Real Time Sorting of Plastic Recyclables Using an FPGA based SVM

REAL TIME SORTING OF PLASTIC RECYCLABLES USING AN FPGA BASED SVM

BY

BRYAN HOUSE, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2011)	McMaster University
(Electrical & Computer Engineering)	Hamilton, Ontario, Canada

TITLE:	Real Time Sorting of Plastic Recyclables Using an FPGA based SVM
AUTHOR:	Bryan House B.Eng., (Engineering Physics) McMaster University, Hamilton,Canada
SUPERVISORS:	Dr. David Capson Dr. Derek Schuurman
NUMBER OF PAGES:	xiv, 103

I would like to dedicate this thesis to my family who has shown me the utmost support in everything I do.

Abstract

The amount of recyclable material being processed worldwide is increasing. There is a demand for new technologies that can quickly sort these materials for maximum purity while maintaining high throughput. This thesis proposes a method to automatically sort two materials: Polycoat containers and Polyethylene terephthalate (PET) bottles. This method utilizes a visible light camera and does not rely on Near-Infrared spectrometry. A high-speed method to automatically locate regions that likely contain these materials within the image and remove them from the background is presented. These regions are merged into whole containers and are classified as either a Polycoat container or PET bottle. This is accomplished using a linear support vector machine (SVM) trained on the histogram of pixel intensities. A novel graph theoretic based region growing technique is proposed and experimental results are provided to characterize the system. The proposed method obtained a 93% recognition rate while running in real-time on an FPGA.

Acknowledgements

First and foremost, I would like to acknowledge the tireless work of my supervisors Dr. David Capson and Dr. Derek Schuurman. Without their guidance, knowledge, support and endless patience, this thesis would have never been possible. I owe them a great deal of gratitude that I could not hope to express in such few words.

I would also like to thank my colleagues of the Computer Vision Lab for endless thought provoking conversations, midnight coffee runs and most of all, their friendship. Without them, my time spent completing my Masters would have not been nearly as enjoyable.

Finally I would like to thank everyone who has helped me throughout this process but are far too many to enumerate. Your kindness will not be forgotten.

Notation and abbreviations

α	Lagrange Multiplier
ALM	Adaptive Logic Module
b	bias
$C(X, Y, F(X))$	Cost function
EPIC	Environment and Plastics Industry Council
$F(X)$	Classifier
FPGA	Field Programmable Gate Array
FPS	Frames per second
h	VC dimension
$H(X, Y, F(X))$	Hinge-Loss function
HDL	Hardware Description Language
HDPE	High-density polyethylene
k	Region merging parameter
K()	Kernel function
KKT	KarushKuhnTucker conditions
LE	Logic Element
LUT	Look-up-Table
M	Margin
MRF	Material Recovery Facility
N	Dimensional of feature vectors
n_{bits}	Number of bits used to represent a pixel
NIR	Near-Infared
$\Phi(X)$	Non-linear mapping
PET	Polyethelne terephaleate
R^2	2 Dimensional Euclidean Space
R^3	3 Dimensional Euclidean Space
R_{emp}	Empirical Risk
SPI	Society of Plastics Industries

SVM	Support Vector Machine
$T(R)$	Threshold function
w	Normal vector
w_{ij}	Edge weight between node i and j
X	Feature vector
X^-	Feature point on negative hyperplane
X^+	Feature point on positive hyperplane
Y	Label
(X_i, Y_i)	Set of feature vectors and labels
ζ	Slack variables
$Z(X, Y, F(X))$	Zero-One loss function

Contents

Abstract **iv**

Acknowledgements **v**

Notation and abbreviations **vi**

1 Introduction and Problem Statement **1**

1.1 The Importance of Accurate Sorting in Recycling Plastics 1

1.1.1 Recycling Of Plastics 2

1.1.2 Recycling Rates 5

1.1.3 Collection 7

1.2 Sorting 9

1.2.1 Resin Codes 9

1.2.2 Automatic Sorting 12

1.3 Problem Statement 16

1.4 Thesis Contributions 18

1.5 Chapter Summary 18

2 Background **20**

2.1 Supervised Pattern Recognition 20

2.1.1 Risk 22

2.1.2 Shattering 27

2.2 Support Vector Machines 28

2.2.1 Hyperplanes 28

2.2.2 Maximum Margin Hyperplanes 29

2.2.3 KKT and Optimization 33

2.2.4 Non-Linearly Separable Data 35

2.2.5 Kernel Functions 38

2.2.6 Support Vector Machines and Computer Vision 41

2.3 Field Programable Gate Arrays (FPGA) 42

2.3.1 FPGAs in Computer Vision 46

2.4 Chapter Summary 48

3 Algorithm Overview and Hardware Implementation 49

3.1 Introduction 49

3.2 Experimental Setup 50

3.3 Algorithm 51

3.3.1 Block Construction 55

3.3.2 Background Removal 57

3.3.3 Region Growing 58

3.3.4 SVM Training 61

3.4 Hardware Implementation 61

3.4.1 Data Serializer 62

3.4.2 Sub-Block Creation 63

3.4.3 Background Classifier 64
3.4.4 SVM 65
3.4.5 Dot Product Module 67
3.4.6 Region Growth 68
3.5 Chapter Summary 72

4 Results and System Performance 74

4.1 Implementation and Testing 74
4.2 Results 75
4.2.1 Timing 80
4.3 Effect of Block Size 81
4.3.1 Region Growth 82
4.3.2 Classification 85
4.3.3 Hardware Implementation 88
4.3.4 Throughput 90
4.3.5 Overall Classification and Number of Significant Pixel Bits . . 92
4.4 Effect of k 93
4.4.1 Segmentation Results 93
4.4.2 Throughput 95
4.4.3 Chapter Summary 96

5 Conclusions and Future Work 97

5.1 Future Work 99

List of Figures

1.1 Bales of sorted PET bottles packed for selling 4

1.2 Resin codes as identified by the Society of the Plastics Industry (SPI) 10

1.3 Process for recyclable classification 17

2.1 Two examples of features used to separate two classes in an arbitrary feature space. In a) the features allow for the separation of the data into two distinct classes. b) the features do not allow for the separation into different classes 21

2.2 Three examples of potential classifiers for data in Figures 2.1a). a) a simple classifier which might not generalize well. b) a more complex classifier that appears to follow the data. c) a complex classifier that makes no mistakes but appears to over fit the data 22

2.3 Two examples of linear classifiers for data presented in Figures 2.1a). b) correctly classifies more points than a) 25

2.4 Underlying structure can influence the data. Two regressions are shown for the data in this feature space. Both regressions are exactly correct on the training data but both are clearly not right and predict very different future values 26

2.5 All 8 possible labels are shown for 3 points and how they can be classified with a linear classifier. 28

2.6 Two parallel plane next to the hyperplane 31

2.7 Final hyperplane and the support vectors 34

2.8 This data is non-linearly separable due to the overlap of the two sets of points. This data set will cause the previous derivation to fail since no solution can satisfy all constraints 35

2.9 Using a slack variable, the requirement can be eased such that every point must be classified correctly. It now becomes a problem of minimizing the normal vector,w, and the slack variables 37

2.10 Data can be arranged in 1D to be non-linearly separable. There is no meaningful linear classifier for this data 39

2.11 By allowing non-linear basis function the previous data now become linearly separable. No new data was added; the data is merely projected into a new higher dimensional space 40

2.12 Typical FGPA architectue with logic blocks, I/O, and interconnects . 43

2.13 Typical LE configuration in an Altera Arria II GX device 44

3.1 Experimental Setup (PC not shown) 51

3.2 Overview of Proposed Algorithm 53

3.3 Typical Polycoat container and histogram 54

3.4 Typical PET bottle and histogram 55

3.5 Enlarged Boundary Region of a PET Bottle. The red box contains similar pixels to the blue box, the only difference is the presence of the specular reflection 56

3.6 Graph used in region growing technique. Each node N corresponds to a 10×10 pixel block, with edges connected to the four neighboring blocks. Each edge has weight w_{ij} equal to the dot product between the histograms of node i and node j 59
3.7 Overview of the FPGA implementation 62
3.8 Histogram Construction Hardware Implementation 64
3.9 Background Averaging Filter 65
3.10 Block diagram of the SVM module 66
3.11 Dot product hardware module 68
3.12 Block diagram of the region growing module 69
3.13 Hardware Module that computes edge weights for region growing . . 70
3.14 Packet structure 70
3.15 Example Radix Sort 71
4.1 Example images 76
4.2 Results of the region growing method and classifier for several test images. The first column shows input image. The second column contains the result of the region growing (shade indicates region). The third column contains the result of the classifier. White regions indicates PET bottles and grey regions indicate Polycoat containers. This Figure is reproduced from work published in the IEEE International Symposium on Sustainable Systems and Technology(House *et al.*, 2011) 78
4.3 Simulation results with timing information for each section of the algorithm 80
4.4 Input Image 82

4.5 Results for block size 5x5,8x8,10x10,16x16, and 20x20 83
4.6 The True Positive rate for classification of PET bottles for 3 different histogram sizes 84
4.7 The True positive rate for classification of Polycoat bottles for 3 different histogram sizes 85
4.8 True and false positive rates vs. block size for background classifier . 86
4.9 Number of LEs Required vs. Block Size 87
4.10 The amount of embedded RAM required to implement the design vs. block size. The red dotted line is the amount of RAM available on the Arria II GX chip and thus anything over this value will not fit on the FPGA but can still be simulated 88
4.11 Frame rate vs. block size 91
4.12 Average accuracy vs. block size and number of histogram bits 92
4.13 Optimum k value vs. block size 94
4.14 Recognition rate vs. k for an 8×8 block size 95
4.15 Fps vs. block size for different values of k 96

Chapter 1

Introduction and Problem Statement

1.1 The Importance of Accurate Sorting in Recycling Plastics

The primary goal of the recycling process, with respect to plastics, is to divert material that would otherwise be disposed of and convert it into a polymer that can be reused in consumer products and packaging. By using recycled polymers, instead of virgin material, the energy required in the manufacturing process can be reduced which lowers costs and mitigates the environmental footprint. It is a known fact that the purity of the reprocessed material determines what applications that it can be used in. Highly pure reprocessed material can offset the most virgin material and has the highest savings in cost and energy, while less pure material must be used for lower grade products. Thus one of the primary concerns of a reprocessor is to produce material of the highest purity. Additionally it is of significant financial importance that the material be as pure as possible, and command the highest selling price. Thus the separation of plastic materials from would-be contaminates (which often include other types of plastics) is critical. Currently several methods have been proposed and working examples are employed in Material Recovery Facilities (MRFs) around the world. However these methods are not without their drawbacks and limitations.

The amount of recycling is increasing worldwide with an increase in population. There is also increased awareness and willingness to participate in recycling programs. This is helped by initiatives such as single-stream or dual-stream recycling, which are less demanding on households but offsets the sorting burden onto the reprocessors. The extra sorting effort can be helped by the adoption of fast and accurate sorting techniques that allow the MRFs to produce the highest quality recycled plastics while maintaining the required throughput.

In addition to the increased amount of recycling, new packaging, in the form of novel polymers or hybrid containers composed of several different materials, are beginning to find their way into common use and inevitably into the recycle stream. These new packages require novel techniques for optimal sorting such that they do not end up contaminating existing material streams, but are themselves reprocessed or disposed of properly.

1.1.1 Recycling Of Plastics

The recycling of plastics has become a common practice and is in line with the mantra of sustainability and green living. It has become almost synonymous with the new green age, being one of the famous three Rs (Reduce, Reuse, and Recycle). However the process from being left on the curb side to being incorporated into a new container is still only vaguely understood by and mostly hidden from the common person. There is still much confusion over what is recyclable or what can be processed by a local MRF, despite efforts such as resin codes and increased educational information. This confusion requires that great care is taken to sort the waste so as not to contaminate the supply of new materials.

Recycling is a required step towards sustainability. It allows for the recovery of raw materials that would otherwise be wasted, and to recoup the energy associated with the transportation, extraction, and manufacturing in transforming raw materials into a consumer packaging. It has been estimated that recycling of PET bottles can reduce greenhouse gas emissions by 0.42 metric tonnes of carbon dioxide and save 53 million BTUs of energy per tonne of recycled material (Morawski, 2009).

The biggest benefit to the environment is when the material can be upcycled which means it can be used as feed material in a manufacturing process that was of the same style as the orginal product. This directly displaces the virgin material from the process and reduces the overall amount of material required. Plastics can often have a life cycle of 15-20 products before they have to be downcycled to a lower grade product (Morawski, 2009). Downcycling is when a plastic is recycled into a product that is of lesser quality and that is eventually not meant to be recycled. An example of this downcycling is when PET bottles are recycled into plastic carpet or filler in sleeping bags. These products are not intended to be recycled and are ultimately disposed of in landfills or incinerators. This is an inevitable step due to the accumulation of microscopic defects within the plastic (Selke, 2006). Figure 1.1 shows sorted and packaged bales of PET at the Hamilton, ON MRF.

Figure 1.1: Bales of sorted PET bottles packed for selling

A large determining factor of where a recycled material ends up is its purity and chemical properties. Often, contamination results in undesirable changes to the chemical and physical properties of the polymer (Morawski, 2009),(Selke, 2006). When different polymers are mixed together they do not, in general, form a solution, but will separate into different regions (Selke, 2006). Microscopically this manifests as separate domains or pockets of one plastic in another, which has a detrimental effect on the mechanical and chemical properties of the plastics (Selke, 2006).

One such classic example of contamination that must be avoided is PVC in recycled PET plastic. Small quantities of PVC in PET (or vice versa) in the range of 4-10 ppm can have a large detrimental effect on the batch to be recycled (Selke, 2006). When the material is recycled, the contaminant will often turn into visible black flakes within the desired material (Selke, 2006). These black flakes significantly reduce the quality of the produced polymer and thus limit its application and selling price.

Not only are the largest environmental effects realized when pure material is obtained from the recycling process, but there is an economic incentive to produce high purity materials. Market price is directly proportional to material purity with the purest materials demanding the highest selling prices (Morawski, 2009). Since MRFs end goal is to sell the reprocessed materials that they produce, which either offsets the cost of running the recycling program for local municipalities or produce a profit for privately contracted firms, the highest price per quantity of material is desired. Materials with physical and chemical properties closer to virgin material will fetch the highest prices due to their ability to be substituted as feed material in many different products.

1.1.2 Recycling Rates

It has been a general trend over past decades that the amount of solid waste produced, and accordingly, the amount of material that is being processed for recycling has increased. The total amount of solid waste produced by Canadians in 2006 was 35 mega tonnes up from 30 mega tonnes in 2002 (Smith, 2009). Of all the solid waste produced, 7.7 mega tonnes (22% of the produced material) were processed for recycling in 2006, compared to 6.64 mega tonnes since 2002 (Smith, 2009). Although the percentage of recycled material has stayed relatively constant, the amount of material has grown by 9%.

Not only has the amount of material increased over the years, but more households have access to and are recycling than in the past. The availability of recycling programs of every type and the percentage of those who take advantage of them has risen dramatically in recent years within Canada. This reflects a changing attitude and greater education of the importance of recycling. This will have a significant impact on MRFs, as they will have to process larger quantities of solid waste and at higher speeds to keep up with the demand. This is where the use of novel technologies can ease the burden and help MRFs deal with the challenge of reprocessing the ever increasing volume solid waste.

In Canada, the percentage of households that had access to at least one type of recycling program (paper, metal, or plastic) was just over 70% in 1994 (Marshall, 2006),(A. Babooram, 2007). As of 2006 this number had increased to 93% of households with access to at least one form of recycling (Smith, 2009). Also the number of household that are taking advantage of these facilities has risen as well. Previously only 85% of households that had access to recycling facilities used them in 1994. The number has since grown to 97% in 2006 (Marshall, 2006),(A. Babooram, 2007).

In the USA, the numbers are similar in terms of growth. The amount of solid waste produced in 2003 was 236 million tons, which has grown to 250 million tons in 2008 (Selke, 2006),(Enviromental Protection Agency , 2009) ,(Enviromental Protection Agency, 2003). This is substantially more waste then the amount produced in Canada, which can be explained by the difference in population. The recycling rate for all materials is 33.2% up from just 29.0% in 2000 (Enviromental Protection Agency , 2009),(Enviromental Protection Agency, 2003).

Despite the growing trends of increased recycling, there is still room for increased amounts for recycling of plastics. Of the material processed for recycling in 2002, only 0.144 mega tonnes was plastic recyclables(Morawski, 2002). An EPIC (Environment Plastic Industry Council) study in 2002 (Morawski, 2002) estimates that 0.085 mega tonnes of this was comprised of plastic bottles. This represents a recycling rate of only 36% of the total 0.235 mega tonnes of plastic bottles that were produced in that year. The recycling rate taking into account all plastics will be much lower.

Again a similar situation can be found in the USA where the overall recycling rate of plastics was 5.2% in 2003, but for containers and bottles the rate was 38.6% in the same year (Selke, 2006). Its hard to compare with other countries due to differences in definitions of solid waste, but in western Europe, the recycling rate of plastic varies from 2.2% in Greece to 27.1% in Germany (Selke, 2006).

It is clear that not only the amount of material has increased, but also the number of people who are recycling. However, there is still considerable room for growth, especially in the area of plastic recycling. In order to encourage more participation, and to reduce cost to the municipalities, new curb side collection programs are starting to be implemented. Single-stream or commingled recycling is now being offered in more places. These programs allow the home owners to place all recycled goods within one receptacle. This has been shown to increase participation and reduce upfront collection costs. However much of the upfront savings are reduced by the increased workload of the reprocessors.

1.1.3 Collection

There are several ways in which the material can be collected at the curb, which is mainly decided by the municipality or the agency collecting the solid waste. The most restrictive is *source separation* which requires that each of the materials be sorted at the curb-side before pick up. The opposite end of the spectrum is *single stream* in which all materials are placed in the same receptacle. Many places fall in between the two extremes by employing dual stream systems with certain materials being lumped together. A common example is paper separated from containers, usually plastic, metal and glass.

Source separation can be expensive from a municipality point of view if they are only concerned with the collection (Selke, 2006),(Morawski, 2009),(Eureka Recycling, 2002). This is because either different trucks are required for different materials, or trucks with different compartments for keeping the materials separated. Processing the material is generally cheaper however due to the fact that most contaminants have already been sorted out by the time the trucks arrive at the MRFs. In addition, the material produced is often more pure than other methods, so it can be sold for a higher price. This collection style generally sees the least participation rates due to the required effort on the part of the community (Selke, 2006),(Morawski, 2009),(Eureka Recycling, 2002).

Single stream systems are much less restrictive; all materials that are able to be processed are collected together in a single receptacle. This reduces collection costs because the same trucks can be used for all materials, and no division is required (Selke, 2006),(Eureka Recycling, 2002). However the overall cost of the processing is significantly higher due to the extra sorting that is required (Morawski, 2009),(Eureka Recycling, 2002). Home owners are more likely to rate themselves as satisfied with this method and overall recycling participation rates are higher due to the ease of sorting. However, the chance of materials ending up contaminated or improperly sorted as garbage are much higher, and thus profits are less for the MRFs compared to dual streams (Selke, 2006),(Morawski, 2009),(Eureka Recycling, 2002).

Single Stream Recycling is on the rise; in the USA there were 160 single stream municipalities as of 2009, up from just 70 in 2005 (Morawski, 2009). This will put further burden on the MRFs to deal with the extra sorting required to ensure that top quality polymers are produced. The secondary effects will be to increase the recycling rate and therefore the amount of material to sort. This leaves the MRFs in the position with more material to sort through at higher contamination rates. Automatic sorting will need to be used in order to allow the MRFs to maintain a high purity material and maintain the throughput to deal with the increase in solid waste.

1.2 Sorting

1.2.1 Resin Codes

The first line of sorting happens at the curb-side either in the form of source separation where the home owner will do an initial sort and/or in terms of rules in what can and cannot be recycled. To aid the common person in identifying what the local MRF is able to process, the SPI (Society of Plastic Industries) developed a resin code system. This code was designed to help quickly determine what is recyclable without having to memorize the individual products and packaging. Figure 1.2 shows the different resin codes.

All the recyclable plastics are given a number which must be present on the packaging. This number identifies which resin type the plastic container is made of.

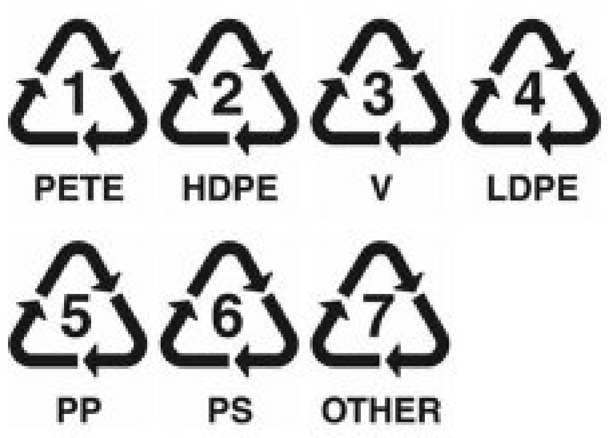

Figure 1.2: Resin codes as identified by the Society of the Plastics Industry (SPI)

The MRF will then outline which numbers of resin codes should be placed in the bins for recycling.

The resin code system is not perfect though, as materials with the same resin codes cannot always be mixed together in the final sorting. This is often because of differing additives required in the manufacturing process. Several containers are made of composites of more than one type of resin, which means they are marked by two or more numbers. The resin code 7 is merely a catch all for all other types of plastics. This means that it is impossible to tell what type of plastic it is made of by using the resin codes only. These container types are generally not recycled.

At the MRF another sort must be done to remove contamination in the form of dirt and debris, as well as to sort the plastics by resin. Each plastic must be sorted by the resin type and generally cannot be mixed (Selke, 2006). However, there are additional considerations when the materials are grouped together. Simply having the same resin code does not guarantee that they can be processed together.

Different manufacturing methods or working environments often requires polymers with different chemical or physical properties. These properties can be controlled by the use of additives. These additives will often contaminate the recycling process within the same polymers.

