set, the lexical estimate of the scene is not updated.

This Bayesian system is based on two key weights. The first is that of the lexical prior estimate. This weight determines how much impact prior estimates of the lexical scene estimate have on current predictions. After exploration of this weight, shown in Figure 7-15, it was found to be most effective when weighting heavily on the prior estimates, effectively smoothing out spiky behavior of new information, while not ignoring it entirely.

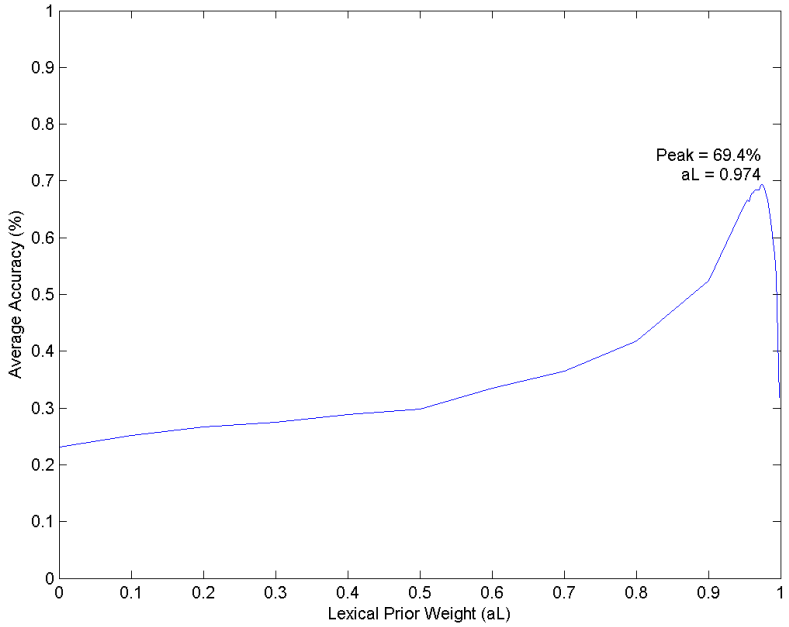


Figure 7-15. Exploration of impact of lexical prior weight on accuracy

The lexicovisual weight influences how much influence to give to the lexical estimate versus the visual estimate in determining the final scene classification. Shown in Figure 7-16, the speech content of a scene can provide a great deal of information about the context of a scene. Using this limited dataset, speech alone provides better accuracy than visual cues, with isolated key words such as “strike”, “par”, or “lap” rapidly skewing the lexical predictions to classes such as “Baseball”, “Golf” or “Auto Racing”, respectively.

Fusion of speech and visual data results in improved accuracy over vision alone, but lower with respect to speech alone. A larger database can be expected to moderate this disparity somewhat, but still the value of speech in the domain of scene and context recognition cannot be ignored.