Not only must each resin must be sorted from each other, but they must also be sorted by colour within a resin code (Selke, 2006). This is because the colour will often remain after the recycling process and be visible in the new product. This can be mitigated by using mechanical recycling, where a layer of off-colour material is sandwiched between two virgin material layers (Selke, 2006). However, to obtain the highest price and purity the materials must be sorted by colours.

Other additives are used in the manufacturing process to give strength or other desirable properties to the material. For example black microwave dinner trays are injection molded and must have a working temperature range from well below freezing to the operating temperatures produced in a microwave oven. The additives required to make the polymer work in such environments makes them incompatible with PET that is blow-molded into plastic bottles (Selke, 2006). It also makes them opaque and black, which makes them difficult to sort (Selke, 2006).

Despite recent technological advancements, a large proportion of the overall sorting is done by humans (Selke, 2006),(Bruno, 2000). Manual sorting of plastics is done by trained workers that either identify the type of material by memorizing products or by the resin code. Generally the workers will take the object off a conveyor belt and place it into the proper receptacle. Manual sorting suffers from high turnover rate among employees, which makes it difficult to keep properly trained sorters available. This makes automatic sorting more attractive from a MRF's point of view, because fewer sorters are needed and a large cost savings can be realized (Bruno, 2000). The process will inevitably go through some form of human quality control despite the presence of automatic sorting (Selke, 2006),(Bruno, 2000).

1.2.2 Automatic Sorting

Automatic sorting machines are becoming more common in MRFs due to the reduction in cost and time required to sort materials. There are a variety of types that use a wide range of technologies depending on the type of material to be sorted. The emphasis of this chapter and the thesis as a whole will be on macrosorting techniques. Another class of automatic sorting techniques is microsorting, in which the material is usually cut or ground into flakes and subjected to density based separation to mechanically separate the plastics types. These methods are rarely used in practice (Selke, 2006).

Macrosorting is applied to the material as it comes into the facility, either after or before a wash is performed. The material is usually loaded onto a conveyor belt and it is tested using one of the methods described below. Some rejection method, such as air knife or trap door, is used to remove material that is classified as the wrong type, or to move it onto another line for further processing. The cost of setting up these machines depends on the type of technology employed, throughput, and number of items to sort. The set-up cost is generally much higher than a manual sort but this is offset by both the reduction of reoccurring costs (wages, benefits, retraining, etc) and the increase of sale prices due to higher-levels of purity (Selke, 2006),(Bruno, 2000).

Some of the different technologies for automatic macrosorting are listed below. All of these methods have current working examples around the world.

X-Ray Fluorescence Spectroscopy

X-Ray fluorescence can be used to detect different functional groups (specific combinations of atoms) within a polymer. Certain functional groups will absorb certain wavelengths in the X-ray spectrum, thus by measuring the amount of X-rays passing through; one can tell if a particular wavelength of X-ray has been absorbed. By comparing the results to precompiled charts, the presence of specific functional groups can be determined (Selke, 2006),(Bruno, 2000).

This method is mainly used for detection of PVC due to the chlorine atoms response to X-rays frequencies. It is an important detection method because it is used to keep the PVC from contaminating PET which is an important recycled plastic. As mentioned earlier, even small quantities of PVC will contaminate large quantities of PET or vice versa (Selke, 2006),(Bruno, 2000).

Due to the use of X-rays, these machine require special training to use them along with bulky safety equipment (Selke, 2006),(Bruno, 2000). This makes them relatively expensive in light of the fact that they only are able to sort out one type of plastic.

Near-Infrared Spectroscopy

Near-Infrared (NIR) spectroscopy is similar to X-ray fluorescence in that it also examines the absorption of a specific type radiation by the material. The infrared radiation causes the atoms within the polymers to vibrate (Hsu, 1997). Since each of the resin's atomic structure varies, different vibrational modes are observed. If an incident photon has a similar frequency to a mode of vibration, or is a multiple of the fundamental frequency, then it will be absorbed (Hsu, 1997).

The result is that certain wavelengths will be noticeably attenuated when observed with an appropriate sensor. This results in a pattern that can easily distinguish different types of polymers. In academic literature several methods for discriminating these spectra have been proposed. LDA (Linear Discriminant Analysis) and QDA (Quadratic Discriminant Analysis) (R. Leitner, 2003), artificial neural networks (D.M. Scott, 1995) and SVMs (Support Vector Machines) (A. I. Belousov, 2002) have been suggested as possible methods of discriminating between different types of plastics using NIR spectroscopy.

NIR spectroscopy is the most widely used technology to distinguish between different polymers (EPIC, 2008). In most commercial system this technology is employed either by itself or with the aid of machine vision systems utilizing NIR cameras. This approach is preferred because of its accuracy and speed; several scans can be taken of an object as it passes in order to increase the probably of correct identification (Bruno, 2000).

There are however, a few drawbacks to NIR systems. One is that black coloured plastics are almost impossible to detect using these systems. This is because the black colouring has good absorption in the NIR region, which results in a flattened absorption spectrum (Selke, 2006). Common black plastic containers include motor oil containers, as well as injection moulded microwave dinner containers. The flattened spectrum then contains relatively little information for the classification of the polymer.

Since the spectrum is obtained by the absorption of different wavelengths, if two or more materials are overlapping completely then they cannot be correctly identified because the absorption profile produced will be a combination of both materials (Bruno, 2000),(Selke, 2006). As long as some of the material is exposed this method will work, so labels and debris do not impede this method. However this requires that material generally be fed in such a way as to ensure that the material is in a single layer. By arranging the items in a single layer, it is guaranteed that one item will not be completely covered by another object.

Certain other types of containers consisting of two materials are also troublesome for this method. Polycoat or aseptic containers are cardboard coated with a thin layer of high density polyethylene (HDPE) or aluminum foil to make them able to hold liquids. Since the NIR radiation is absorbed by both materials, the spectra produced is hard to sort from existing spectrum and the material is often incorrectly identified.

Other types of containers have only started to be tested against this detection method. New biodegradable polymers that will decompose under special conditions have been produced in recent years. Polylactic acid (PLA) has started to show up in a few commercial products, and in some cases it is visually indistinguishable from other plastics types, specifically when it is in clear bottle form. Nature Works, a company that produces PLA bottles, has performed a few tests on how this will impact the recycling stream, with some preliminary success (Sawyer, 2009).

Machine Vision

A recent study by Environment and Plastic Industrial Counsel (EPIC) (EPIC, 2008) has shown several manufactures that incorporate some form of a machine vision system into their processing. The exact details of the algorithms are proprietary, but they are often used to detect coloured bottles along with aiding in identification using shape, size, and texture analysis. These methods are sometimes performed using special cameras and/or filters to examine different wavelengths of the NIR spectrum.

The academic literature in the area of computer vision to aid in the recycling process is lacking. Several approaches have been suggested; (Ramli *et al.*, 2007) suggests using boundary to describe the plastic bottle types. In (Ramli *et al.*, 2010), partial erosion is introduced as a method to describe the boundary of plastic bottles, and Shahbudin *et al.* (2010) develops this into a feature that an SVM is trained on to sort PET bottles. In (Ramli *et al.*, 2008), the idea of using histograms of different sections of the bottle to identify the material is proposed.

1.3 Problem Statement

The problem this thesis addresses is the design of a process to sort two types of containers in a mixed stream. The two materials to be sorted are Polycoat containers and PET plastic bottles. Polycoat containers are constructed of a layer of paperboard between two layers of HDPE film. Due to their composite structure they are difficult to detect in a PET stream and some facilities still hand sort these products.

Previous work in this area has focused on the shape and size variation between different bottle types. However these features are deemed to be of limited usefulness in a practical system because shape is not necessarily constant, as bottles can arrive crushed or cut up which limits the usefulness of this feature.

Instead, optical properties of the materials are used because they will remain relatively constant despite the state of the container. The two main optical properties that are examined to classify the container are transparency and reflectivity. Transparency can be measured by imaging against a black background, while reflectivity is measured by illuminating the containers with a high-intensity light. By examining the pixels that make up the image, and constructing a histogram of pixel intensities,

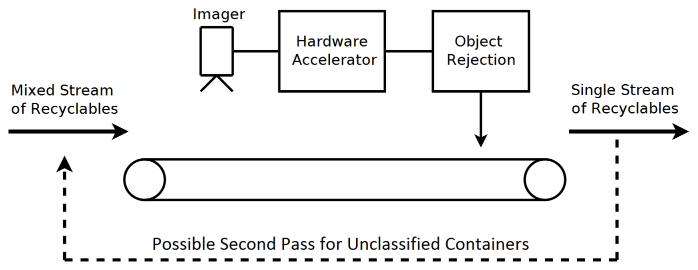
Imager
Hardware Accelerator
Object Rejection
Mixed Stream of Recyclables
Single Stream of Recyclables
Possible Second Pass for Unclassified Containers

Figure 1.3: Process for recyclable classification

these material properties can be measured indirectly.

Previous work on this subject was presented in (Nawrocky *et al.*, 2010) where histogram of pixel intensities were used as inputs to a linear SVM for classification purposes. The main limitation with this work was that only single bottles could be classified at a time because a simple bounding box segmentation algorithm was used.

This works extends the ideas presented in (Nawrocky *et al.*, 2010) by removing the unrealistic constraint of classifying single bottles. The proposed method utilizes novel background removal and segmentation methods to achieve this result. By identifying regions that are not part of the background and then growing these regions into different containers, many bottles can be imaged at once and classified by using their histograms. Ideally this would be part of a system which is able to remove Polycoat containers from a stream of PET bottles.

Of considerable importance to MRFs would be the throughput of such a system. Thus the proposed algorithm will be implemented in hardware to allow for real time operation with high frame rates. The developed hardware accelerator will accept visible light image data and classify the containers based on the optical properties of the two materials.

Figure 1.3 shows the setup of the machine. Several machines can be cascaded together to allow for a more complete sorting or non-classified bottles could be added back to the stream for reclassification. In this way each pass would become more accurate as less bottles are present to add clutter.

1.4 Thesis Contributions

The two main problems that were dealt with in this thesis were the extension of the classification algorithm to multiple containers within one image and high speed recognition. The first problem was solved by implementing a novel graph based segmentation method to extract multiple container regions within an image. The segmentation method uses the histograms of smaller blocks as a similarity metric to group regions together. The second problem of speed was addressed by implementing the entire algorithm in an FPGA. This allowed for the parallel nature of the algorithm to be exploited and a real time implementation to be achieved.

1.5 Chapter Summary

Recycling is an efficient means to recover spent manufacturing energy and materials from consumer goods that would otherwise go to waste. However, to be economical these materials must be sorted to a high degree of accuracy. Chapter 1 introduced the importance of fast, accurate sorting of recyclables and why this is a pressing problems. It demonstrated the requirement for high speed accurate sorting due to the large volumes of recyclables that need to be processed and the many materials that can end up in the processing stream. Previous work in this area has been found to be lacking in academia, although many examples of industrial machines are currently operating. This thesis grows off other work that is similar in scope, but extends the problem one step closer to a real-world implementation by increasing the speed of processing and the number of containers that can simultaneously be processed.

Chapter 2 will build the concepts required to understand the processing tools and implementations of the solution to this problem. Pattern recognition is introduced, with the specific example of the Support Vector Machine (SVM). The underlying mechanics of this classification tool are explained and justifications for its use are presented. A brief introduction to FPGAs and how they will be used to obtain real-time results is also included.

Chapter 2

Background

This chapter introduces the background subject matter from which this thesis draws. The first section provides some important concepts in supervised pattern recognition. It also introduces the powerful tool of the support vector machine which is used for classification in this thesis. The second section briefly introduces FPGAs and demonstrates their utility in computer vision and machine learning tasks.

2.1 Supervised Pattern Recognition

Supervised pattern recognition deals with developing methods to classify new data by generalizing from existing data. More formally, given a pattern X_{new} the goal is to assign a label Y_{new} by finding a mapping of $F(X) \mapsto Y$. In supervised learning this is done by using previous data X_i with known labels $Y_i \ \ \forall i$. This set of data (X_i, Y_i) is referred to as the training set. When there are only two possible labels, the pattern classification is called binary. For mathematical simplicity these are given values of (-1,1). An example is presented in Figure 2.1 of the same class with two different sets

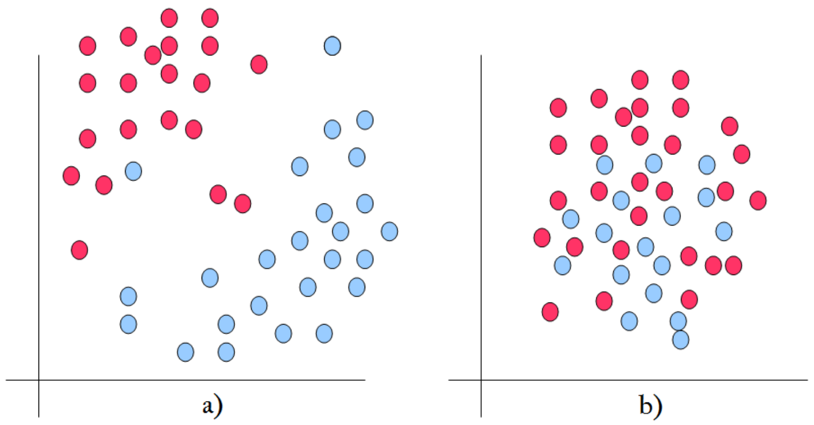

Figure 2.1: Two examples of features used to separate two classes in an arbitrary feature space. In a) the features allow for the separation of the data into two distinct classes. b) the features do not allow for the separation into different classes

of features in an arbitrary feature space. Unless it is stated otherwise, all examples use this generic feature space.

To allow for data to be classifiable, features have to be chosen that allow for a unique description of the data. Figure 2.1a shows an example in which the data points are readily separated and the two classes are obvious and distinct. In Figure 2.1b they are merged and there is considerable overlap, no obvious classifier exists for this data. Even in the case of Figure 2.1a, a mapping needs to be found that will correctly and consistently classify new data. The boundaries of three candidate functions are presented below in Figure 2.2.

Each of these classifiers has their strengths and weaknesses. While the first classifier example is linear in nature and relatively simple, it misclassified the most points. The second classifier is more complicated than the first, but correctly classifies points that the first classifier misses. The third classifier correctly classifies every point in the training set however does not appear to fit the distribution of points. To discuss

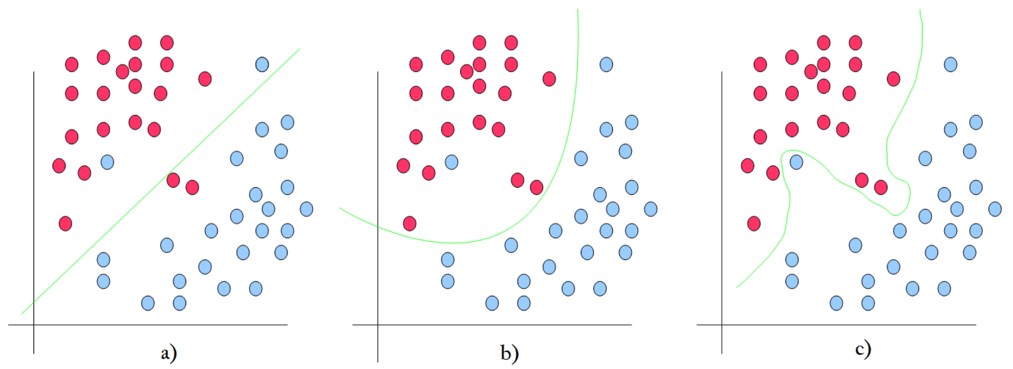

Figure 2.2: Three examples of potential classifiers for data in Figures 2.1a). a) a simple classifier which might not generalize well. b) a more complex classifier that appears to follow the data. c) a complex classifier that makes no mistakes but appears to over fit the data

how these effects can be quantified, a few concepts are introduced.

2.1.1 Risk

The risk of a classifier can intuitively be defined as the probability of a misclassification weighted by the cost or penalty associated with that mistake. Therefore some error can be tolerated in the classification if the penalty associated with that particular error is small, or conversely, points for which a mistake carries a heavy penalty are forced to be correctly classified to minimize the risk.

The risk of classifier F is simply the cost of assigning a label $F(X)$ to pattern X with true label Y, weighted by the probability of obtaining the combination (X,Y). The total risk is the sum of all such risk and can be calculated by integrating a cost function $C(X, Y, F(X))$ over the joint probability of X and Y (B. Scholkopf, 2002):

$$R(F) = \int C(X, Y, F(X)) dP(X, Y) \tag{2.1}$$

where the cost function is simply the cost of assigning pattern X a label $F(X)$ where the true label is Y. Many standard cost functions exist and the particular use of one depends upon the scenario in question. If all mistakes carry an equal weight and correctly classified points have no penalty, then the zero-one loss function can be used (B. Scholkopf, 2002),(Burges, 1998):

$$Z(X, Y, F(X)) = \begin{cases} 0 & F(X) = Y \\ 1 & F(X) \neq Y \end{cases} \tag{2.2}$$

The zero-one loss function is equal to 1 for incorrectly classified points and 0 for correctly classified points. Therefore all points have equal weight and only the fact that they are misclassified is important.

A common assumption in such a problem is the previous data is independently drawn from an identical underlying distribution $P(X, Y)$. If $P(X, Y)$ is known ahead of time or can be accurately approximated, then it is possible to classify the incoming data using Bayes rules. Assuming that the zero-one loss rule is used (every mistake is equally as costly and carries the same penalty), then the risk is minimized by assigning pattern X_{new} the label Y_i which maximizes the probability (B. Scholkopf, 2002) :

$$G(X) = Y_i | \max P(Y_i | X_{new}) \tag{2.3}$$

where :

$$P(Y_i | X_{new}) = \frac{P(Y_i) P(X_{new} | Y_i)}{P(X_{new})} \tag{2.4}$$

However the underlying distributions are rarely known a priori. The goal is then to find a function $F(X)$ such that $F : X \mapsto Y$. Ideally the desired $F(X)$ will correctly classify all future data. In practice, this is generally impossible due to noise or overlap of the underlying distribution, and limitations on a function's ability to classify data. The best that can be accomplished is to find $F(X)$ that minimizes the risk of classification.

It is the realm of machine learning to take previous examples of inputs and labels (X,Y) and use this previous data to develop a method to classify new inputs with a label Y_{new} that minimizes the total risk without knowing the underlying distributions. Several algorithms exist in order to either estimate the distribution, or minimize the risk through training data. Since the probabilities are unknown, a bound on the risk is all that can be found.

Re-examining the example presented earlier, a classifier can be trained with the given data. This data then becomes the training set in supervised learning. For the training set, all the labels are correctly known ahead of time for their given inputs. Presented in Figure 2.3 are two examples of classifiers each with a different linear decision boundary.

The two classifiers presented above classify the data differently and it is clear that the second classifier correctly classifies more data. Intuitively this would be the better classier, all else being equal. The risk associated with the training set is referred to as the Empirical Risk (Burges, 1998),(B. Scholkopf, 2002). This is defined as:

$$R_{emp}(F) = \frac{1}{n}\sum_{i=0}^{n} C(X_i, Y_i, F(X_i)) \tag{2.5}$$

which is the average value of the cost function on a particular set of training data, given a classifier $F(X)$. The Empirical Risk is a measure of how well the classifier performs on the training data. It is an approximation of the test risk, when the

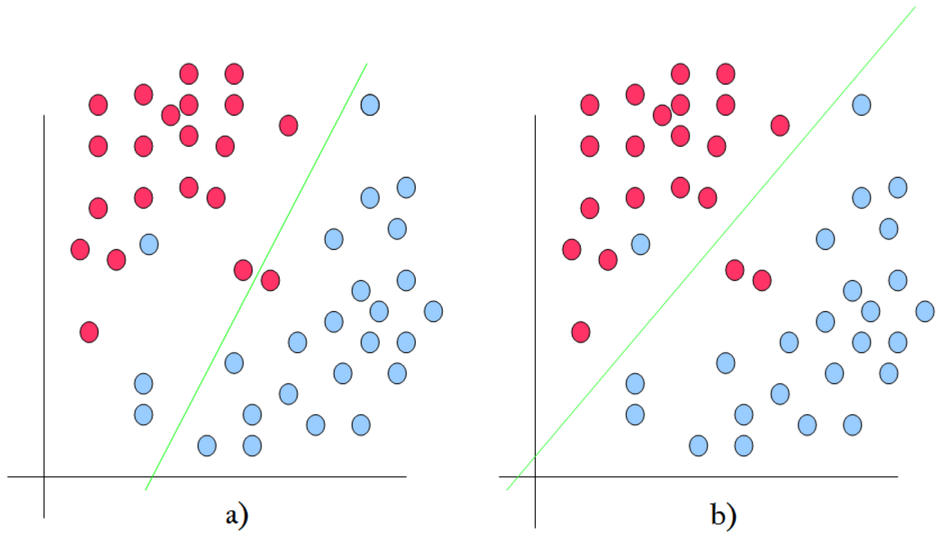

Figure 2.3: Two examples of linear classifiers for data presented in Figures 2.1a). b) correctly classifies more points than a)

probability is estimated from the training set. One way to attempt to minimize the total error is to minimize the test error.

However the empirical error is only an approximation and there are other effects that need to be considered when dealing with classifiers of different complexity. It is easy to construct an example such that given a training set (X_i, Y_i) and a second set of data (X_k, Y_k) where a classifier predicts exactly $F(X_i) = Y_i \quad \forall i$ and minimizes the empirical error, but $F(X_k) \neq Y_k \quad \forall k$. An example of this is shown in the regression in Figure 2.4. In one case the training data is fitted with a linear relation, and in a second case with a sinusoidal function. Due to the distribution of points both regressions fit the data exactly, however for no other values do they agree. Thus they have successfully fit the data, but one is clearly wrong for all other values.

In statistical learning theory, a bound on the total risk can be found which is the sum of two terms(Burges, 1998). As discussed earlier the first term is the risk

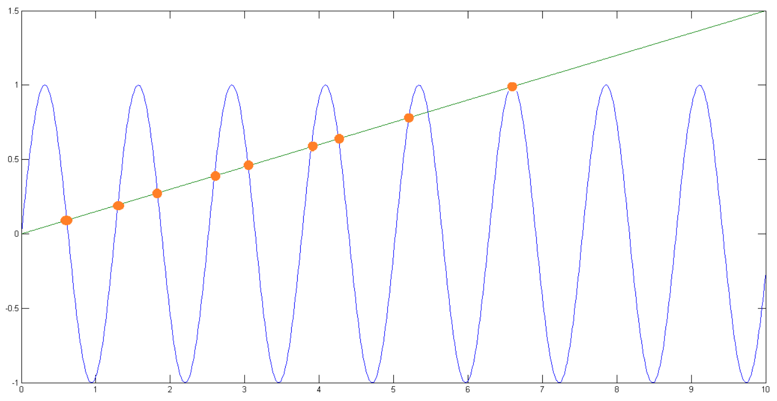

Figure 2.4: Underlying structure can influence the data. Two regressions are shown for the data in this feature space. Both regressions are exactly correct on the training data but both are clearly not right and predict very different future values

associated with the ability to classify the training set, namely the empirical risk (R_{emp}). Thus one way to reduce the total risk is by minimizing training error by a classifier of the same complexity. This makes intuitive sense as a function that has a small training error would probably continue to explain new data correctly, while one that cannot classify the training set would likely not generalize well to new data, all else being equal.

As shown above, this does not guarantee that a classifier approximates the underlying model. In fact, as shown in Figure 2.4, this can lead to disastrous results if this were to be the only criteria for selecting a classifier. When selecting classifiers, it is important to select one that has the right power or complexity that matches the underlying model. However the model is rarely known and thus a bound on the total risk is sought.

The bound for the risk is :

$$R(f) \leq R_{emp}(f) + \sqrt{\frac{h(log(\frac{2m}{h}+1) - log(\frac{\eta}{4})}{m}} \tag{2.6}$$

which holds with probability of at least $1-\eta$ (Burges, 1998),(B. Scholkopf, 2002) and $h < m$, where m is the dimension of the classifier f. The second term is a complexity term. It is proportional to h which is a non-negative integer that can be thought of in some respects as a measure of the complexity of the classifier. This h is the VapnikChervonenkis (VC)(B. Scholkopf, 2002) dimension of a classifier. It represents the complexity or power of a classifier to separate data. A high VC dimension allows a classifier to separate large amounts of data in a fairly complex arrangement but will have a large probability to overfit the data unless a classifier of such complexity accurately fits the underlying model. Classifiers that have a small VC dimension cannot divide complex data but will be less likely to overfit.

2.1.2 Shattering

A more formal definition requires the notion of shattering. A learning machine is said to shatter n points if for all possible sets of labels (X_i, Y_{ik}) all points can be correctly labeled for all sets k. An example is shown for a line in $\mathbf{R}^2$ with three points in Figure 2.5. All configurations of labels can be learned with 100% success rate by a linear classifier. However there are examples using four points, for which there is always one set of labels such that at least one point must be misclassified. The VC dimension for a machine is then just the maximum number of points a machine can shatter for a particular dimensional vector. For the case of a line in $\mathbf{R}^2$ the VC dimension is then three.

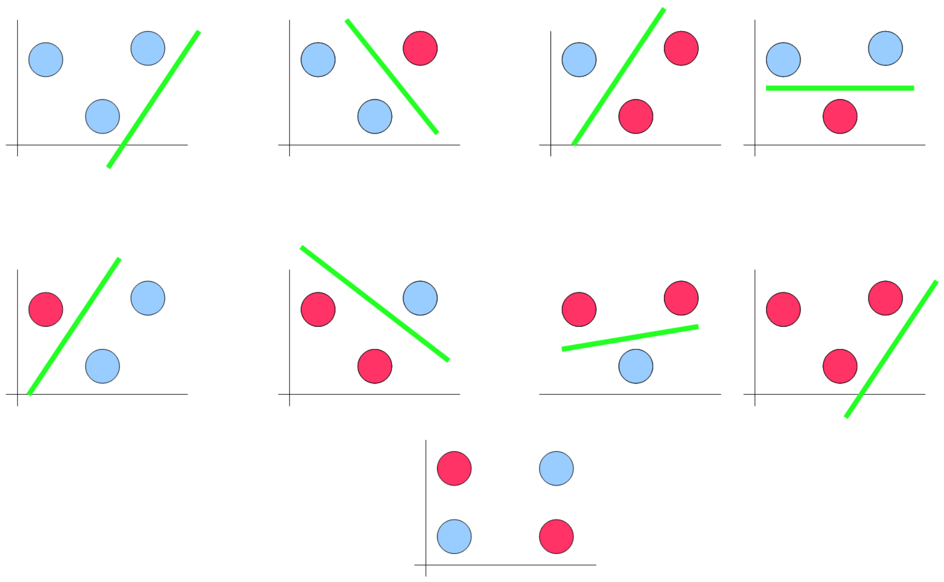

Figure 2.5: All 8 possible labels are shown for 3 points and how they can be classified with a linear classifier.

Therefore it makes sense to choose a classifier with a small VC dimension while still correctly classify the training set. Vapnik (Vapnik, 1998) has shown that for a linear classifier the VC dimension of the classifier decreases as the margin (distance to the closest points) of the classifier is increased if certain criteria are met (Burges, 1998).

2.2 Support Vector Machines

2.2.1 Hyperplanes

A hyper plane is an N dimensional generalization of a plane in $\mathbf{R}^3$ and a line in $\mathbf{R}^2$. It is defined by the characteristic of being linearly dependent in each dimension. That is to mean that every dimension appears linearly in its equation. The general equation for a hyperplane is:

$$a_1x_1 + a_2x_2...a_nx_n + c = 0 \tag{2.7}$$

which can be succinctly written using a normal vector w and the distance to the plane from the origin. The normal vector is a vector perpendicular to the hyperplane and defines a family of hyperplanes, while the distance uniquely defines the plane in that family:

$$w^T X + b = 0 \tag{2.8}$$

To classify using a hyperplane in a binary classification problem, the hyperplane is used to divide the space into two regions. Each of these regions corresponds to one of the two classes. Everything on one side is assigned a negative value and on the other, a positive value. The choice of sign is arbitrary and can easily be switched simply by multiplying through by -1. The classification is then simply to find which part of the space the feature vector is on. This is accomplished, for an unknown feature vector, by taking the dot product with the normal of the hyperplane and adding the bias. This gives the distance along the direction of the normal (the shortest path to the hyperplane). Then the sign of this result will tell which side of the space the input feature vector is on. Mathematically this can be represented as:

$$Y = sign(w^T X + b) \tag{2.9}$$

2.2.2 Maximum Margin Hyperplanes

The question still remains as to which hyperplane to use. Justifications have been stated for using hyperplanes and how to classify with the result but do not include an algorithm to find a hyperplane which will act as a classifier.

One way to do this is to find a hyperplane that maximizes the margin. The margin is the perpendicular distance from the closest feature point to the hyperplane in both directions. By maximizing the margin, small errors in the feature vectors' positions will not result in a large change in the hyperplanes location. Since the maximum margin is desired, this can be formulated as an optimization problem. In certain forms, optimization problems can be solved quickly and can be guaranteed to be a global maximum. It is assumed that the data is linearly separable for this derivation, and the derivation will be generalized to non-linearly separable cases later on in this chapter. To maximize the margin, the problem can be formalized mathematically as an optimization problem in terms of the normal vector w. As mentioned earlier, to classify a point, the dot product with w is taken and the bias added in order to determine the sign. If the value is positive it is one side, and negative if it is on the other:

$$w^T X + b \tag{2.10}$$

Two additional planes can be defined, which will be referred to as the positive and negative planes. These planes are parallel to the hyperplane and intersect points that are closest to the hyperplane in the dot product sense. For mathematical simplicity, the distance to these planes can be forced to be -1 and +1 for the positive and negative plane respectively. Any value greater than or equal to 1 could be used, but it is mathematically convenient to use -1 and 1. Taking one point on each plane (X^- on the negative plane and X^+ on the positive plane) that both lay on a line defined by the normal vector, w, and evaluating the classifier:

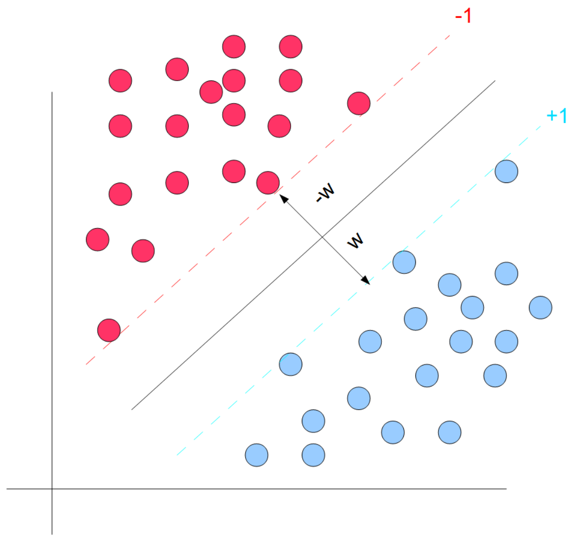

Figure 2.6: Two parallel plane next to the hyperplane

$$w^T X^- + b = -1 \tag{2.11}$$

$$w^T X^+ + b = +1 \tag{2.12}$$

The margin, M, is simply the distance between these two planes. This can be found by taking the difference between the two vectors and taking the vector norm. The margin can be found as:

$$|X^- - X^+| = M \tag{2.13}$$

since these points lie on the same line, and the vector defining this line is w, one point can be expressed as the first point plus a multiple of the w.

$$X^+ = X^- + \lambda w \tag{2.14}$$

where lambda is an unknown multiple. To find the margin in terms of w, Equation 2.12 can be combined with Equation 2.14 to give:

$$w^T(X^- + \lambda w) + b = 1 \tag{2.15}$$

$$w^T X^- + b + \lambda w^T w = 1 \tag{2.16}$$

but $w^T X^- + b = -1$:

$$-1 + \lambda w^T w = 1 \tag{2.17}$$

$$\lambda = \frac{2}{w^T w} \tag{2.18}$$

The margin can be written in terms of w by using Equation 2.13:

$$M = |X^+ - X^-| = |\lambda w| = \frac{2|w|}{w^T w} = \frac{2}{\sqrt{w^T w}} \tag{2.19}$$

This allows for the margin to be expressed in terms of w. Since margin is inversely proportional to w, it can be maximized by minimizing w while requiring all points to be labeled correctly. This can be expressed mathematically as:
minimize:

$$\frac{1}{2} w^T w \tag{2.20}$$

subject to:

$$y_k.(w^T X_k - b) \geq 1 \quad \forall \;\; k \tag{2.21}$$

using Lagrange multipliers, α_i this can be written in the primal form as:

$$L_p = \frac{1}{2}||w||^2 - \sum \alpha_i y_i (w^T X_i - b) + \sum \alpha_i \tag{2.22}$$

2.2.3 KKT and Optimization

The Karush Kuhn Tucker conditions is a set of conditions in non-linear programming that are necessary for a point to be an maxima point (B. Scholkopf, 2002). In convex optimization they are not only necessary but sufficient for a point to be a maximum. The maximum-margin hyperplane problem is a convex problem, and there is guaranteed to be one minimum at which the KKT conditions will be satisfied. The KKT conditions for the maximum margin hyperplane derivation above can be stated as (B. Scholkopf, 2002):

$$\frac{\partial}{\partial w}L_p = w - \sum \alpha_i y_i X_i = 0 \tag{2.23}$$

$$\frac{\partial}{\partial b}L_p = \sum \alpha_i y_i = 0 \tag{2.24}$$

$$y_i.(w^T X_i - b) - 1 \geq 0 \ \ \forall i \tag{2.25}$$

$$\alpha_i > 0 \ \ \forall i \tag{2.26}$$

$$\alpha_i(y_i.(w^T X_i - b) - 1) = 0 \ \ \forall i \tag{2.27}$$

Equation 2.25 is a condition that every point must be correctly classified, and falls either outside or on the positive or negative hyperplane. Equation 2.26 is required; otherwise the minimization could be accomplished by sending the values to negative infinity.

The last condition (Equation 2.27) requires that either the corresponding multiplier be equal to 0 or else $y_k.(w^T X_k - b) - 1 = 0$. However if $y_k.(w^T X_k - b) - 1 = 0$, then X_k must lie on the either the positive or negative hyperplanes. Thus only points that lie on the margin will have non-zero Lagrange multipliers. The solution to this

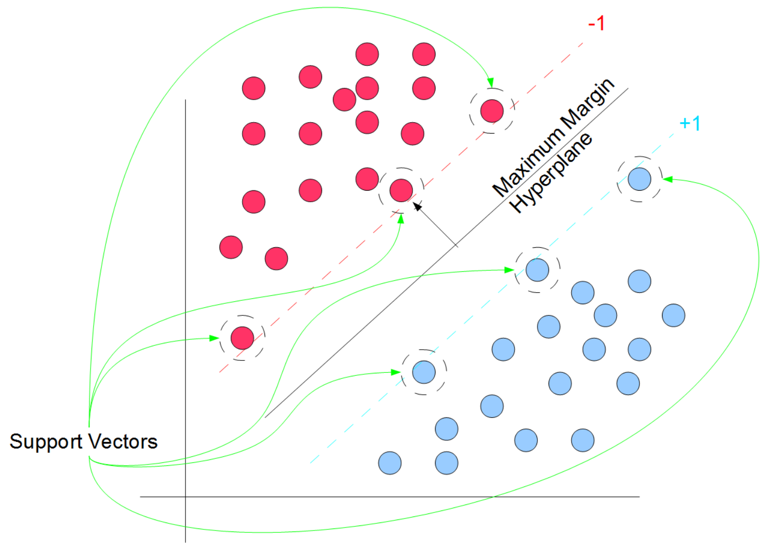

Figure 2.7: Final hyperplane and the support vectors

problem has the form:

$$w = \sum_{i=0}^{n} \alpha_i Y_i X_i \tag{2.28}$$

However, this sum only needs to be over the non-zero terms. Only the vectors that lie on the margin will be included in the sum. These vectors are referred to as the support vectors and they uniquely define the hyperplane.

Once w is determined the bias, b, can be determined from the support vectors using the fact that $y_k.(w^T X_k - b) - 1 = 0$. In practice, this is usually taken either as an average over all support vectors or determined using the support vector with the largest α_i (Burges, 1998).

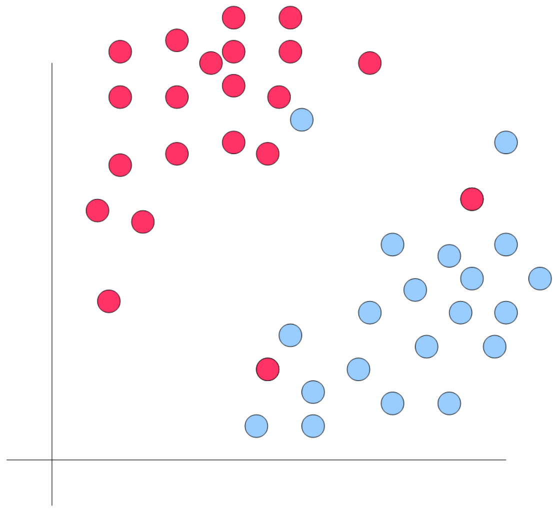

Figure 2.8: This data is non-linearly separable due to the overlap of the two sets of points. This data set will cause the previous derivation to fail since no solution can satisfy all constraints

2.2.4 Non-Linearly Separable Data

This derivation only works due to the assumption that the data is linearly separable. It is easy to see if the training data would be similar to Figure 2.8, then there will be no w such that $y_k(w^T X_k - b) - 1 \geq 0$ is true for every point.

The assumption of being linearly independent is not always a practical assumption due to several factors. Noise in the measurements can result in the feature points not being in their true location, which can cause the data to become linearly inseparable. In addition, the underlying distribution may have a highly non-linear boundary, resulting in features that are not linearly separable by construction.

To deal with features that are not linearly separable, a similar derivation can be constructed as for the linearly separable case. The main difference is that a cost function, $C(X,Y,F(X))$, for each error needs to be introduced and added to the function to be minimized. By adjusting the cost function, the weight of errors can be adjusted.

If the zero-one loss function is used, then the problem can be formulated as:

$$\frac{1}{2}w^T w + C\sum Z(X_k, X_k, F(X_k)) \tag{2.29}$$

where $F(X_k)$ is $\text{sign}(w^T X_k + b)$ and C is the cost associated with misclassifying a point.

The problem with this formulation is that it cannot be expressed as a quadratic programming problem and all errors are handled the same, regardless of the amount of misclassification.

To combat this issue, the hinge loss function can be used instead. The hinge loss function is defined as 0 for correctly classified points and increases linearly for misclassified data, proportional to the degree misclassified.

$$H(X,Y,F(X)) = \begin{cases} 0 & F(X)Y > 1 \\ 1 - YF(X) & \text{otherwise} \end{cases} \tag{2.30}$$

where $F(X)$ is $w^T X + b$ and real valued. By adding slack variables ζ_k which are equal to $H(X,Y,F(X))$ (as shown in Figure 2.9), the goal now is to minimize the function (B. Scholkopf, 2002):

$$\frac{1}{2}w^T w + C\sum \zeta_k \tag{2.31}$$

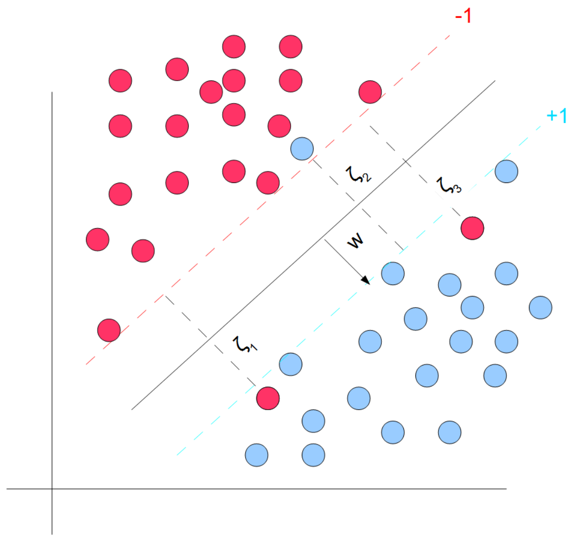

Figure 2.9: Using a slack variable, the requirement can be eased such that every point must be classified correctly. It now becomes a problem of minimizing the normal vector,w, and the slack variables

subject to the constraints:

$$y_k.(w^T x_k - b) \geq 1 - \zeta_k \tag{2.32}$$

$$\zeta_k \geq 0 \tag{2.33}$$

The first constraint is similar to the constraint in the original construction of the problem, except the error term is subtracted from the constant. Thus it is proportional to the amount of misclassification of that particular feature vector. Since these terms appear in the minimization problem, the ideal value should be 0, unless they are required because the points are misclassified.

The second set of constraints is required, otherwise the ζ_k can be sent to negative infinity to minimize the problem. This changes the final condition of the KKT criteria to be:

$$0 < \alpha_i < C \tag{2.34}$$

This helps the problem by limiting the effect of any one feature point on the position of the hyperplane so that mistakes can be accounted for. This new formulation requires an additional N constraints where N is the number of feature vectors in the training set.

2.2.5 Kernel Functions

Despite the ability to deal with data that has some overlap, there are still examples where these derivations cannot be applied, or their results are meaningless. These are cases where the data, by construction, has a boundary that is highly non-linear. A simple 1D example of this is shown in Figure 2.10 where one group of features is surrounded by another in a non-linear fashion.

This can be dealt with by allowing non-linear basis functions in the construction of the optimization problem. This is equivalent to projecting the data into a new, higher dimensional feature space where the goal is that the data will become linearly separable in the new space. By applying a quadratic mapping to the points of Figure 2.10 the data becomes linearly separable as shown in Figure 2.11. The new dimension is simply the square of the distance from the origin. It is important to note that no additional information is created; the new dimensions are determined purely based on the existing data.

Thus by applying a mapping $\Phi(X)$ which maps X onto an M>N dimensional feature space, non-linearly separable data now becomes separable. The only constraint on $\Phi(X)$ is that the dot product in this space must exist. Thus the formulation is

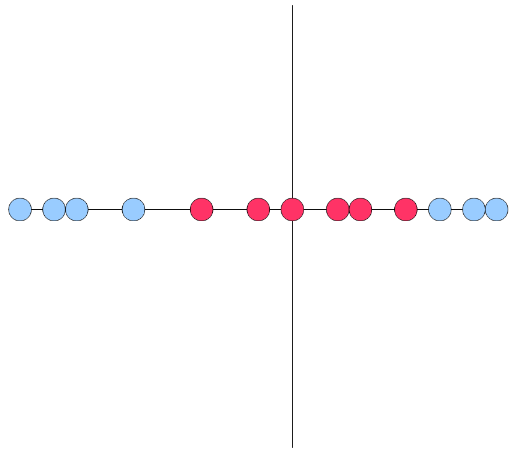

Figure 2.10: Data can be arranged in 1D to be non-linearly separable. There is no meaningful linear classifier for this data

identical, except every vector is now mapped onto this space. The set of constraints $y_k.(w^T X_k - b) - 1 \geq 0$ become $y_k.(w^T \Phi(x_k) - b) - 1 \geq 0$ instead and the solution takes the form:

$$w = \sum \alpha_i Y_i \Phi(X_i) \tag{2.35}$$

To classify a new pattern X_{new}, the equation becomes:

$$Y_{new} = sign(\sum \alpha_i Y_i \Phi(X_i)^T \Phi(X_{new}) + b) \tag{2.36}$$

Although this approach has allowed for non-linear data to be separated, it has come at a great cost. The new dot products are M dimensional which requires M operations. In practice the number of dimensions M will be much greater than the the original dimension of the vector, N. This can be demonstrated for a simple polynomial

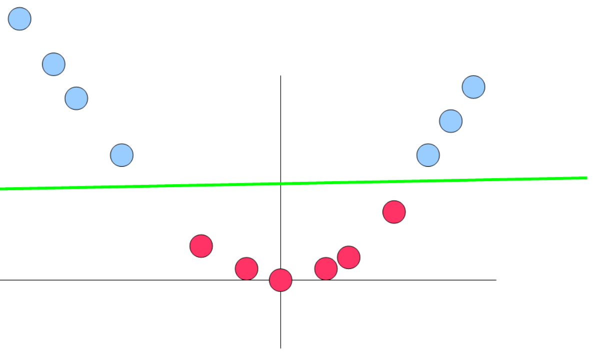

Figure 2.11: By allowing non-linear basis function the previous data now become linearly separable. No new data was added; the data is merely projected into a new higher dimensional space

mapping.