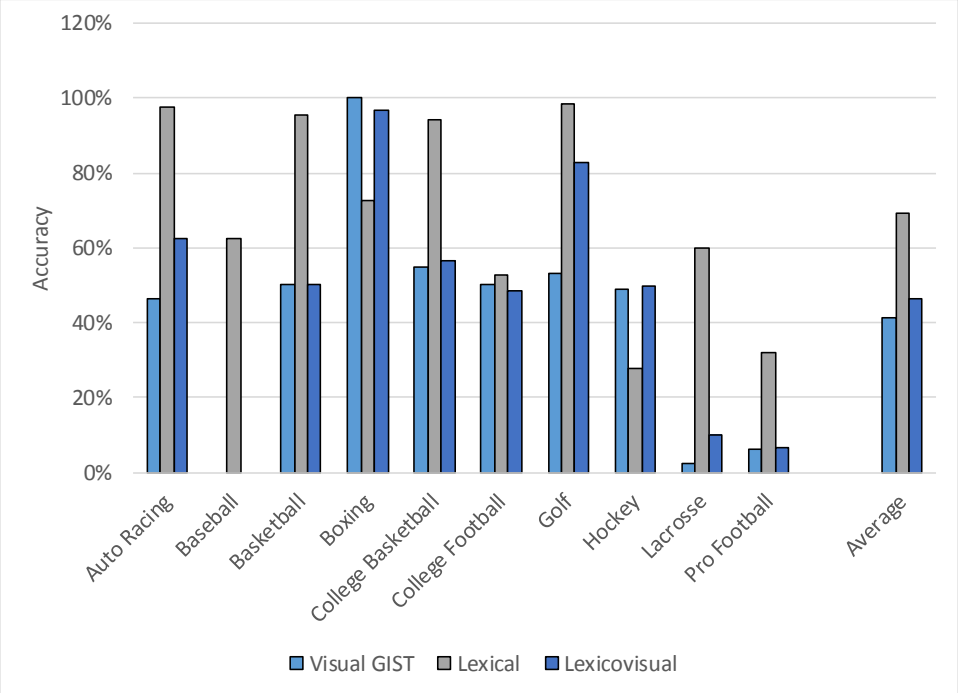


Figure 7-16. Evaluation of scene recognition using both visual and audio features

Chapter 8

Summary

Intelligent visual systems require a number of components in order to realize the goal of developing a system that is capable of seeing and interacting with the world in a manner similar to that of humans. An enormous amount of research has been devoted to the development of these components, however, each of these components must be integrated into a common system in order to be maximally effective.

This work presents a unified framework for the realization of such a system. This framework contributes a solid, yet flexible foundation for the development and evaluation complex visual systems. This pipelined framework defines a non-rigid communication flow, allowing a variety of components to communicate easily and as-needed in order to enable intelligent understanding and recognition the visual environment. This flexibility not only supports the inclusion of vision-based modules, but modules which derive information from additional data modalities, such as audio, for improved performance and recognition. The underlying implementation of these modules is irrelevant, allowing for multiple variants of each algorithm, including those accelerated through the use of high-performance customized hardware modules.

This dissertation also details the contributions of the Cerebrum tool framework. This framework provides the facility for non-hardware engineers to quickly and efficiently compose hardware accelerators on reconfigurable FPGA fabrics. This tool enables composition of such accelerators requiring detailed knowledge of the underlying hardware on the part of the user.

This framework is used in order to compose and synthesis a highly efficient hardware accelerator for the HMAX algorithm. This accelerator, the detailed exploration and development of which is included as part of this dissertation, was selected as the foundation for evaluation of

future neuromorphic accelerators as part of the DARPA Neovision2 project. IN addition to the HMAX object recognition accelerator, this work also contributes a treatment of a similar accelerator for the AIM salient region detection algorithm. The accelerator developed for this algorithm was selected for inclusion as part of the Office of Naval Research's (ONR) Supervised Autonomous Fires Technology (SAF-T) program.

This work has further demonstrated the utility and real-world effectiveness of these accelerators and the intelligent systems which they enable. Hierarchical classification of grocery products through the use of multiple object recognition algorithms shows an improved accuracy by leveraging the benefits (speed and accuracy) of independent recognition algorithms.

A more complex system, targeting personal retail assistance, demonstrates the social benefits such a system can provide. The system demonstrated is shown to be capable of empowering visually impaired persons in the task of grocery shopping. The use of detection, recognition, and tracking algorithms along with an interactive user-feedback system allows such persons to locate and pickup items they require from a pre-set shopping list.

This dissertation also contributes to the development of future visual systems through the exploration of emerging devices, novel architectures, and non-visual data modalities. Tunnel FETs, a promising device technology that is gaining prominence, have been evaluated in terms of their benefits relative to existing device technologies—MOSFETs and FinFETs. Novel architectures, such as coupled oscillator arrays, have been demonstrated here to provide reasonable implementations for many vision-related algorithms. Towards the realization of human-like intelligent vision systems, additional data modalities, such as audio, may be integrated into the pipeline enabling improved recognition and awareness.

Implementations based on these new devices, technologies, or modalities may all be easily be injected into the framework detailed in this work, reducing integration time and enabling rapid evaluation within the context of a fully-fledged vision system.