For an N dimensional vector X, let the mapping $\Phi(X)$ be defined as:

$$\Phi(X) = [1, x_1, x_2, x_3 ... x_n, \sqrt(2) x_1 \dot{x}_2 .... {x_1}^2, {x_2}^2 ... {x_n}^2] \tag{2.37}$$

For this simple quadratic mapping, there are still N linear and square terms but now N^2 cross terms of the form $\sqrt(2) x_i x_k$. The computation of the dot product in this space is now $O(N^2)$ instead of $O(N)$ previously. This makes directly calculating the mappings time consuming and quickly becomes infeasible for increasingly larger vectors.

The goal is to find kernel functions $K(X_1, X_2) = \Phi(X_1)^T \Phi(X_2)$. In this way no mapping is explicitly done, but is instead implicitly done in the kernel function.

For the example presented above with the polynomial basis function, an appropriate function would be $K(X_1, X_2) = (X_1^T X_2 + 1)^2$. The main difference between these two functions is the number of operations, but the results are equivalent. With the explicit mapping, it takes $O(N^2)$ to perform the dot product, while with the kernel function the computational complexity is still only $O(N)$. The kernel function can also map vectors into much larger dimensional space, or even infinite dimensional spaces. Some common kernel functions are presented below in Table 2.1:

Polynomial	$K(X_1, X_2) = (X_1^T X_2 + 1)^k$
Radial Basis Function	$K(X_1, X_2) = e^{-\gamma \lVert X_1 - X_2 \rVert}$
Hyperbolic Tangent	$K(X_1, X_2) = tanh(\kappa X_1^T X_2 + C)$

Table 2.1: Common Kernel Functions

2.2.6 Support Vector Machines and Computer Vision

Support vector machines and classification pattern recognition algorithms in general are best suited when a descriptive set of features is known but the variation in the features is only vaguely understood or difficult to express numerically. Instead of relying on an algebraic thresholding for features, SVM relies on past data to classify new data. Another key for the successful use of classification algorithms is the abundance of consistent past data which clearly illustrates the variation within a feature set.

This is the case with many computer vision problems, where there are several examples of previous data but where exact numeric thresholds are difficult to establish. One classic example of this is optical character recognition (OCR) (Chanda *et al.*,

2007) (Chen *et al.*, 2001). Where it is difficult to numerically describe all the variations of what a nine may look like when handwritten, it is instead much easier to produce many examples to train a classifier on.

This is well suited for the problem of classifying PET and Polycoat containers. Each container type has a characteristic set of pixels that make up the image. Images of PET containers are made up of predominantly black pixels, while Polycoat containers show a large degree of variation in pixel intensities. However it is difficult to quantify exactly what this means consistently. PET bottles can have labels which will increase the variation of pixel intensities, and Polycoat containers can have regions with black or with bright reflections which make thresholding difficult.

However examples images are relatively easy to obtain, and can be used to train a classifier. These factors make this problem ideally suited for a machine learning algorithm. A feature is known that can distinguish the two container types, which is difficult to express numerically, but where many past examples are available.

2.3 Field Programable Gate Arrays (FPGA)

Field programmable gate arrays (FPGA) are essentially a reconfigurable integrated circuit that allow for the construction of custom digital circuits by programming the interconnection of smaller logic blocks. Starting off as mere glue logic chips, FPGAs are now used as standalone embedded systems often for real-time performance and even replacing ASICs (Application Specific Integrated Circuits) in areas with lower volume sales. The main feature of an FPGA over ASIC is the reprogramability that allows updating with firmware.

Figure 2.12 shows the simplified architecture of an FPGA. The basic element of

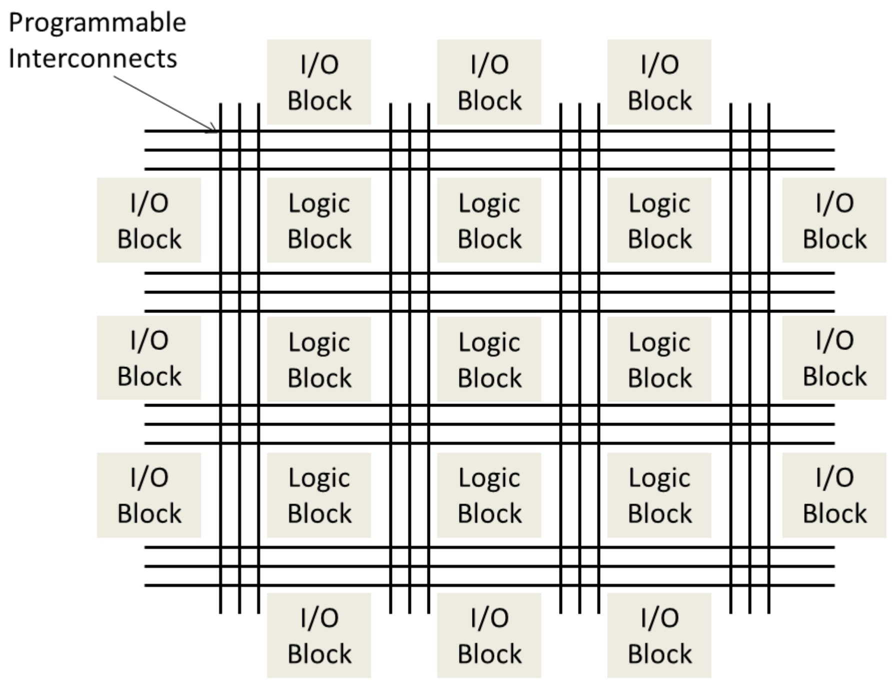

Figure 2.12: Typical FGPA architectue with logic blocks, I/O, and interconnects

an FPGA is the logic element (LE). These are the building blocks of all the logic circuitry. In general, a logic block consists of a few logical elements. The logic blocks are connected by programmable interconnects, which connect the logic blocks together in such a way to form the desired output. These outputs then can be routed to various I/O blocks to communicate to the external world.

In most modern FPGAs, logic elements contain one or more LUTs (Look Up Table) that can be programmed to implement any logic function provided the LUT has enough inputs. Figure 2.13 shows how a typical LE looks in Altera Arria II FPGA. These LEs have eight inputs to two separate four-input LUTs, although a variety of configurations are possible. The outputs of the logic element can be programmed to be either buffered in a flip-flop, to provide sequential logic circuits, or to be taken

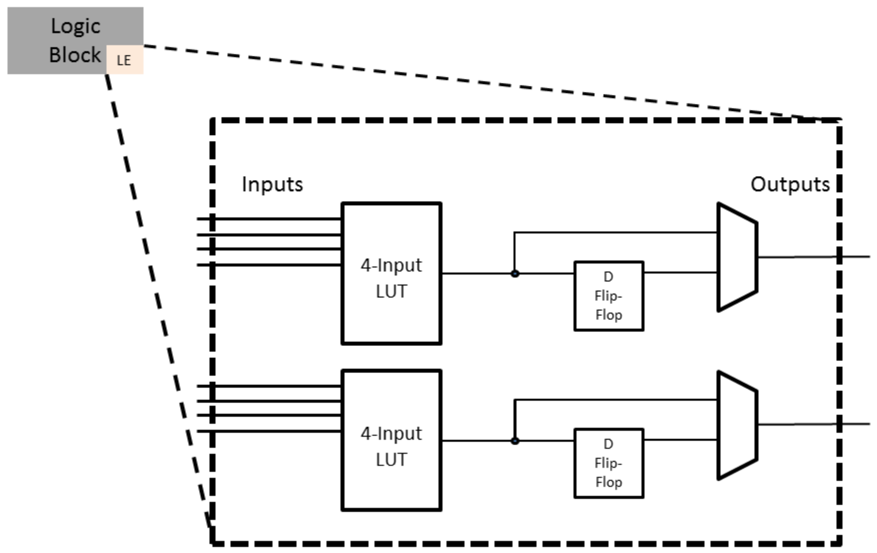

Figure 2.13: Typical LE configuration in an Altera Arria II GX device

straight out of the LUT for combinational circuits.

To create complex logic circuits, the LEs are connected via programmable interconnects. The contents of the LUTs in the LE and the connections between LEs are controlled by SRAM cells which are programmed by downloading a bit stream into the device. The bit stream is generated by compiling an HDL (Hardware Description Language) description of the desired behavior. The compiler will attempt to minimize area and maximize speed as much as possible.

In addition to LEs and their connections, modern FPGAs have many additional features. Often a small subset of the total LEs can be programed to provide additional functionality such as predefined arithmetic circuits or additional memory blocks. Prefabricated dedicated hardware blocks can provide often used functionality for less space and faster clock speeds then can be made by connecting LEs. Typically 100s - 1000s of hardwired multipliers are provided to perform quick multiplications and are implemented with commonly used DSP functionality. Large prefabricated multi-port memory blocks that can be used for fast on-chip storage that can in the 100s of Kb to 10s of Mb range.

By allowing for custom circuits to be implemented in the FPGA fabric, many types of computations can be sped up by incorporating large degrees of parallelization. Common strategies for speeding up algorithms can take advantage of the nature of the computations required, and implement several concurrent calculations to happen at once. Often algorithms can be implanted in stages which are independent from each other further increasing the number of computations performed in a given time period.

The overall computational ability of an FPGA can be measured in number of hardware blocks consumed. It is desirable to consume the least possible amount of area while still meeting throughput requirements. In general, the speed of a block and its size are inversely related. It is possible to create fast blocks that make many computations in parallel but take up a large area, while creating a circuit that performs the exact same calculations in more clock cycles will in general be smaller. Thus there is a design trade-off between speed and area.

A main drawback of using such devices is the limited ability to use floating point arithmetic. New devices are showing more custom blocks to deal with this disadvantage but floating point arithmetic is still much slower than the fixed point computations within FPGAs. This means designers are often forced to use fixed point computations to be able to obtain the highest speed up. Often this limitation can be overcome by the use of appropriate scaling; however in applications requiring a large dynamic range, the lack of floating point compatibility can be a significant disadvantage.

2.3.1 FPGAs in Computer Vision

The use of FPGAs in computer vision is well documented. (MacLean, 2005) studied the effectiveness of using FPGAs in embedded vision systems, and concluded that they were a promising technology. It was also noted that the main problem were the fixed point algorithms, design tools, and device sizes. All of these improve with time, making the case for FPGAs stronger as time goes on.

There are several common traits of computer vision problems that lend themselves well to parallel computations and FPGAs specifically. The first is the large amount of data that needs to be processed when dealing with images. Secondly, many computer vision algorithms tend to be divided into stages, where low-level data is translated in to a higher level, such as taking raw pixel data and translating it into facial data. Finally, there is often a need for real time processing in many systems.

Practically all computer vision tasks start with the examining of an image or a set of images. An image represents a large amount of data to process, usually hundreds of thousands to millions of pixels. Each pixel or neighborhood of pixels is often subjected to the same set of operation which can be done in parallel with each other. This is especially true for so called low-level tasks such as edge detection, filtering, and thresholding. This is ideally suited for hardware implementation and can greatly simplify logic circuits, due to the lack of complicated control logic as the same operations are repeated.

Many algorithms, as described above, can be easily pipelined. Often low-level computer vision tasks are used as preprocessing for higher level ones. The higher level tasks can operate on the output from lower levels as it becomes available allowing for the two stages to operate independently which allows for a significant speed up in computation time. Additional levels of parallelism can often be found within the various stages of an algorithm.

In (Goshorn *et al.*, 2010) an FPGA implementation of a parts-based object recognition system is presented that demonstrates many of the features mentioned above. For low level extraction of candidate parts they first start with a simple Harris-corner detector. From the corners, windows are extracted and are compared against reference windows for each part. Once all the parts are identified each part votes to where the object is located. This shows the heavily pipelined nature of the algorithm. In addition many stages of the algorithm exhibit large degrees of parallelism such as computing the SAD (Sum of Absolute Difference) of each window with the reference part. By doing this, a 266 FPS realization of their algorithm was achieved.

Lastly many applications require that data be processed in real-time such as in navigation, visual-servoing, and many industrial inspection techniques. By using the parallel nature of FPGA real-time performance can be obtained in a relatively small package often with a small energy foot print.

Sorting of Polycoat and PET containers using visible light cameras will have many of these properties. As a system there are many vision functions that need to be performed. The containers need to be extracted from the image and then classified using an SVM. SVM themselves have many aspects that lend themselves nicely for parallel computation. Since the goal is to sort as fast as possible, an FPGA implementation is a natural choice.

2.4 Chapter Summary

Chapter 2 introduced the concepts of pattern recognition with the specific example of an SVM and described in detail what an FPGA is and how they are used for fast computation. It was shown how machine learning algorithms like support vector machines can be used in computer vision tasks, and why they are suitable for the problems dealt with in this thesis. A brief introduction to FPGA and their architecture was also presented. Examples of previous computer vision work that have taken advantage of FPGAs were presented, and the current sorting task is naturally adapted to this style of implementation.

Chapter 3 describes the exact details of the algorithm and the hardware implementation. The algorithm is broken down into several sections, such as background removal and classifications, and each subsection is explained. The implementation of the algorithm in hardware is then presented. Each of the major blocks and their functionality are demonstrated.

Chapter 3

Algorithm Overview and Hardware Implementation

3.1 Introduction

This chapter introduces the inner workings of the algorithm. Each major section of the algorithm is explained in detail as well the general intuition behind the approach. The FPGA implementation is also described. Descriptions are given of how the major algorithmic functions are translated into hardware and the low-level interactions between hardware blocks. Parts of this chapter were published in the IEEE International Symposium on Sustainable Systems and Technology (ISSST) (House *et al.*, 2011).

The goal of the algorithm is to be able to correctly and consistently locate and classify containers in an unknown scene. The containers belong to one of two classes: PET bottles or Polycoat containers. The PET bottles are predominately transparent while the Polycoat are opaque. By illuminating the scene with a high power light, the reflective properties of the materials can be observed by the presence or absence of specular reflections. Once the containers have been isolated from the background, they are classified as a Polycoat or a PET container. In this chapter an overview of the developed algorithm and its implementation in an FPGA are demonstrated.

The proposed method for segmentation of the containers starts with the division of the image into many non-overlapping equally sized blocks. A histogram of pixel intensities is constructed from the pixels contained within each block. These histograms are used to differentiate between blocks that contain background pixels and those that contain container pixels. The non-background blocks are grouped together based on similarity to form regions with similar histograms. The histogram over the entire region is used to classify the region as either a PET bottle or a Polycoat container.

An FPGA is used to accelerate the algorithm to allow maximum throughput of image data and real time performance. There are many aspects of the algorithm that can be implemented in such a way as to exploit the parallel nature of computations on an FPGA. The first is the ability to break the algorithm in an efficient pipeline structure. Each stage operates independently and in parallel of the previous block. By stretching out the pipeline many calculations can be performed at once. In addition many independent calculations can be trivially performed in parallel to increase speed.

3.2 Experimental Setup

The experimental setup consists of a high speed GiG-E Vision camera that images the containers against a black cloth background as shown in Figure 3.1. The black background is used to aid in the segmentation process and transparency measurements. A 300W work light provides intense intense illumination to create many specular

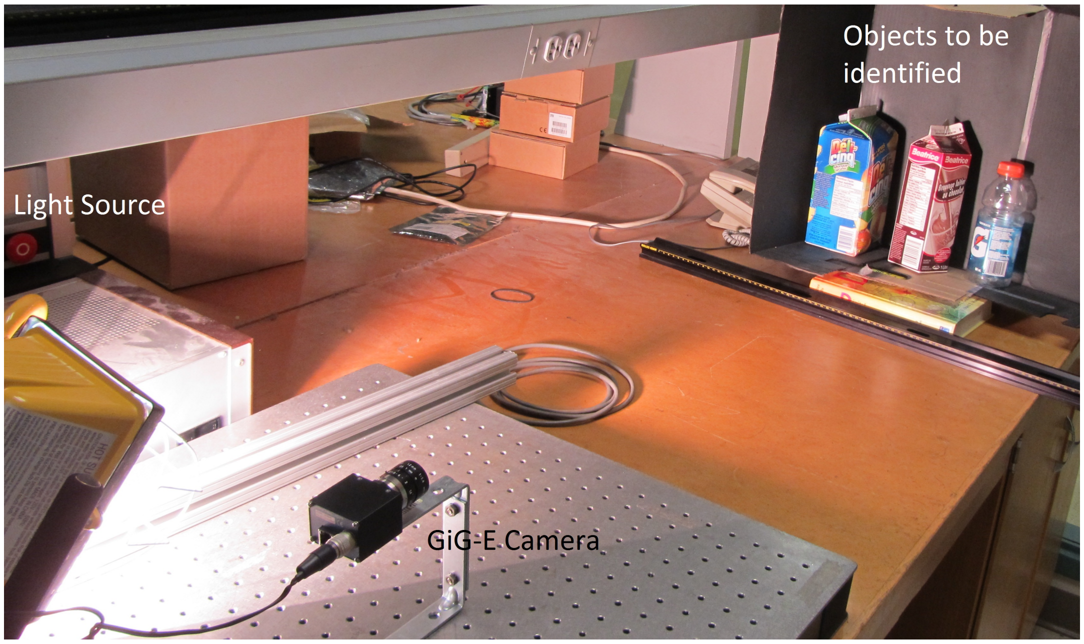

Figure 3.1: Experimental Setup (PC not shown)

reflections. The amount of specular reflections, and correspondingly the amount of white pixels, is an indicator of the reflectivity of the package. The images acquired are sent to the PC using Gigabit Ethernet. This allows for a frame rate of up to 120FPS for a 640×480 image. The image data is streamed to an FPGA daughter card for processing via the PCI-E bus. The FPGA daughter card has an Altera Arria II GX device where the data is processed and the results are returned to the PC via the PCI-E bus.

3.3 Algorithm

An overview of the proposed algorithm is shown in Figure 3.2. There are four main sections to the algorithm: Histogram Construction, background removal, Region Growing, and Region Classification. Histogram construction divides the image into small blocks and the histogram of each block is constructed. Blocks are determined to be part of the background by using a linear SVM, trained on background histograms. Region growing joins together non-background blocks based on the similarity of their histograms to form regions that are likely one type of container. Finally each region is then classified based on it histograms and assigned to one of the two container types.

The proposed method of classifying bottles uses the histogram of pixel intensities of black and white images. Histograms provide many ideal properties that make them well suited to the task of classifying different containers. Since the recyclables can arrive in any condition at an MRF, a desirable feature of the histogram is the robustness against transformation in scale, size, rotation and deformation. Even under extreme transformation, deformations and small amounts of occlusions the histogram of an object is very repeatable.

Histograms provide a large dimensionality feature vector that can be scaled for a balance between computational cost and accuracy. For an 8 bit greyscale image, the histogram has 256 (2^8) dimensions. By reducing the number of dimensions, a reduction in the overall computation effort can be achieved, at the expense of accuracy. By only using the first most significant n bits, the dimensionality is reduced to 2^n.