References

[1] N. D. B. Bruce, "Features that draw visual attention: An information theoretic perspective," *Neurocomputing*, pp. 125-133, May 2005.

[2] N. D. B., Tsotsos, J. K. Bruce, "Saliency Based on Information Maximization," *Advances in Neural Information Processing Systems*, pp. 155-162, June 2006.

[3] M. Riesenhuber and T. Poggio, "Hierarchical Models of Object Recognition in Cortex," *Nature Neuroscience*, pp. 1019-1025, 1999.

[4] C. Siagian and L. Itti, "Rapid Biologically-Inspired Scene Classification Using Features Shared With Visual Attention," *IEEE Transactions on Pattern Analysis and Machine Learning*, vol. 29, no. 2, pp. 300-312, Feb 2007.

[5] T. Serre and M. Riesenhuber, "Realistic Modeling of Simple and Complex Cell Tuning in the HMAX Model, and Implications for Invariant Object Recognition in Cortex," Massachusetts Institute of Technology, Cambridge, MA, CBCL Paper #239 2004.

[6] L. Itti, C. Koch, and E. Niebur, "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis," *IEEE Transactions on Pattern Analysis and Machine Learning (PAMI)*, pp. 1254-1259, 1998.

[7] L. Itti, "Models of Bottom-Up Attention and Saliency," in *Neurobiology of Attention*, L. Itti, G. Rees, and J. K. Tsotsos, Eds. San Diego, CA: Elsevier, 2005, pp. 576-582.

[8] L. Itti and C. Koch, "Target Detection using Saliency-Based Attention," in *Proc. RTO/SCI-12 Workshop on Search and Target Acquisition (NATO Unclassified)*, 1999, pp. 3.1-3.10.

[9] M. Riesenhuber and T. Poggio, "Models of Object Recognition," *Nature Neuroscience*, 2000.

[10] J. Mutch and D. G. Lowe, "Multiclass Object Recognition with Sparse, Localized Features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 11-18.

[11] J. Mutch and D. G. Lowe, "Object class recognition and localization using sparse features with limited receptive fields," *International Journal of Computer Vision (IJCV)*, vol. 80, no. 1, pp. 45-57, October 2008.

[12] M. Debole et al., "A Framework for Accelerating Neuromorphic-Vision Algorithms on FPGAs," in *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2011, pp. 810-813.

[13] (2010) iLab Neuromorphic Vision C++ Toolkit (iNVT). [Online]. http://ilab.usc.edu/toolkit/

[14] A. Torralba, A. Oliva, M. Castelhano, and J. M. Henderson, "Contextual Guidance of Attention in Natural scenes: The role of Global features on object search," *Psychological Review*, vol. 113, no. 4, pp. 766-786, October 2006.

[15] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin, "Context-based Vision System for Place and Object Recognition," in *IEEE International Conference on Computer Vision (ICCV)*, 2003.

[16] N. D. B. Bruce and J. K. Tsotsos, "Saliency, attention and visual search: An information theoretic approach," *Journal of Vision*, pp. 1-24, 2009.

[17] T. Serre and A., Poggio, T. Oliva, "A feedforward architecture accounts for rapid categorization," in *Proceedings of the National Academy of Science*, 2007, pp. 6424-6429.

[18] T. Serre, L. Wolf, and T. Poggio, "Object Recognition with Features Inspired by Visual Cortex," in *IEEE CS Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[19] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of Exemplar-SVMs for Object Detection and Beyond," in *2011 IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 89-96.

[20] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *European Conference on Computer Vision (ECCV)*, 2006.

[21] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000, pp. 142-149.

[22] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of the Imaging Understanding Workshop*, 1981, pp. 12-130.

[23] J-Y. Bouguet, "Pyramidal Implementation of the AÆne Lucas Kanade Feature Tracker," Intel Corporation, 2001.

[24] G. Nebehay, Robust Object Tracking Based on Tracking-Learning-Detection, 2012, Master's Thesis.

[25] G. Nebehay, B. Micusik, C. Picus, and R. Pflugfelder, "Evaluation of an online learning approach for robust object tracking," AIT Austrian Institute of Technology, Technical Report AIT-DSS-TR-0279, 2011.