Histograms are useful because it allows for an indirect method of measuring the optical properties of the two materials. Since the PET plastics are in general much more reflective and provide more reflective surfaces, specular reflection shows up as bright white areas. However PET is also transparent, and by using a black background, most of the pixels of a PET bottle are imaged as black. Polycoat containers on the other hand are in general opaque and end up with a histogram that is more spread out in the central regions. Two histograms are presented in Figures 3.3 and

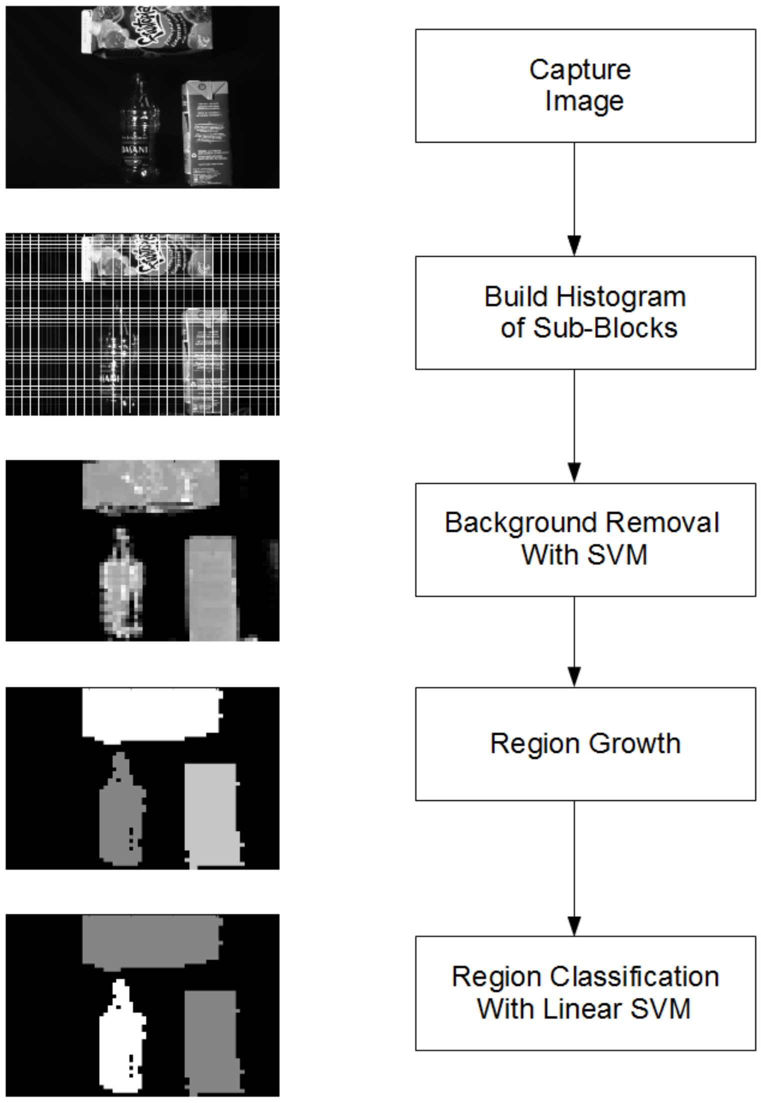
Capture Image
Build Histogram of Sub-Blocks
Background Removal With SVM
Region Growth
Region Classification With Linear SVM

Figure 3.2: Overview of Proposed Algorithm

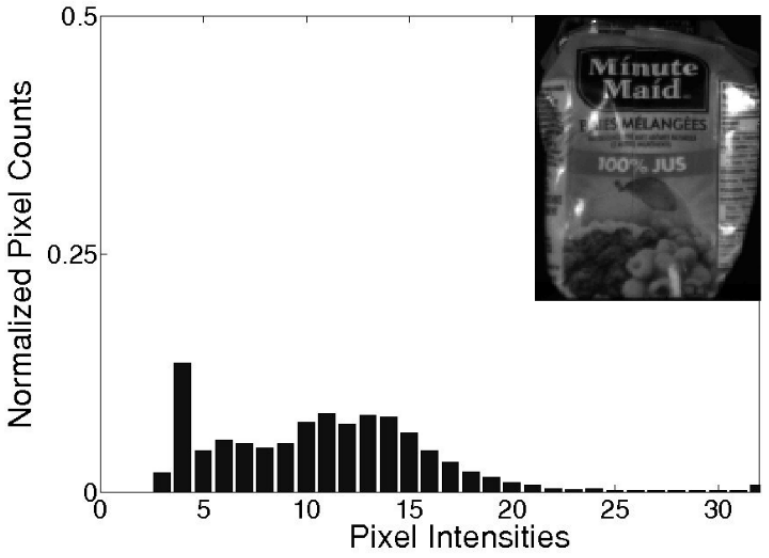

Figure 3.3: Typical Polycoat container and histogram

3.4:

The PET histogram has a large spike in the black regions, which is the background showing through the transparent PET bottle. There is also a much smaller peak at high intensities which correspondent to the specular reflections, and very few pixel counts in the central regions. The Polycoat container histogram has a much more spread out histogram in the central pixels with a much smaller peak in the black due to some background pixels being included in the histogram at the boundary.

Although these properties hold globally over the two container types, there can be much variation locally. Polycoat containers can exhibit specular reflections, which contributes to a similar peak to PET bottles in the high pixel ranges. PET bottles often have labels attached to them, which have very similar optical properties to Polycoat containers.

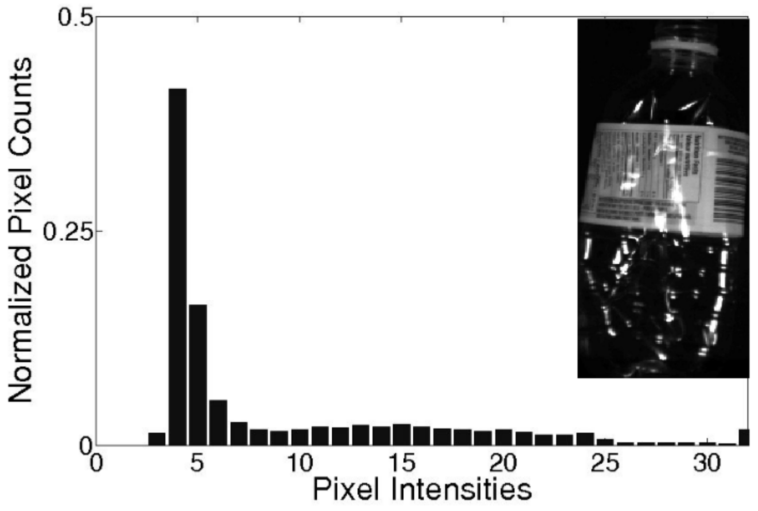

Figure 3.4: Typical PET bottle and histogram

3.3.1 Block Construction

Although globally the histogram details are well defined, locally it can be difficult to determine which pixels belong to the background, and can be subsequently ignored, and which are part of an object to be classified. For a Polycoat bottle the difference is trivial and a simple threshold could be set. However for the PET containers this is not the case. PET bottles are transparent and allow the black background to show through. The transparent sections have similar intensity values as compared with the background, meaning that a simple threshold will likely cause regions of PET to be incorrectly labeled as part of the background. Figure 3.5 shows a close up on two areas: one of a PET bottle, and the other is part of the background.

By examining Figure 3.5 it is apparent that the only indicator that this particular region belongs to a PET bottle, as oppose to the background, is the presence of near-by specular reflections. Thus the proposed solution is to sub-divide the image into small N×N blocks and using the intensities contained within each block, to make a

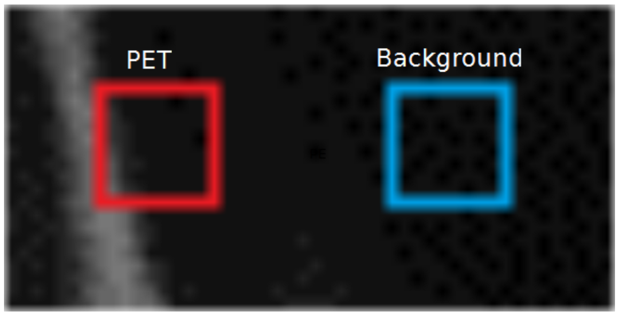

Figure 3.5: Enlarged Boundary Region of a PET Bottle. The red box contains similar pixels to the blue box, the only difference is the presence of the specular reflection

decision as to whether or not it belongs to an object or the background.

The goal is to include regions that have specular reflections, with black pixels. Blocks that are purely part of the background will have histograms which contain mostly black regions, while those that have histograms with a significant peak at high intensities will likely not be part of the background. These blocks will then contain both black pixels and white pixels for the PET, while Polycoat containers have almost no fully black pixels and are readily distinguishable from the background.

In addition to be being a way of aiding in removing the background, the block construction acts as a method of data reduction. By only storing the histograms of each block a significant savings on memory requirements can be obtained. For an N×N block, there are a maximum of N^2 pixels that belong to one bin of the histogram. Therefore there are at most needed $log_2(N^2)$ bits allocated for each bin. If there are n_{bits} per pixel then there are a total of $2^{n_{bits}}$ bins. The total memory requirement to represent a histogram is $2log_2(N^2)2^{n_{bits}}$. This compared to the original requirement of $N^2 n_{bit}$, a significant savings for even moderate N by reducing the number of bits used to construct the bins.

In essence the reduction in data is due to the loss of spatial information. In a histogram only the pixel intensities are known, and not their distribution in space. This is unimportant in the overall algorithm because the end result is to construct a histogram over only a single bottle. Although the spatial distribution within a block is discarded, the only fact that is relevant is which container is each block a part of. This is addressed further on, the first step of which is to remove the background.

3.3.2 Background Removal

Background removal is accomplished by determining if a block is likely to be part of the background or belong to a container. A block is classified as part of the background based on the histogram. To remove the background, each block needs to be classified as either a background block or a container block. Each individual block is classified using a support vector machine trained against the background and PET bottles histograms. Polycoat were not required because they are readily distinguishable from the background, and their histograms were found to put them far into the container sections whether they were included in the training set or not.

Using the histogram blocks constructed in the last section, each block is subject to classification by using the SVM. The SVM used in background removal is trained against blocks obtained from a background only image and a set of PET bottle images. The background only image is an image with the same the lighting conditions, but with no containers present. The same block construction method is applied to the background only image to obtain histogram vectors of containerless blocks.

A sample of PET bottles was then chosen and the PET bottles are extracted using a simply bounding box procedure. A Canny (Canny, 1986) mask is applied to each image and the edge pixels at the extremes are identified. A bounding box is then applied and only pixels within that region are formed into the histogram vectors to train the SVM.

Only PET bottles are used to form vectors of non-background examples, not Polycoat containers. This is because Polycoat containers very rarely have dark pixels in them, and even then they are almost never the same intensity level as the background. Using only the PET saves on possible support vectors and simplifies the calculations.

The output value of the SVM goes through one more filtering stage in order to fill in holes caused by blocks that do not have many specular reflections in them, but are close to blocks that do. An averaging 3x3 smoothing filter is applied to the output, which averages the output of the SVM with its 8 closest neighbours. The output of the filter is then used to classify the block where the standard check of the sign is required.

3.3.3 Region Growing

Once the background is removed only blocks that are likely to be part of a container remain. These blocks need to be further grouped into whole containers. This is done by combining blocks that are similar in nature into regions. These regions will then be used for the final SVM to determine what type of material the region is likely made out of.

The method proposed is a modified version of the local variation as shown in (Felzenszwalb and Huttenlocher, 1998). Each pixel is considered a node in a graph, which is connected to its 4 nearest neighbours. The edge weights are based on the degree of similarity to the pixels. Regions are constructed by finding subgraphs that contain the smallest edge weights.

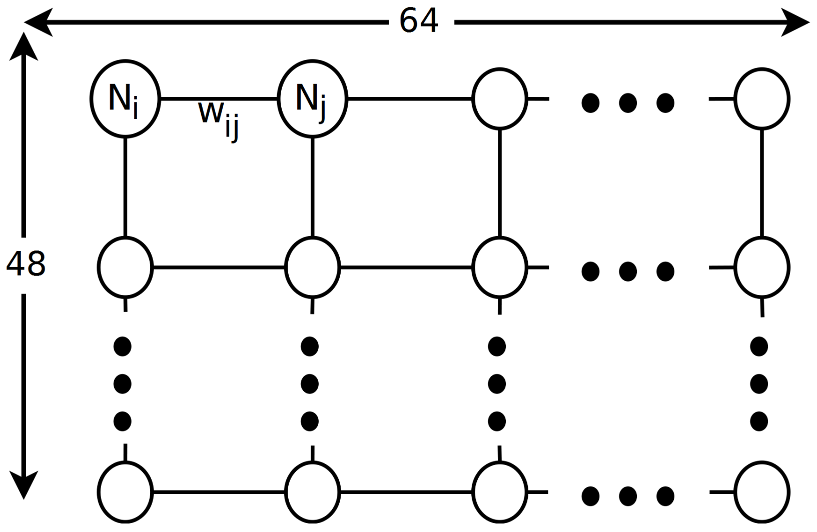

Figure 3.6: Graph used in region growing technique. Each node N corresponds to a 10×10 pixel block, with edges connected to the four neighboring blocks. Each edge has weight w_{ij} equal to the dot product between the histograms of node i and node j

In (Felzenszwalb and Huttenlocher, 1998), the difference between neighboring pixel intensities are used to measure the similarity of two adjacent pixels. This thesis uses a modification to this approach by considering the similarity of two blocks by the value of the dot product between the block histograms. The change requires slight modifications to the existing formulation because the difference in pixel intensities increases as two pixels become more dissimilar, while the dot product of the histograms decreases with increasing dissimilarity.

The first step is to consider each block as a node in a graph. Each node is connected to its 4 nearest neighbors by edges. These edges have weight equal to the dot product between the histograms of the connected block. The goal is to find subgraphs that represent the minimum spanning tree of each region. Figure 3.6 shows an example of such a graph constructed using 10×10 blocks on a 640×480 image.

All the edges are created for each block, then are removed from the graph. The edges are sorted by value and added to the graph one at a time, starting with the highest value. There are three cases when an edge is added to the graph:

1. When an edge is added between two nodes, where neither node is assigned to a region. The two nodes are then assigned the next highest unassigned region number.

2. If an edge is added between a node within region i, but the other node is not assigned to a region. The unassigned node is then added to region i.

3. If an edge is added between two nodes that are already assigned to two separate regions R_i and R_j. The two regions R_i and R_j will be merged if:

$$w(E_{cur}) > min(T(R_i), T(R_j)) \tag{3.1}$$

where $w(E_{cur})$ is the weight of the current edge (E_{cur}) that is being added to the graph, and $T(R_i)$ is a threshold function defined for each region as:

$$T(R_i) = w_{min}(R_i) - \frac{k}{|R_i|} \tag{3.2}$$

where $w_{min}(R_i)$ is the minimum similarity edge (the edge with the smallest weight added to region R_i), k is a constant that is chosen to control the degree of merging, and $|R_i|$ is the number of nodes contained in R_i.

If the two regions are not merged then the edge is ignored. An edge is also ignored if it is between two nodes that are assigned to the same regions. The parameter k is chosen to control the size of the regions. While the number of nodes in a region is small, the $T(R)$ function will be dominated by the k term, allowing the merging of all but the most dissimilar regions. As the number of nodes increases $T(R)$ approaches $w_{min}(R)$, so only blocks that are as similar as the least similar edge will be added.

3.3.4 SVM Training

All the support vector machines were trained using the LibSVM package (Chang and Lin, 2001). To construct training images, individual containers were imaged and a bounding box was used to segment the container from the background. These histograms of the segmented containers were used as input to the training to classify PET and Polycoat containers. For the background removal SVM, several images of the background were taken and divided into blocks. These blocks were used as positive background examples, while blocks taken from PET bottle images were used as negative examples.

3.4 Hardware Implementation

The algorithm is implemented on an Altera Arria II GX development board. The development board has embedded circuitry to interface to a computer's PCI-E bus. Image data is sent to the computer via Gig-E camera. Image data is sent to be processed on the FPGA via the PCI-E bus. An overview of the FPGA architecture is shown in Figure 3.7.

Image data arrives through the PCI-E express bus and is placed into a FIFO 8 bytes at a time. The image data is sent one line at a time starting at the top left corner. A wrapper then extracts 8 bytes (8 pixel) per clock cycle and passes it onto the block construction module. These are then serialized via a finite state machine and sent off to the next stage for processing.

The Sub-Block Creation Module extracts the data from the serializer and builds the histograms for one image line of blocks at a time. In parallel to block construction

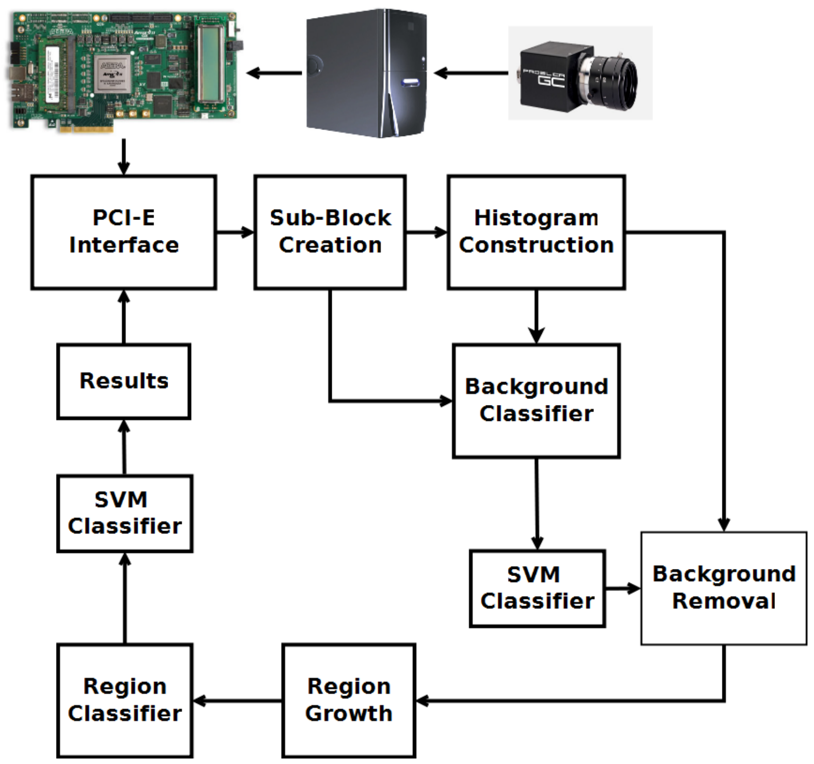

Figure 3.7: Overview of the FPGA implementation

is the background classifier stage. This stage classifies each block as either being part of the background, or a container. Once all the blocks are created, and background blocks removed, the blocks must be assembled into regions

3.4.1 Data Serializer

The Data Serilizer Module is part of the PCI-E interface. Data on the PCI-E bus is sent to the FPGA cards in packets of 64 bits (8 pixels) at a time. A simple finite state machine (FSM) parses the 8 pixels, and places them into packets which correspond to one line of each block (N pixels for an N×N block). These packets are passed onto the block construction module. A data available signal is used to signal the availability of a data packet. If the signal is asserted, then a data packet is ready, otherwise no packet is present and the block construction is paused until data becomes available.

3.4.2 Sub-Block Creation

The data packet from the data serializer is broken up into individual pixels, and the most significant n bits of each pixel are used to construct the histogram of each block. All the data for one line of each block arrives in one packet, and is used to add to the blocks histogram. The total number of clock cycles required to construct a block depends the block size, since only 8 pixels values are available each clock cycle.

The pixel values are used to construct the histogram vector for each block. The histogram vector is a vector where each dimension contains the number of occurrences of each pixel value within the current block. For an n bit vector there are a total of 2^n values that a pixel can take. Thus there are 2^n dimensions or bins in each histogram vector.

Each histogram vector is stored in one location of a dual-port RAM, with a total size of up to 256 bits per location. The entire histogram vector is stored in one location of the RAM allowing it to be read out in one clock cycle for quick updating and processing.

The most significant n bits of the pixel value are used as an address input to a look-up table (LUT). Each LUT contains a bin vector. The bin vector has the same dimensionality as the histogram vector but contains only a single one in the position that corresponds to the pixel value e.g., if the input is 13, the one will appear in the 13th position, with all other values equal to 0.The histogram vector for a block is then the sum of all the bin vectors corresponding to the pixel values in that block.

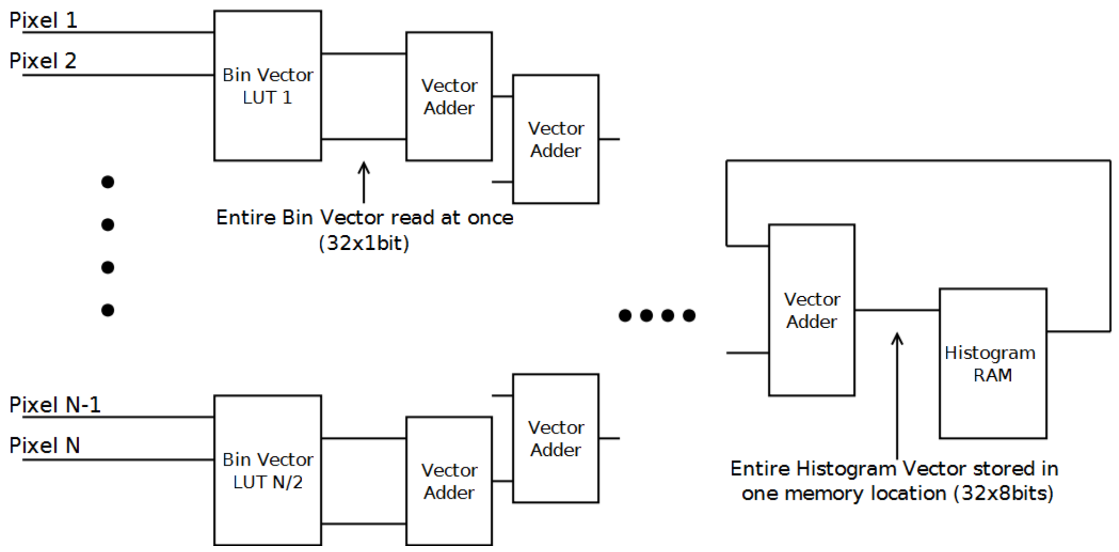

Figure 3.8: Histogram Construction Hardware Implementation

Dual-port ROMs that are embedded in the FPGA are used for the look-up tables. Dual-port ROMs are used because they allow for two values to be read out at once and they are already implemented on the FPGA. For a N×N block, a total of N/2 ROMs are required to read out one line of a block at once. Each of the N bin vectors that are read out are summed, and added back to the current histogram. This is shown if Figure 3.8.

3.4.3 Background Classifier

The background classifier process runs concurrently to the sub-block creation. This hardware module uses the second port of the dual port RAM used to store the histogram information for each block. The data is sent off to the SVM module for classification and the result returned is used to classify the block. The real valued output of the SVM is averaged by the surrounding block output values in order to decrease the likelihood of a false positive on the background pixel.

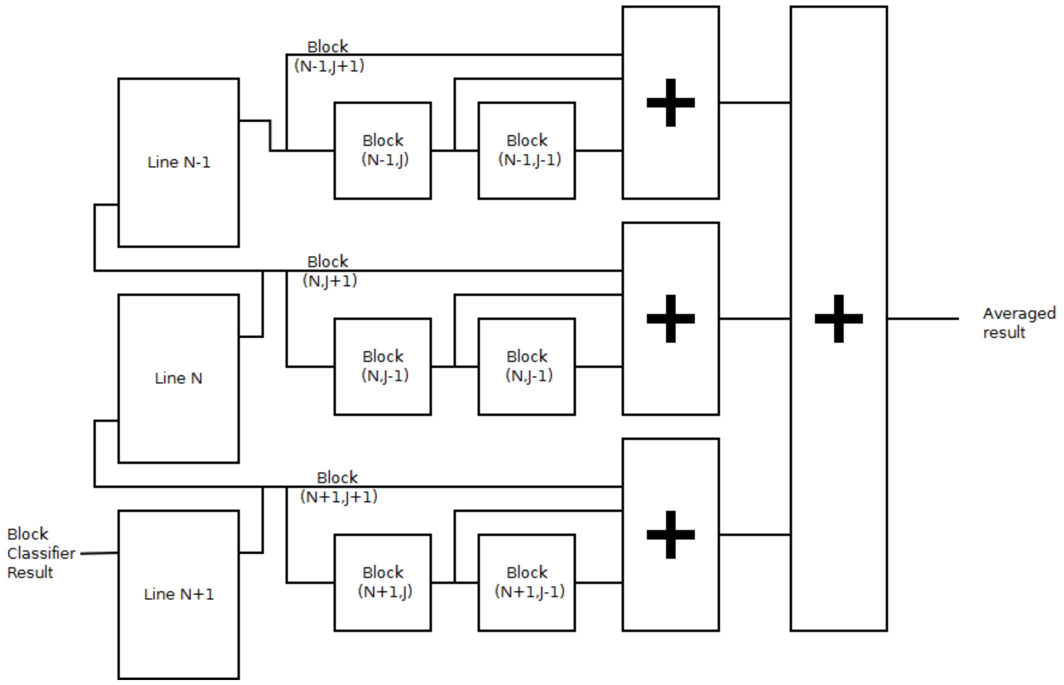

Figure 3.9: Background Averaging Filter

The filter is simply an averaging filter using the nearest 8 points surrounding the block being classified. The filter block consists of three RAMs, that contain the real valued classification results for a row of blocks. Each value is read out and is also read into the next RAM. Once an entire row is finished, row n becomes row n-1 and is already in the correct location. Since only the sign is important, the value does not need to be divided by 9, thus saving hardware area. The hardware implementation is shown in Figure 3.9.

3.4.4 SVM

The SVM classifier was presented earlier in Chapter 2. The main components of the classifier are the support vectors. The dot product is taken between the input vector and every support vector, multiplied by the corresponding Lagrange multiplier and

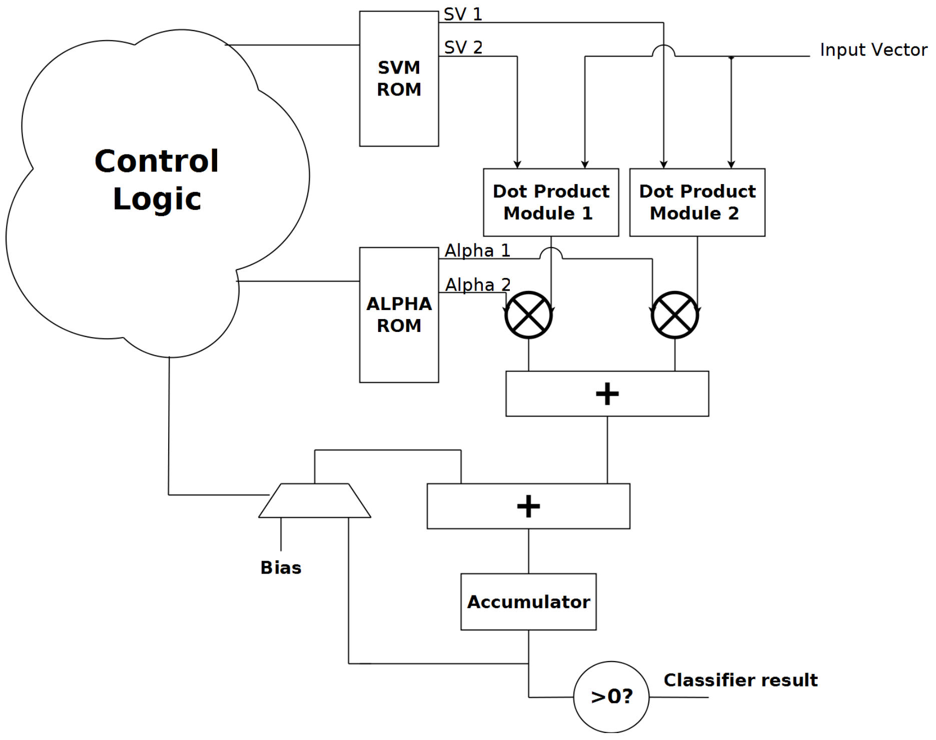

Figure 3.10: Block diagram of the SVM module

the result is summed together.

Figure 3.10 shows the overview of the SVM module. Each dot product is performed independently. Results are multiplied by the corresponding α, and accumulated for classification. The only limitation is the speed at which the support vector can be fetched from memory. For a small number of support vectors such as used in this implementation, the fastest throughput can be obtained when the support vector data is stored on-chip.

The SVM module accepts one vector and a signal to tell which set of support vector to use, either those for background classification, or those for classifying PET bottles from Polycoat containers. This input is simply used as a switch that selects which half of the memory to use. The background classification values are stored in the lower half, while the recyclables classifier values are stored in the upper half of the memory. The implementation is identical for the corresponding Lagrange multipliers.

An FSM is used for data control, which loops through the address space to read out all the required data. Dual Port ROMs are used to increase available bandwidth, allowing double the throughput of data. Two entire support vectors are read out at the same time and are presented as input to the dot product module along with the input histogram vector.

Once the result is returned from the two dot product modules, they are multiplied by their corresponding Lagrange multiplier values and added to the cumulative total. The Lagrange multiplier and support vector data are scaled by a factor of 2^{16} to allow for fast fixed point math, while still maintaining accuracy. For large amounts of support vectors this error may accumulate, however the largest number of support vectors observed for any classifier was no more than 30.

Polynomial kernel functions were also tested by applying the following to the output:

$$(X^T Y + 1)^d \tag{3.3}$$

for various d. This was computed by successive multiplications of the result to obtain the correct power. It was found that the number of bits required to accurately represent the result slowed down the overall clock speed, and were not justified in the marginal increase in classification results, especially for large d.

3.4.5 Dot Product Module

The dot product module accepts two vectors as inputs and returns the dot product of the two vectors as an output. The summation of the dot product takes a total of

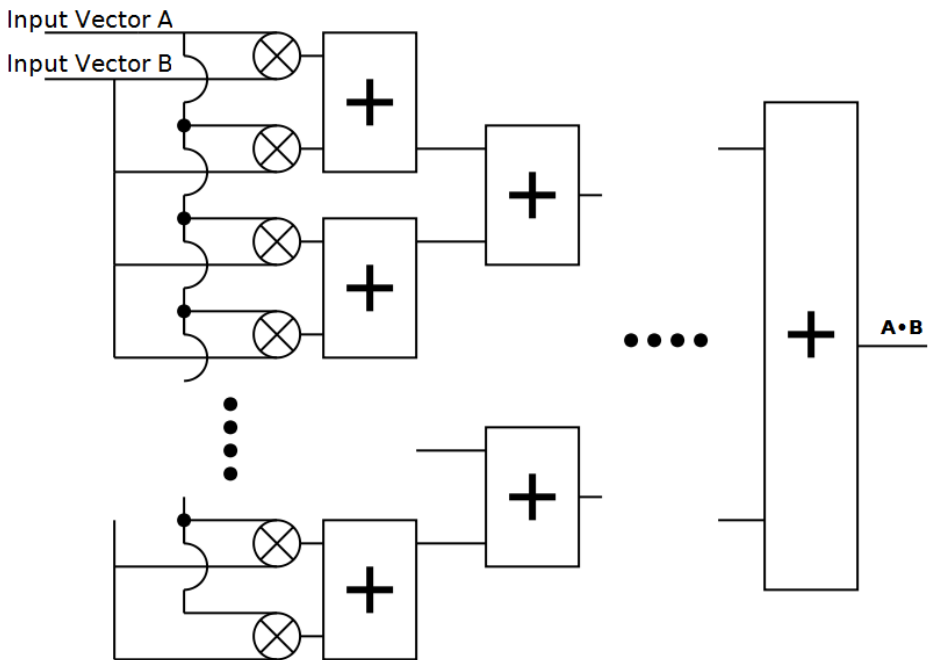

Figure 3.11: Dot product hardware module

3 clock cycles, but the design is pipelined to allow input of new vectors every clock cycle.

The structure of the module is straight forward and is shown in Figure 3.11. Each dimension of vector A is multiplied by the corresponding value of Vector B. The results of these multiplications are summed together in an adder tree with intermediate storage to allow for pipelining.

3.4.6 Region Growth

The region growing module is the module that consumes the most time to process due to its lack of parallelization. The region growth as described above takes the block histograms and converts them into values based on similarities to the neighbour. Similar regions are added together to form container regions.

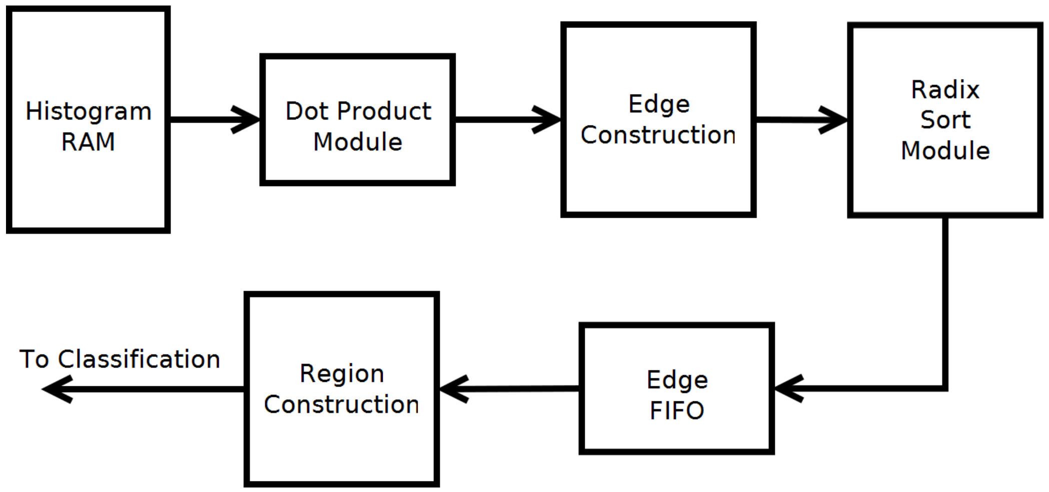

Figure 3.12: Block diagram of the region growing module

The first step is to use take the dot product between adjacent block histograms. Because there are two dot-product modules from the SVM that are idle when the Region Growth Module is active, they can be reused to save on resources. Dual-Port RAMs are also used to facilitate the bandwidth require to keep both Dot Product modules in use. The hardware module that performs this function is shown in Figure 3.12.

The results of the dot product are packed together with position information to form a packet of data. The packet contains the position of the current block, which direction the dot product (either to the right or down) was taken in, and the result of the dot product which represents the edge weight. The first bit of the packet determines direction, 1 indicating the dot product was between the current node and the one below it, while 0 indicates that the dot product was taken between the current node and the node to the right. The next 15 bits contain the block address of the current block. The last 16 bits are used to store the edge weight information. Once the edges are sorted by edge weight, the position information allows for determining

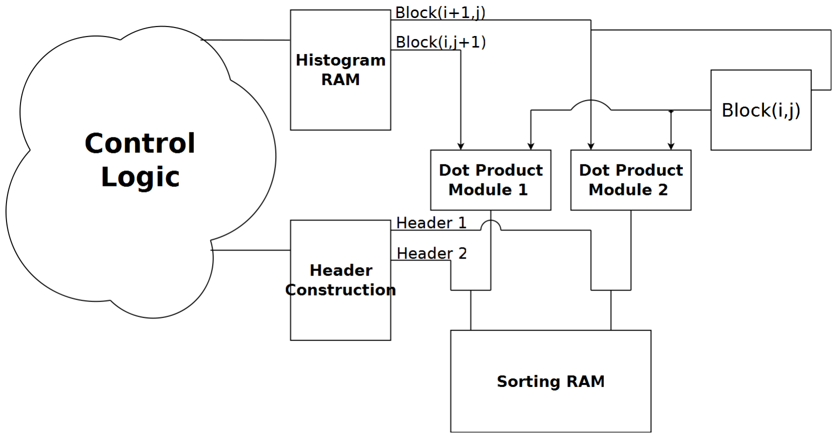

Figure 3.13: Hardware Module that computes edge weights for region growing

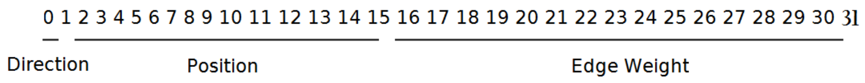

Figure 3.14: Packet structure

the beginning and end node. The start node address is simply stored in the packet, while the second address is start node address+1 if the first bit of the package is 0, else it is first node address + number of blocks per line if the first bit is a one. The structure of these packets are shown in Figure 3.14.

These packets are then sorted by edge value from highest to lowest. The method used for sorting is a radix sort, which sorts data by iteratively grouping values by examining individual bits, starting at the least significant and finishing at the most significant. At each iteration of the sort, the data is arranged based on the current bit. If the current bit is a one it is places ahead of all values that have a zero. In the case of ties order is maintained from previous iterations.

The hardware implementation of the radix sort involves a Sorting RAM and a Sorting FIFO. Values that have the current bit equal to one are read directly into the RAM, while if they have a zero they are placed in the FIFO. Once an iteration is over

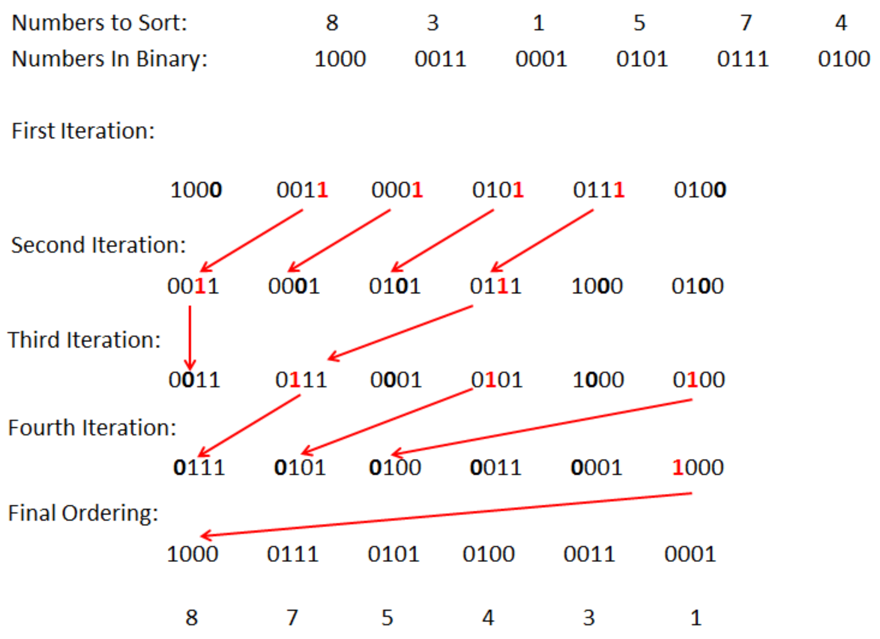

Figure 3.15: Example Radix Sort

all the data currently located in the RAM is read out and placed accordingly (1-back into RAM, 0- FIFO). Once all the data stored in the RAM is sorted, the remaining edges in the FIFO are placed in the remaining address space of the RAM. To simplify hardware, at each iteration the edge value is rotated to the right so only the last bit is ever checked. At the last iteration all the sorted data is placed in the RAM. An example radix sort is presented in Figure 3.15.

Once sorted, the values are ready to be grouped into regions. Edges are fetched with the highest weights first. The location data are used to determine the location of the two nodes and both nodes are read out at the same time. The regions are built by the rules described earlier in this chapter. To assign a node to a region, the region number is simply written to the address of that block in the Region RAM. To merge two regions, the region RAM is searched through to replace all entries of the region to be replaced by the region that is taking over the blocks.

A second RAM is used to aid in region merging. This Region Information RAM stores the first and last block of the region. This is used when regions are to be merged, by allowing the search to start at the first block, and to terminate to early once the last block is written. These values are updated every time a new node is added to a region or two regions are merged. In addition this RAM stores the minimum edge weight and number of nodes in that region for computing the threshold function.

After the regions are fully constructed the Region Ram is used to fetch all histogram vectors associated with a region number. These histogram vectors are added together to form complete histogram vector for that region. This vector is renormalized by dividing by the number of blocks in the region and then passed to classification. Once classified the classification results and block numbers are returned to the PC for each region.

3.5 Chapter Summary

Chapter 3 presented the algorithm and the implementation details. This showed how using a novel background removing method that classified blocks of the images using a SVM could be used to classifiy several containers at once. Blocks were assembled into regions that were likely containers, and these were used to classify the regions. The details of the FPGA implementation were also examined. The pipeline structure of the algorithm lends itself well to this platform of computation.

The next chapter will examine the results of the presented algorithm on real and synthetic image data. Throughput (frames per second), logic utilization, and accuracy will be measured. Parameters that effect these quantities will be examined to determine optimal values to use.

Chapter 4

Results and System Performance

This chapter examines the results for the algorithm presented in Chapter 3. Factors that affect both the algorithm performance and the hardware implementation are identified. The primary performance characteristics that were measured were recognition rate, false positive rate, and throughput in frames per second. From a hardware perspective, the primary concern is resource utilization in LEs, hardware multipliers, and RAM blocks. These were examined by varying key algorithmic parameters such as block size, k and the number of bins in the histograms. Parts of this chapter were published in the IEEE International Symposium on Sustainable Systems and Technology (ISSST) (House *et al.*, 2011).

4.1 Implementation and Testing

The algorithm presented in chapter 3 was implemented in SystemVerilog, with the accuracy and speed tested on an Arria II GX device and simulated using ModelSim. A total of 38 Polycoat and 49 PET images make up the training set, while the testing set is made up of 17 Polycoat and 32 PET images. 500 images were generated in MATLAB by using the extracted PET and Polycoat containers from the testing set. To generate a test image, a random number of containers were selected to be added to the image, evenly distributed between two and five inclusive. Each container that was chosen to be added underwent a rotation and translation placing it at some part of the new generated image. Small amounts of overlap between containers were allowed to simulate items touching on a conveyor system. The background for the synthetic images was the same background used to image the containers but with no containers present. A small subset of the testing images is shown in Figure 4.1. The bottom right image is the background which was used as the background in the synthetic images and to train the background classifier.

These images were used as input to the ModelSim package, to simulate the hardware implementation of the algorithm. During the simulation process only the algorithm section was tested. To allow for quick simulation times only the outputs of the PCI-E block function were simulated and interactions with the bus were not modeled.

4.2 Results

The parameters that provided the best performance were found to be a block size of 8×8 using 5 bits to create the histogram, and using a k value of 0.2. This setup was tested on an Altera Arria II GX device and simulated using ModelSim. The hardware implementation's resource utilization is shown below in Table 4.1.

The RAM required is the bottleneck for the FPGA implementation. It is the highest used resource at 74.5% but can still fit successfully on the chip with a margin of over 25%. The total logic elements (LEs) required to implement the design is only

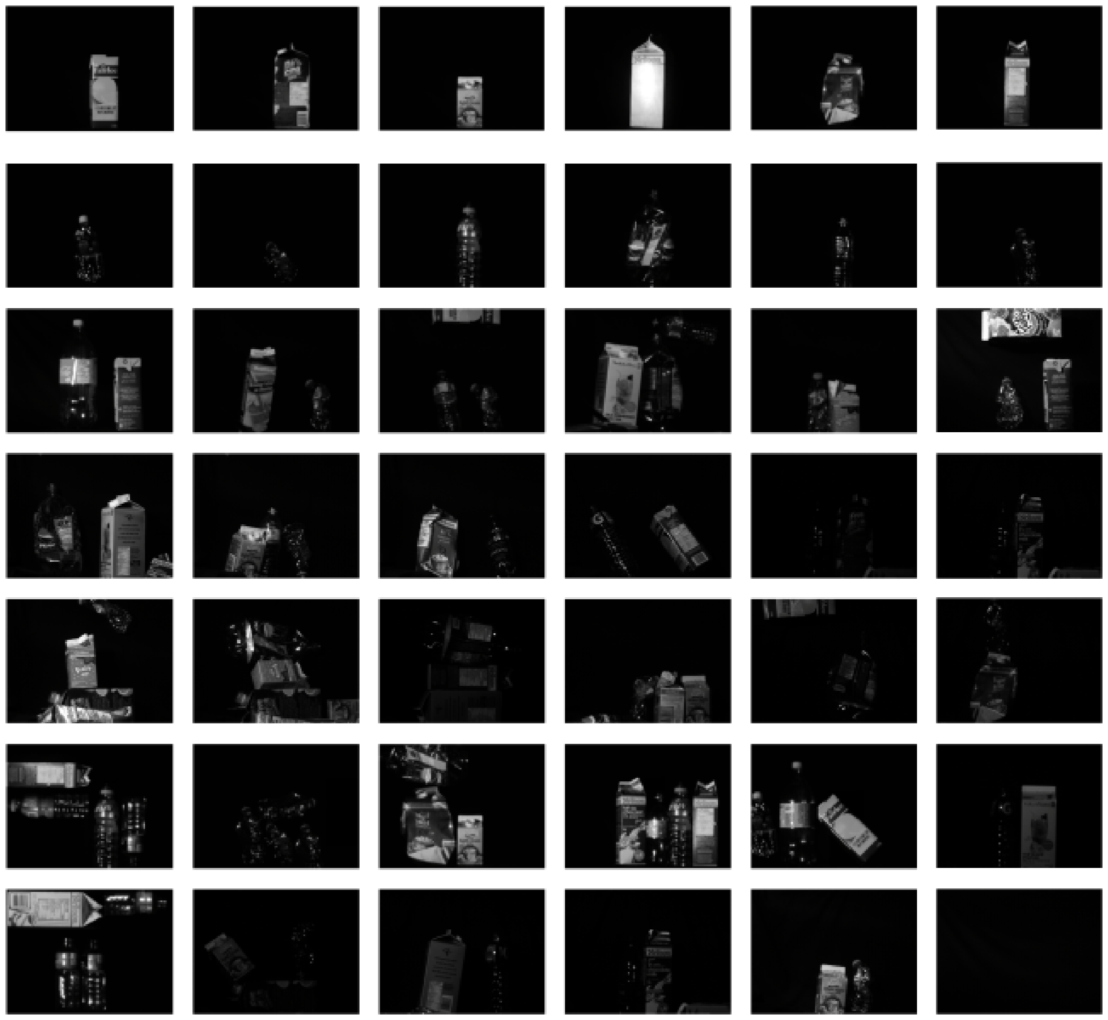
Figure 4.1: Example images

	ALMs	18-bit Multipliers	M9K RAM Blocks
Total	17998	289	544
Precentage	18%	50.1%	74.5%

Table 4.1: FPGA resource utilization

18% of the total LEs on the device. This leaves room for additional logic that could be implemented for increased performance or interfacing to peripherals in a real world system, such as a bottle rejection mechanism. The design uses 289 of the 576 available multipliers, with many still available for additional feature that could be added on.