[26] G. Nebehay and R. Pflugfelder, "Consensus-based Matching and Tracking of Keypoints for Object Tracking," in *Winter Conference on Applications of Computer Vision*, 2014.

[27] A. Torralba, "Contextual priming for object detection," *International Journal of Computer Vision (IJCV)*, vol. 53, no. 2, pp. 169-191, 2003.

[28] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *International Journal of Computer Vision (IJCV)*, vol. 42, no. 3, pp. 145-175, 2001.

[29] N. D. B. Bruce, Saliency, Attention and Visual Search: An Information Theoretic Approach, 2008, Ph.D. Thesis, York University.

[30] L. Itti, Models of Bottom-Up and Top-Down Visual Attention, 2000, Ph.D. Thesis, California Institute of Technology.

[31] T. Serre, L. Wolfe, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust Object Recognition with Cortex-Like Mechanisms," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 29, no. 3, pp. 411-426, 2007.

[32] S. Park et al., "System-On-Chip for Biologically Inspired Vision Applications," *IPSJ Transactions on System Design LSI Methodology (T-SLDM)*, vol. 5, pp. 71-95, August 2012.

[33] J. Mutch. (2011) HMIN: A Minimal HMAX Implementation. [Online]. http://cbcl.mit.edu/jmutch/hmin/

[34] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Computer Vision and Pattern Recognition*, 2005, pp. 886-893.

[35] Defense Advanced Research Projects Agency (DARPA). (2014) Neovision2 Solicitation. [Online]. https://www.fbo.gov/index?id=c955bdcd43727e05f4a4814db497baea

[36] Office of Naval Research (ONR). (2014) Supervised Autonomous Fires Technology (SAF-T) Fact Sheet. Document. [Online]. http://www.onr.navy.mil/~/media/Files/Fact-Sheets/30/SAF-T.ashx

[37] A. A. Maashri et al., "Hardware Acceleration for Neuromorphic Vision Algorithms," *Journal of Signal Processing Systems (JSPS)*, pp. 1-13, September 2012.

[38] J. Mutch, U. Knoblich, and T. Poggio, "CNS: A GPU-based Framework for Simulation of Cortically-Organized Networks," Massachusetts Institute of Technology, Cambridge, MIT-CSAIL-TR-2010-013/CBCL-286, 2010.

[39] NVIDIA. (2011) Tesla M200 Board Specification. Online Document. [Online]. http://www.nvidia.com/docs/IO/43395/Tesla-M2090-Board-

[40] G. Griffin, A. Holub, and P. Perona, "Caltech-256 Object Category Dataset," California Institute of Technology, Technical Report 7694., 2007.

[41] (2007) VOC2007 Preliminary Results. [Online]. http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/results/index.shtml

[42] (2014) Neovision2 annotated video datasets. Video. [Online]. http://ilab.usc.edu/neo2/dataset/

[43] M. Cotter, S. Advani, J. Sampson, K. Irick, and V. Narayanan, "A Hardware Accelerated Multilevel Visual Classifier for Embedded Visual-Assist Systems," in *IEEE/ACM 2014 International Conference on Computer-Added Design (ICCAD)*, 2014.

[44] T. Judd, K. Ehinger, F. Durand, and A. Torralba, "Learning to Predict Where Humans Look," in *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, 2009, pp. 2106-2113.

[45] M-M. Cheng, Z. Zhang, and W-Y., Torr, P. H. Lin, "BING: Binarized Normed Gradients for Objectness Estimation at 300fps," in *Computer Vision and Pattern Recognition*, 2014.

[46] J. Liu and Y. Liu, "Grasp Recurring Patterns from a Single View," in *Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2003-2010.

[47] B. Russell, A. Torrabla, K. Murphy, and W. Freeman, "LabelMe: A Database and Web-Based Tool for Image Annotation," *International Journal of Computer Vision*, May 2008.

[48] R. Mukundrajan et al., "Ultra Low Power Circuit Design Using Tunnel FETs," in *2012 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2012, pp. 153-158.

[49] R. Mukundrajan et al., "Design of energy-efficient circuits and systems using tunnel field effect transistors," *IET Circuits, Devices & Systems (IET-CDS). Special Issue: Design Methodologies for Nanoelectronic Digital and Analogue Circuits*, pp. 294-303, June 2013.

[50] H. Liu, M. Cotter, V. Narayanan, and S. Datta, "Technology Assessment of Si and III-V FinFETs and III-V Tunnel FETs from Soft Error Rate Perspective," in *2012 International Electron Devices Meeting (IEDM)*, 2012, pp. 25.5.1-25.5.4.

[51] M. Cotter, Y. Fang, S. P. Levitan, D. M. Chiarulli, and V. Narayanan, "Computational Architecture Based on Coupled Oscillators," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2014.

[52] S. Datta, N. Shukla, A. Parihar, A. Raychowdhury, and M. Cotter, "Neuro Inspired Computing with Coupled Relaxation Oscillators," in *51st Annual IEEE/ACM Design Automation Conference (DAC)*, 2014.

[53] N. Shukla et al., "Pairwise Coupled Hybrid Vanadium Dioxide-MOSFET (HVFET) Oscillators for Non-Boolean Associative Computing," in *2014 International Electron Devices Meeting (IEDM)*, 2014.

[54] S. P. Levitan et al., "Non-Boolean Associative Architectures Based on Nano-Oscillators," in *International Workshop on Cellular Nanoscale Networks and their Applications (CNNA)*, 2012, pp. 1-6.

[55] Y. Kuramoto, *Chemical Oscillations, Waves and Turbulence*.: Springer-Verlag, 1984.

[56] Y. Kuramoto, "Lecture Notes in Physics: Self-Entrainment of a Population of Coupled Non-Linear Oscillators," in *International Symposium on Mathematical Problems in Theoretical Physics*, 1975, p. 420.

[57] N. Chopra and M. W. Spong, "On Synchronization of Kuramoto Oscillators," in *European Control Conference on Decision and Control*, 2005, pp. 3916-3922.

[58] N. Chopra and M. W. Spong, "On exponential frequency synchronization of kuramoto oscillators," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, 2009.

[59] S. H. Strogatz, "From Kuramoto to Crawford: Exploring the Onset of Synchronization in Populations of Coupled Oscillators," *Physica D*, vol. 143, no. 1-4, pp. 1-20, 2000.

[60] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[61] G. Nebehay and R. Pflugfelder, "TLM: Tracking-Learning-Matching of Keypoints," in *International Conference on Distributed Smart Cameras*, 2013.

[62] M. Debole et al., "FPGA-Accelerator System For Computing Biologically Inspired Feature Extraction Models," in *Conference Record of the 45th Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 2011, pp. 751-755.

[63] A. A. Maashri et al., "Accelerating Neuromorphic Vision Algorithms," in *49th ACE/EDAC/IEEE Design Automation Conference (DAC)*, 2012, pp. 579-584.

[64] M. Cotter, H. Liu, S. Datta, and V. Narayanan, "Evaluation of Tunnel FET-Based Flip-Flop Designs for Low Power, High Performance Applications," in *2013 International Symposium on Quality Electronic Design (ISQED)*, 2013, pp. 430-437.

[65] H. Liu, M. Cotter, V. Narayanan, and S. Datta, "Evaluation of Soft Error Rate Immunity in Emerging Devices," in *Government Microcircuit Applications and Critical Technology Conference (GOMACTech)*, 2013.

[66] H. Liu, M. Cotter, S. Datta, and V. and Narayanan, "Soft Error Performance Evaluation on Emerging Low Power Devices," *IEEE Transactions on Device and Materials Reliability (TDMR)*, vol. 2014, pp. 732-741, June 2014.

[67] Y. Fang, M. Cotter, S. P. Levitan, and D. M. Chiarulli, "Image Segmentation Using Frequency Locking of Coupled Oscillators," in *14th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA)*, 2014.