Throughput was tested using 500 synthetically generated images. For the parameters given above, using 640×480 pixel images an average throughput of 85.27 FPS was achieved. The worst case run time allowed was 12.8 ms corresponding to a framerate of 78.12 FPS, while the best runtime of 11.0ms allows for a maximum framerate of 90.13 FPS.

Figure 4.2a)-d) shows the results of the algorithm applied to various non-synthetic images containing examples PET and Polycoat containers in various arrangements and states of proximity. These images were sent to the FPGA daughter card for processing.

Figure 4.2a) demonstrates for separated objects the region growing algorithm produces accurate segmentation, without too much fragmentation of the regions.

Figure 4.2b) demonstrates the difficulty in finding certain bottles. The areas that do not have many specular reflections tend to be classified as the background and the label gets classified as a Polycoat container. Despite significant portions of the PET bottle being segmented out or mislabeled, enough of the bottle is present such that most of it is correctly classified.

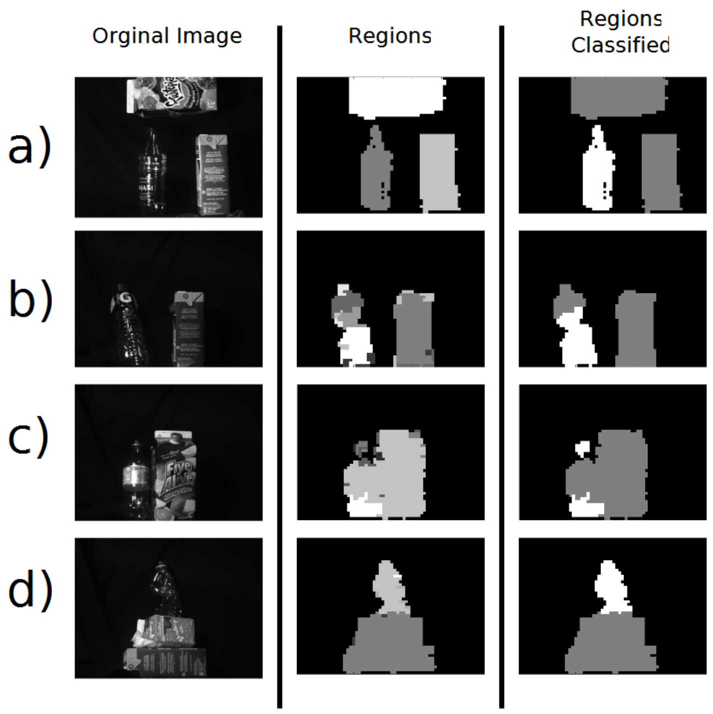

Figure 4.2: Results of the region growing method and classifier for several test images. The first column shows input image. The second column contains the result of the region growing (shade indicates region). The third column contains the result of the classifier. White regions indicates PET bottles and grey regions indicate Polycoat containers. This Figure is reproduced from work published in the IEEE International Symposium on Sustainable Systems and Technology(House *et al.*, 2011)

Figure 4.2c) demonstrates an example of a Polycoat and a PET bottle touching in an image. Because the label has similar optical properties to Polycoat material, there is a tendency for the region growing algorithm to merge the label with a neighbouring Polycoat container. This can be controlled by decreasing k in the threshold function, but this results in a fragmented segmentation. Despite the merging of the label with the Polycoat, there is still enough information for the classifier to correctly identify parts of the PET bottle. Figure 4.2d) demonstrates closely spaced Polycoat will in general be merged into one region.

The success rate of the classifier is shown in Table 4.2. The classifier is able to successfully identify the material with an accuracy of just over 93%. The test data consisted of 500 synthetic images with varying number and types of containers. An object was considered successfully classified if the majority of the region contained within the original image was correctly classified.

	%
Correctly Labelled PET	92.15
Correctly Labelled Polycoat	94.19
Falsely Labelled PET	5.81
Falsely Labelled Polycoat	7.85
Average Accuracy	93.07

Table 4.2: Classification Results

Although this result is somewhat low compared with similar results from (Nawrocky *et al.*, 2010) , which can most likely be attributed to the extra segmentation that is done, due to the nature of the problem and the high speed implementation of the algorithm many passes could be allowed over the same bottles. As the containers are identified, containers with a high degree of confidence could be ejected from the conveyor and the remaining bottles could enter another round of classification or possibly placed into the original stream for another pass at the sorting. Since one of the primary causes of container misclassification is due to the overlap of multiple containers, by making successive passes, fewer containers would be present at each step, which would translate into better rejection rates in later stages. By operating at high speeds, multiple passes over the containers could be done without sacrificing throughput of the MRF.

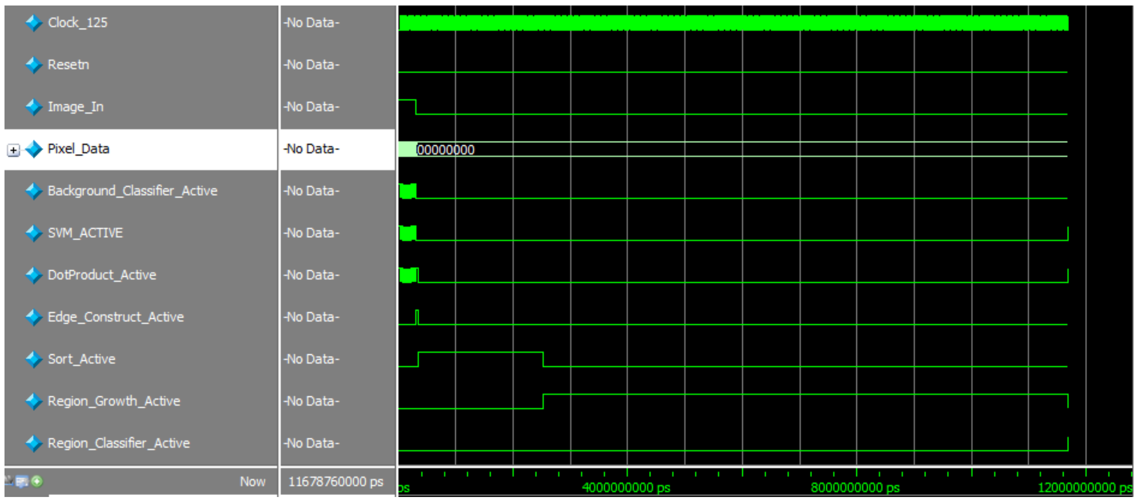

Figure 4.3: Simulation results with timing information for each section of the algorithm

4.2.1 Timing

Figure 4.3 shows the results of a simulation with timing information for the different sections of the algorithm. The runtime parameters are for an 8×8 block size. From the diagram the total runtime is 11.7 ms.

The histogram construction takes a constant amount of time, which is the total time it takes to transfer the image from the PCI-E bus. Since pixel data is sent eight pixels at a time from the PCI-E bus, for a 640×480 image the total number of clock cycles to read in the image is 38400 or 0.307 ms with a 125 MHz clock. The background removal is completed in parallel with the histogram construction and so adds an insignificant amount of time in the order of 1000 clock cycles or roughly 0.008 ms.

The edge construction (see Figure 3.12) module is able to produce two edges every clock cycle with a few lead in and lead out cycles for each row. The total time taken is then proportional to the number of blocks. For a block size of 8×8 there are 4800 blocks which takes approximately 0.039 ms. The radix sort time requirement is proportional to the number of blocks required to sort. Since the total number of bits used for the dot product result is 16, 16 passes over each edge is required to sort all the values. The total number of edges to sort is proportional to the number of blocks, which is proportional to the inverse of the square of the block size. In this example the total time to sort the blocks is 0.124 ms which corresponds to 20% of the blocks not being removed as background.

The region growing section is the most time consuming module and the run time depends on the input to the region and is not a fixed quantity. The time grows as the square of the block size. The total time taken in this run is 10.6 ms which is by far the largest amount of time. The main reason for this time is the poor parallelization of the region growing algorithm and the large amounts of memory accesses required.

4.3 Effect of Block Size

The block size affects many important aspects of the algorithm itself and the hardware implementation. Its effects were examined by using the synthetic images of the testing set. In genera,l by breaking the image into blocks the spatial data is compressed. For large block sizes much of the spatial distribution of the data is lost, and results that depend on accurate spatial information are degraded. Conversely, smaller blocks preserve spatial data but only a small section of an image is considered at one time. Results that require the consideration of larger areas, such as the region growth of a very colourful Polycoat container, suffers.

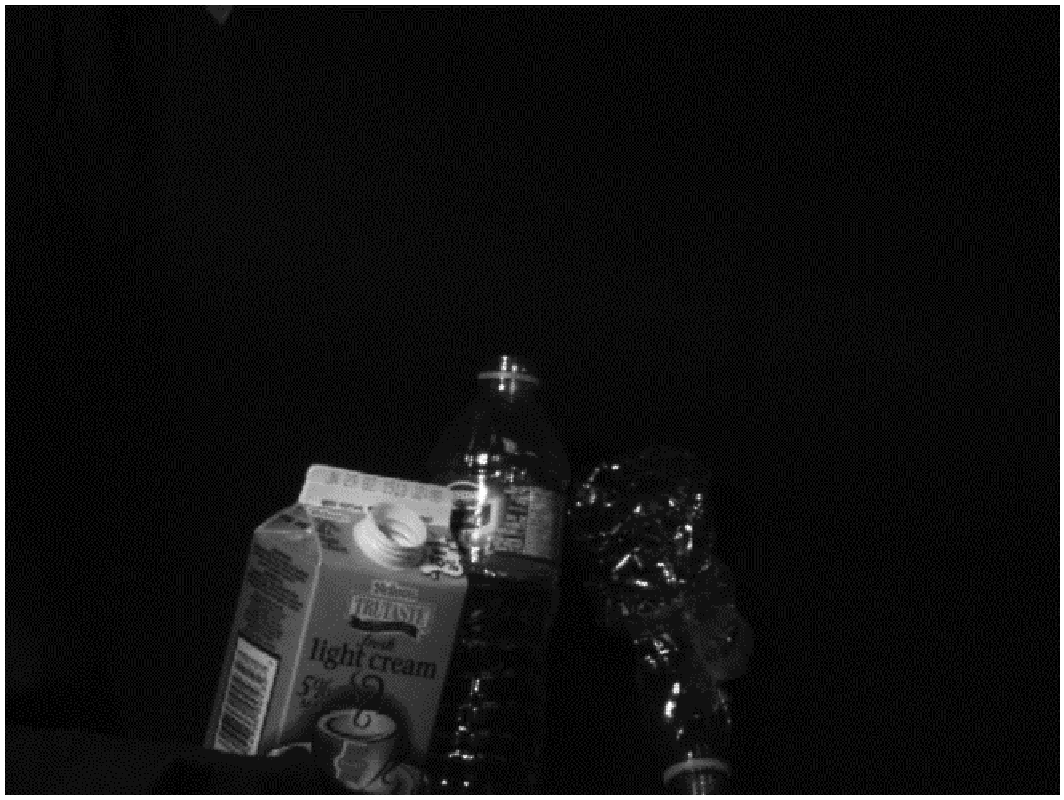

Figure 4.4: Input Image

4.3.1 Region Growth

Figures 4.4 shows an example input image and Figure 4.5 shows the effect of varying the block sizes has on the region growth (left column) and classification results (right column) using optimal k values.

For a block size of 5x5 pixels, the segmentation is very sparse with many disjoint regions. This is due to the small block size only considering very local pixel intensities. If the container has an area which is highly distinct from other regions then these regions are in general not merged. As block size is increased, data is considered from a wider area and the overall effect is larger regions. There is a much larger probability

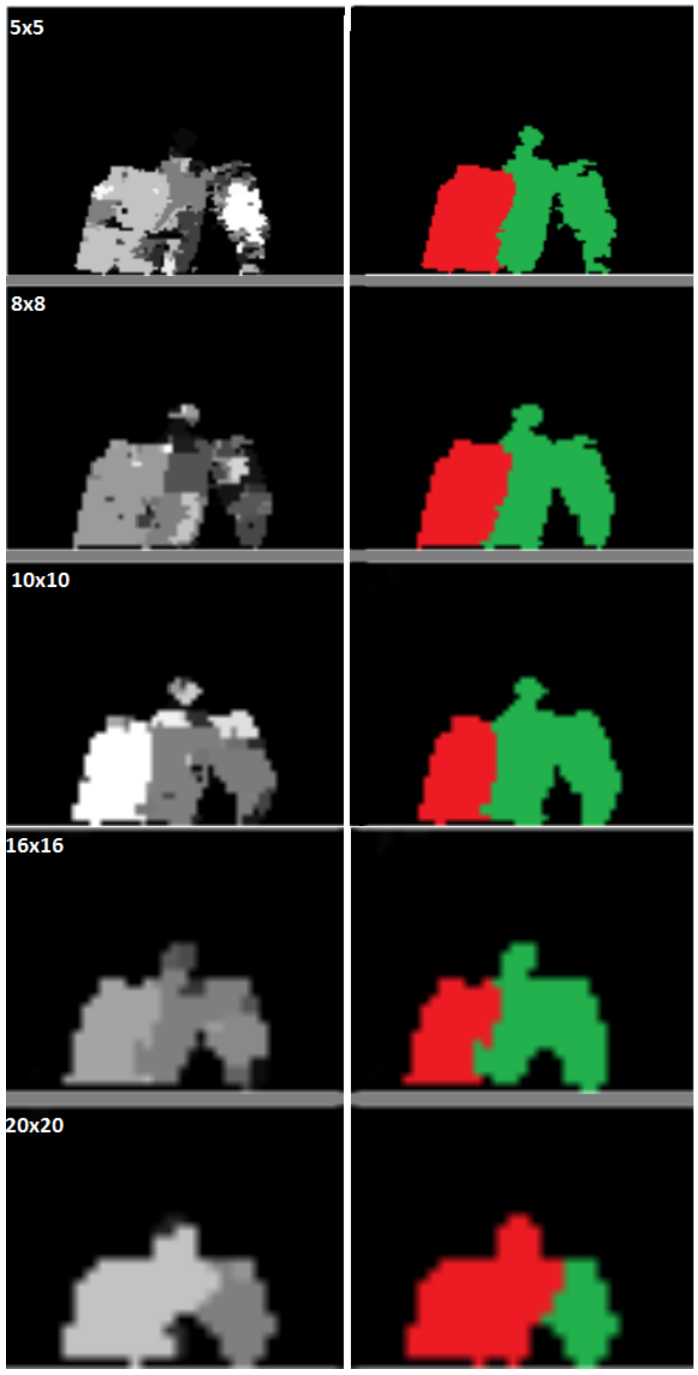

Figure 4.5: Results for block size 5x5,8x8,10x10,16x16, and 20x20

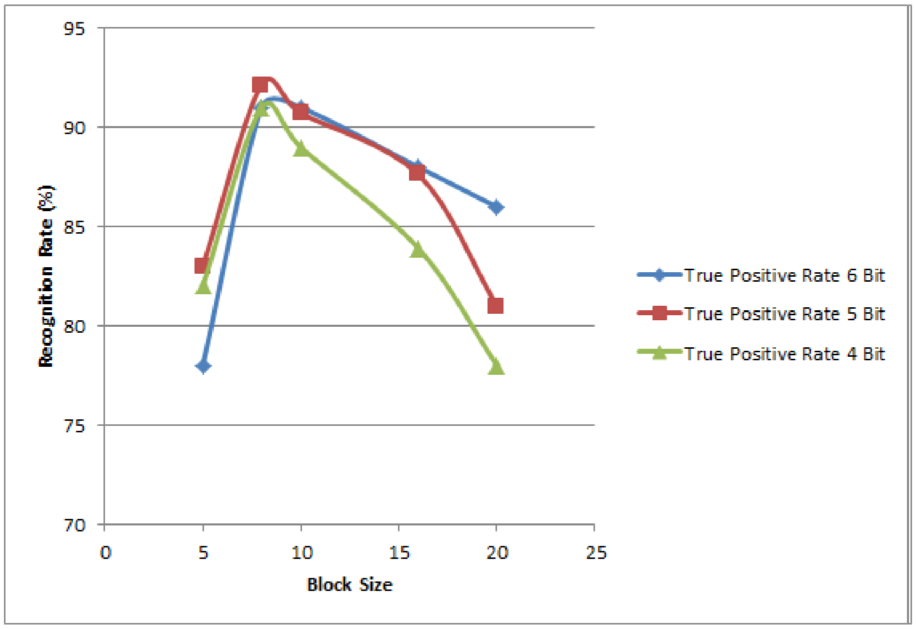

Figure 4.6: The True Positive rate for classification of PET bottles for 3 different histogram sizes

that a larger block will overlap dissimilar regions and contain pixels similar enough to each of its neighbours that it will likely be joined to both of those regions.

As block sized is increased further, the regions become much larger. The extreme of what is mentioned above, is that every region now has overlapping blocks and only regions that contain very different pixels will not be merged. For a block size of 20x20 the region of the Polycoat milk container absorbs the label of the one PET bottle and then merges with the entire second PET bottle. This does not happen for lower block sizes.

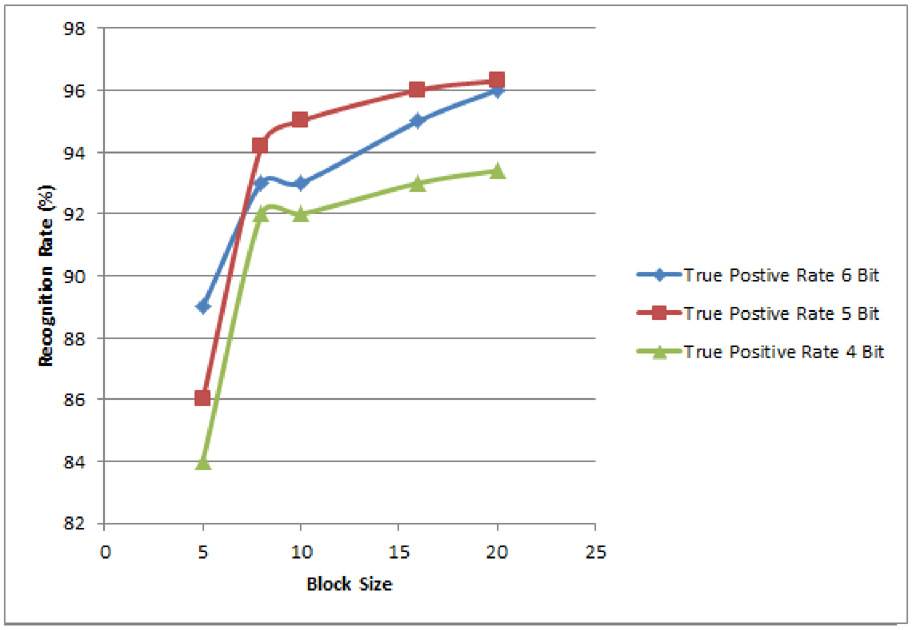

Figure 4.7: The True positive rate for classification of Polycoat bottles for 3 different histogram sizes

4.3.2 Classification

Figures 4.6 and 4.7 shows how the classification accuracy varies with block size for the two materials. For PET classification, the 8x8 performs best with the performance falling off for both ends as block size deviates from this value. The falling off at the larger sized blocks can be explained by the tendency for large blocks to group together dissimilar regions. The works well when a PET bottle is not touching another container. However, with large blocks, PET regions are much more likely to be merged with neighbouring Polycoat regions, especially if a label is present.

For the 5×5 block size there are two main issues why the PET classification rate drops: background misclassification and label regions. With small blocks the region growth cannot merge dissimilar regions effectively. Since labels tend to look

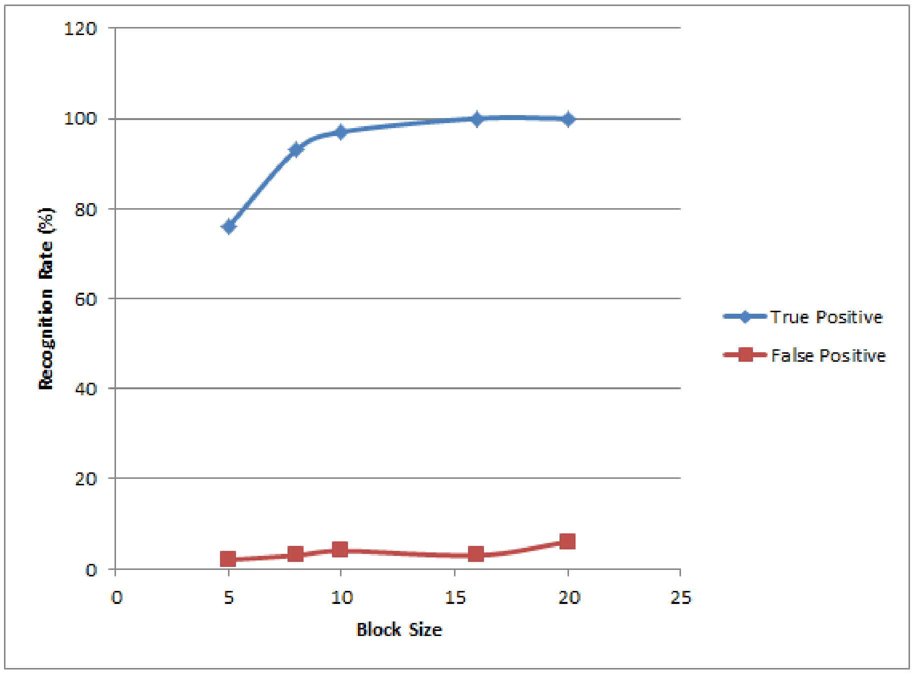

Figure 4.8: True and false positive rates vs. block size for background classifier

very different from the rest of the PET bottle, small block sizes often separate these regions. For certain PET bottles this is enough to cause the majority of the region to be classified incorrectly and register as a mislabeled container. The problem is compounded when a PET label is touching a Polycoat container that has similar characteristics to the PET label. The region is likely to be grouped together and for large labels can result in complete misclassification of the region.