[68] N. Chandramoorthy et al., "“Understanding the Landscape of Accelerators for Vision," in *2014 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2014.

[69] N. Chandramoorthy et al., "Exploring Architectural Heterogeneity in Intelligent Vision Systems," in *21st IEEE Symposium on High Performance Computer Architecture (HPCA)*, 2015.

[70] B. Smith et al., "Using a Visual Co-occurrence Network (ViCoNet) for Large-scale Object Classification," in *Sensors to Cloud Architectures Workshop (SCAW-2015)*, 2015.

VITA

Matthew Joseph Cotter

Ph.D. in Computer Science and Engineering, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA. Major field: Intelligent Visual Systems for Recognition. Research interests: Machine vision, Machine learning, multi-modal systems, hardware accelerators, emerging technologies, embedded systems. Dissertation: *Enabling Intelligent Vision Systems in a Configurable Multi-Algorithm Pipeline*. Chair: Vijaykrishnan Narayanan. **May 2015**.

Bachelor of Science, Computer Engineering, The Pennsylvania State University, University Park, PA. **December 2008**.

Debole, M., Xiao, Y., Yu, C-L., Maashri, A. A., Cotter, M., Chakrabarti, C., Narayanan, V. "FPGA-Accelerator System For Computing Biologically Inspired Feature Extraction Models". 2011 Conference Record of the 45th Asilomar Conference on Signals, Systems and Computers (ASILOMAR). pg. 751-755. Nov. 6-9, 2011.

Debole, M., Maashri, A. A., Cotter, M., Yu, C-L., Chakrabarti, C., Narayanan, V. "A Framework for Accelerating Neuromorphic-Vision Algorithms on FPGAs". 2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). pg. 810-813. Nov. 7-10, 2011.

Maashri, A. A., Debole, M., Cotter, M., Chandramoorthy, N., Xiao, Y., Narayanan, V., Chakrabarti, C. "Accelerating Neuromorphic Vision Algorithms for Recognition". 2012 49th ACE/EDAC/IEEE Design Automation Conference (DAC). pg. 579-584. June 3-7, 2012.

Park, S., Maashri, A. A., Irick, K. M., Chandrashekhar, A., Cotter, M., Chandramoorthy, N., Debole, M., Narayanan, V. "System-On-Chip for Biologically Inspired Vision Applications". IPSJ Transactions on System Design LSI Methodology (T-SLDM) Volume 5. 2012. pg. 71-95. August 6, 2012.

Maashri, A. A., Cotter, M., Chandramoorthy, N., Debole, M., Yu, C-L., Narayanan, V., Chakrabarti, C. "Hardware Acceleration for Neuromorphic Vision Algorithms". Journal of Signal Processing Systems (JSPS), September 2012. pg. 1-13. Sept. 1, 2012.

Chandramoorthy, N., Swaminathan, K., Cotter, M., Li, X., Narayanan, V., Palit, I., Hu, S., Niemier, M., Irick, K. "Understanding the Landscape of Accelerators for Vision". 2014 IEEE International Workshop on Signal Processing Systems (SiPS). Belfast, UK. October 2014.

Cotter, M., Advani, S., Sampson, J., Irick, K., Narayanan, V. "A Hardware Accelerated Multilevel Visual Classifier for Embedded Visual-Assist Systems". IEEE/ACM 2014 International Conference on Computer-Added Design (ICCAD). San Jose, CA. November 2014.

Chandramoorthy, N., Tagliavini, G., Irick, K., Pullini, A., Advani, S., Al Habsi, S., Cotter, M., Sampson, J., Narayanan, V., Benini, L. "Exploring Architectural Heterogeneity in Intelligent Vision Systems". 21st IEEE Symposium on High Performance Computer Architecture (HPCA). San Francisco, CA. February 2015.

Smith, B., Advani, S., Cotter, M., Irick, K., Sampson, J., Narayanan, V. "Using a Visual Co-occurrence Network (ViCoNet) for Large-scale Object Classification". Sensors to Cloud Architectures Workshop (SCAW-2015). San Jose, CA. February 2015.