The other major factor that influences the high error rate for small block sizes on PET bottles is the amount of background misclassification. The result of background classification as a function of block size is shown in Figure 4.8. Small block sizes are much more likely to incorrectly classify non-background blocks as background, while larger blocks almost always classify them correctly. This is predominantly a problem

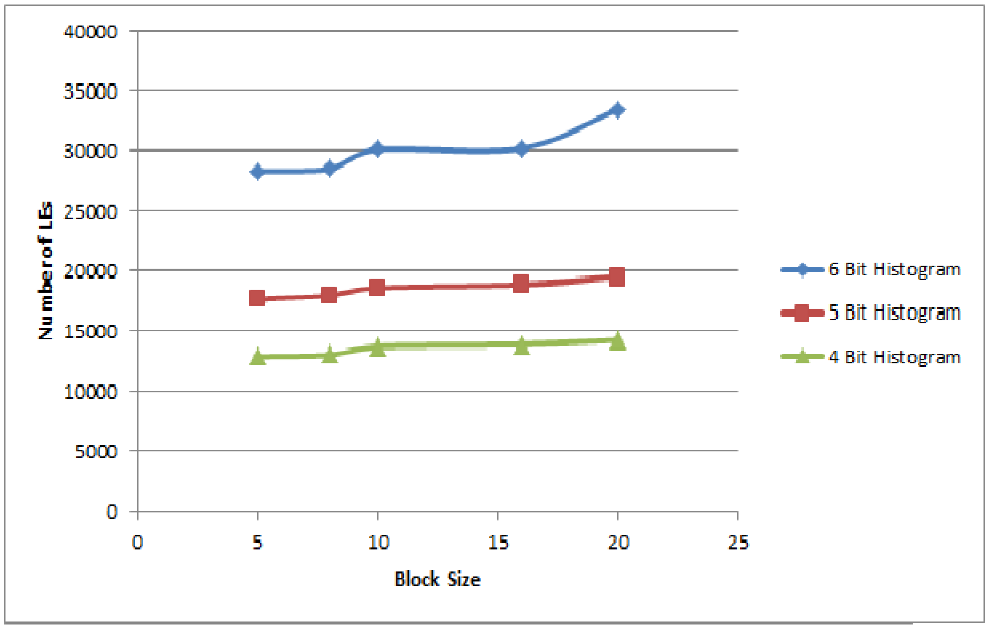

Figure 4.9: Number of LEs Required vs. Block Size

only for PET bottles, as Polycoat containers rarely have sections on them with pixels dark enough to resemble the background.

For Polycoat containers the correct classification rate increases with block size. This is expected because larger blocks are likely to merge many regions together. The histograms of Polycoat containers have their pixel intensities spread over many values, so by merging many regions together, a regional histogram is formed that tends towards resembling a Polycoat container. This results in a large bias towards Polycoats for large block sizes, which can be seen in the decrease in correct classifications for PET bottles and a high false positive rate for Polycoat containers.

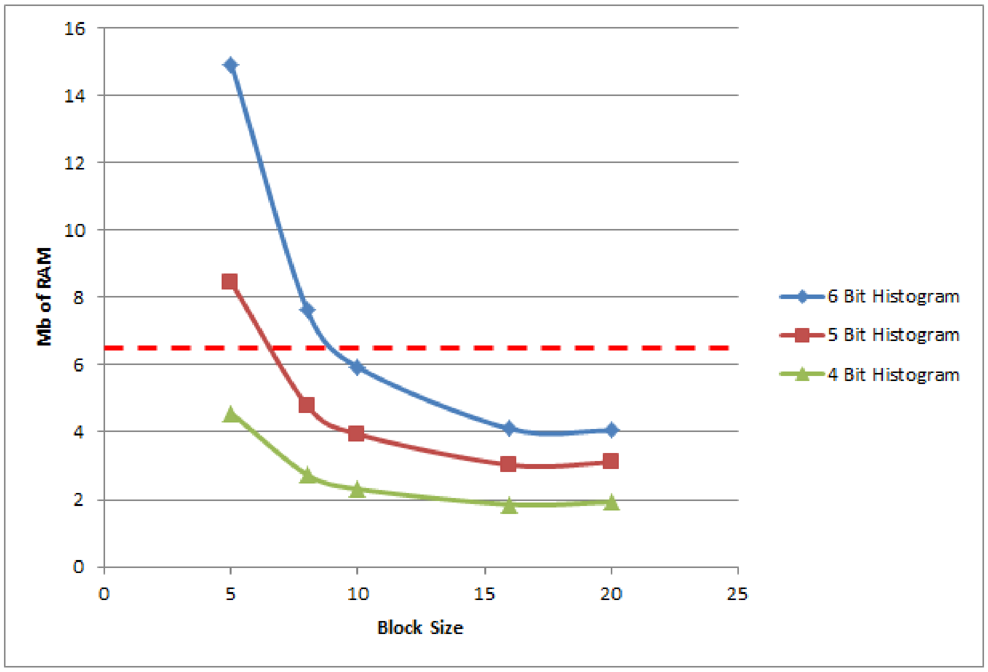

Figure 4.10: The amount of embedded RAM required to implement the design vs. block size. The red dotted line is the amount of RAM available on the Arria II GX chip and thus anything over this value will not fit on the FPGA but can still be simulated

4.3.3 Hardware Implementation

The effect of block size on the resource utilization of the hardware implementation is presented in Figure 4.9 and 4.10. Each of these shows the different effects the block size has on a particular hardware resource. The effect on the number of logic elements required to implement the design is mostly independent of block size with a slight increase for increasing block size. A jump is evident at the 20×20 block size due to histogram vector requiring 9 bits to represent each dimension instead of 8 for smaller blocks. Similarly a slight reduction in logic elements is observed when decreasing from 10×10 to 8×8 for the same reason. The slight variations of the logic element requirement could be explained by slightly larger multiplexers, adders, and surrounding logic.

The number of logic elements required increases as the number of significant bits taken to construct the histogram increases. This can be attributed to the amount of histogram vector arithmetic that takes place. The construction of the vectors, the support vector calculation, and the dot products all require the histogram vector to be buffered and manipulated. A change in size of the vector directly affects the amount and size of registers, adders and switching logic used in these operations. Using 6 instead of 5 bits results in a histogram vector that is 64 dimensional instead of 32. Due to the large increase in the number of logic elements, this implies that most of the logic elements are used in the histogram vector creation and manipulation. When the number of significant bits is reduced from 5 to 4, a less dramatic change is noticed, which implies that the histogram vector construction and manipulation is no longer the dominating factor in the LE usage.

The on-chip RAM resource is one of the overall bottle necks for the FPGA implementation. Large amounts of RAM are required for the implementation of the algorithm in hardware due to the requirement that the histograms for each block be stored on chip. In addition, RAMs are used to store support vector data, speed up logic functions and act as FIFO in the sorting sections.

The RAM requirement for the different block sizes is shown in Figure 4.10. By dividing the image into blocks, the effective resolution of the image is reduced by N in each dimension for an overall reduction of N^2. The diagram shows for implementations that require large amounts of RAM that this relation holds. However due to the addition RAM required mentioned above the trend does not continue for large block sizes and the required number of RAM bits flattens out.

As with the logic elements, when the number of bits used to construct the histogram is increased, a proportional increase in overall RAM usage is noted. For small blocks, the number of blocks to store dominates the RAM requirement. By increasing or decreasing the number of bits used in the histogram construction (and thus doubling or halving the vector size), a corresponding doubling or halving of the required RAM is observed. This is what would be expected when the histogram vector storage is the determining factor for the amount of RAM required. In the sections that are relatively flat, changing the number of bits used to construct the histogram does not have as large as an effect, which implies that the RAM that is being used for other tasks makes up the most significant portion of the used RAM.

The red dotted line is Figure 4.10 shows the total amount of RAM contained in M9K blocks on the FPGA. By using specially designed ALMs which can be converted into RAM, all but the 5×5 block size with 6 and 5 bit histograms can fit on chip, albeit at the expense of more LEs usage.

The on-chip RAM utilization could be greatly eased if off chip storage is used. In general this would reduce the overall throughput as additional time would be required to fetch the off-chip data. The exact effect would depend on memory access patterns.

4.3.4 Throughput

Lastly the block size affects the throughput of the hardware implementation. While the histogram construction section takes approximately a constant amount of time, the region growth sections and radix sort depends heavily on the number of blocks, and thus the block size. Figure 4.11 shows the dependence of frames per second

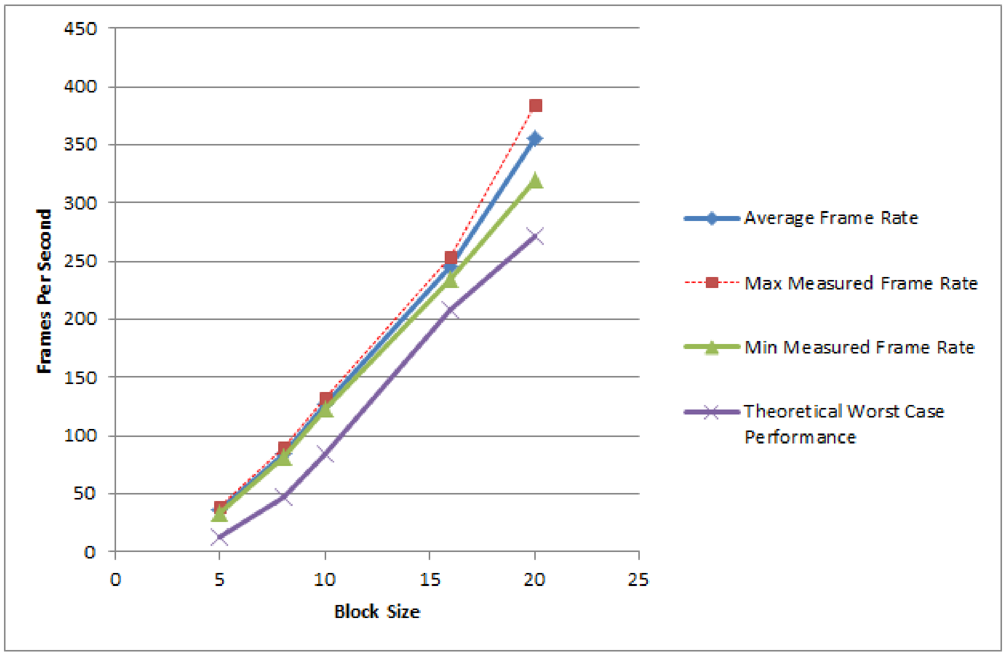

Figure 4.11: Frame rate vs. block size

and block size. Larger block sizes divides the image into fewer blocks, decreasing the required amount of time spent in the region growing section, while more blocks requires additional time spent.

The worst case performance, based on the region growing method growing the whole image in the slowest way possible (contains the most merge), is shown in Figure 4.11. Since the observed FPS follow a similar structure to the worst case where the region growing method dominates, it is clear that this module is the main bottle neck, and thus the main influence on the system's frame rate.

The maximum combined accuracy of the classifiers is 93.07% using an 8×8 block size. If a 10×10 block size is used a large gain in frame rate can be obtained. Current the average frame rate is 85.27 FPS with the smaller block, but by using the larger

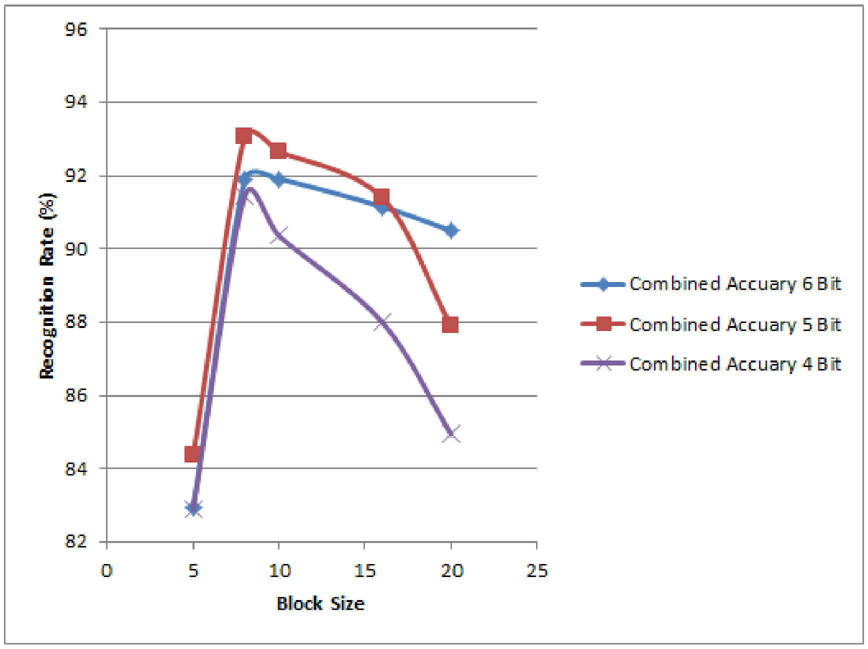

Figure 4.12: Average accuracy vs. block size and number of histogram bits

block size the average frame rate increases to 127.86 FPS with a small decrease in overall accuracy to 92.85%.

4.3.5 Overall Classification and Number of Significant Pixel Bits

Figure 4.12 shows the averaged classifier accuracy, which is the total number of correctly classified containers divided by the total number of containers. The number of significant bits used to construct the histograms has a significant impact on the overall classification rate. All three histogram types have a peak at 8×8 block size, which quickly decays as the block size is increased, except for the 6 bit histogram which levels off and then decreases near the end. This is likely due to the increase in correct classification rate of the Polycoat, which makes up the difference when the PET classification rate starts to fall at larger block sizes. In all cases the performance of the classifier is significantly reduced by using 5×5 block size. This is because of the poor segmentation that results from using this block size.

From Figure 4.12 it is clear that using more bits favours larger block sizes, while smaller block sizes favour fewer bits used to construct the histograms. This is likely due to the sparsity of the resulting histogram blocks. Having many dimensions makes it harder for dissimilar regions to merge, because they will be even less like each other in the dot product sense. This is a problem for large block sizes and would explain why increasing the dimensionality of the histogram would mitigate the effect. Discouraging merges actually helps large blocks form coherent regions instead of grouping everything together into one giant region.

4.4 Effect of k

The parameter *k* controls the size of regions that are formed in the region growing algorithm. A smaller value of *k* will create smaller regions that are similar in pixel intensities, while a larger *k* value will allow regions to grow much larger with some degree of difference within the region. Choosing the correct value of k is important for accurate segmentation and classification.

4.4.1 Segmentation Results

Figures 4.13 shows which value of *k* produces the best classification results based on block size. Several trends continue from the effects of block sizes on region sizes.

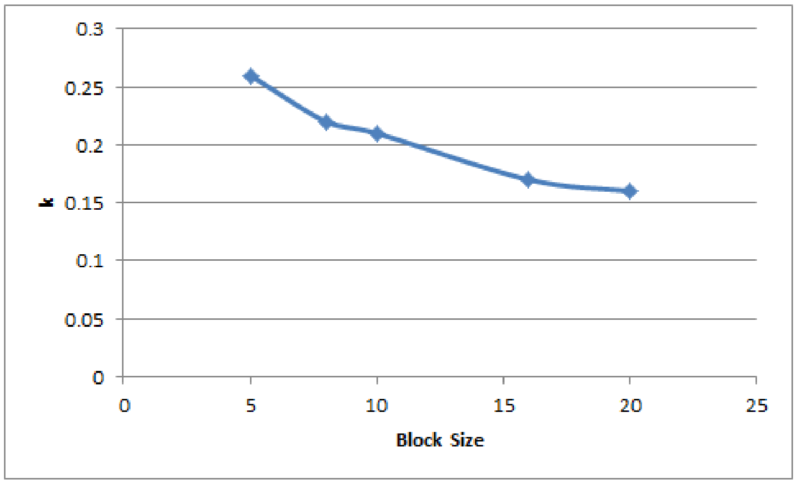

Figure 4.13: Optimum k value vs. block size

Small block sizes produce small regions due to the number of blocks, and the limited similarity to neighbouring blocks. To address this issue, k can be increased to allow for more merges and decrease the number of regions. This allows for whole containers to be grouped together instead of being fragmented and having a larger chance of misclassification. Thus smaller block sizes need larger k to produce accurate results. Large blocks have the opposite effect; they tend to group large regions together which encompasses many containers. This has a bias towards Polycoat containers due to the nature of their histograms. By decreasing k, the ability for the regions to grow very large and diverse is reduced. This biases the results to smaller regions. This trend appears in the data as the optimum k value is small for larger block sizes.

Figure 4.14 shows the effects on the classification results by changing the value of k using a block size of 8×8. Note the results are similar for different block sizes, as the results are similar in structure, just translated. From Figure 4.14 it is clear that there is only a small range of k values that are useful for a particular block size.

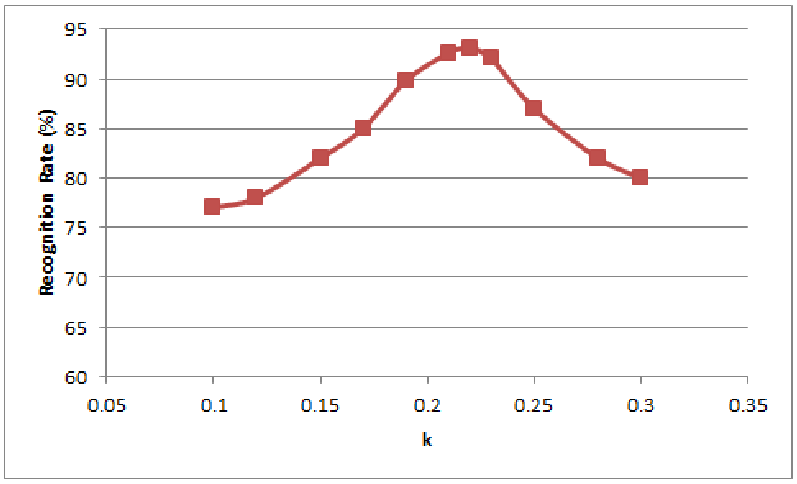

Figure 4.14: Recognition rate vs. k for an 8×8 block size

Deviation from these ideal values results in a steep decline in classification rate which is directly related to the quality of segmentation. Using a k value that is too large of a block size will result in over segmentation, while a value that is too small results in many separate regions.

4.4.2 Throughput

The value of k also weakly affects the throughput of the implementation. Since merging regions is a time consuming process in the hardware implemented, the amount of regions to be merged directly affects the frame rate. Lowering k results in fewer merges and thus a higher throughput is observed. The opposite is true for increasing k, the throughput is decreased. These results are presented in Figure 4.15 which shows FPS for different values of k vs. block size. From the graph it is clear that k only weakly affects frame rate, and block size has a much larger impact.

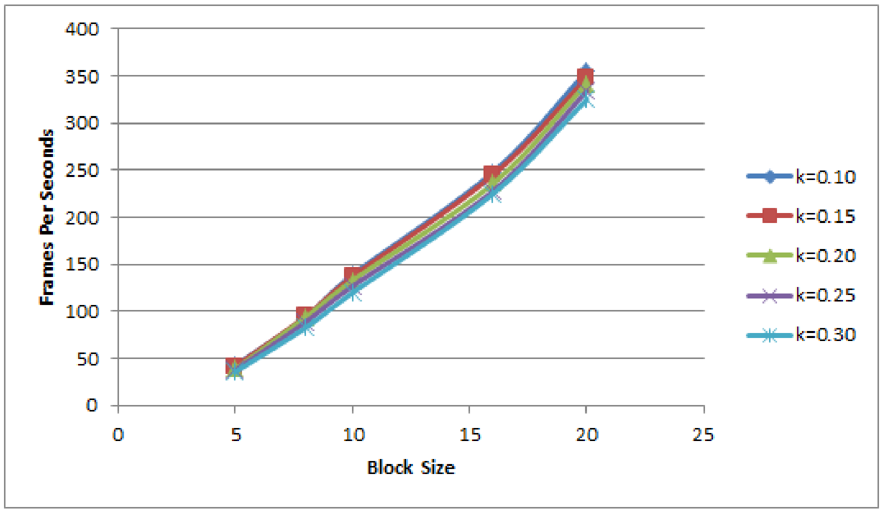

Figure 4.15: Fps vs. block size for different values of k

4.4.3 Chapter Summary

The results of the algorithm's implementation were examined. It was determined that the highest classification rate was obtained when using a block size of 8×8, however a block size of 10×10 has similar performance but is almost 50% faster on average. Other block sizes were found to be not useful based on degraded performance, slower speeds or increased logic usage.

The optimal number of bits to use for the histograms was the 5 most significant bits. This results in a histogram that has 32 bins. Other values negatively impacted the recognition rate and in some cases increased the logic utilization. The value of k was found to be less important but had a very specific range that was useful in terms of recognition rate. The k value was found to only weakly effect throughput and has no effect on logic utilization.

Chapter 5

Conclusions and Future Work

As the amount of recycling grows worldwide, the importance of fast and accurate sorting of recycled plastics increases, to maintain high purity without sacrificing throughput. This thesis addresses this problem by demonstrating a novel sorting algorithm that can correctly identify the resin type over 90% of the time, and is able to run in real time. It utilities a standard visible light camera and off the shelf components as oppose to expensive specialized equipment often found in current sorting machines.

The algorithm presented is able to sort two container types: PET and Polycoat containers. A linear support vector machine was trained to distinguish the two container types using the difference in their optical properties. Both reflectivity and transparency are used to distinguish the container, which is measured using the histogram of pixel intensities.

This work extends previous work in this area by allowing multiple containers to be imaged at the same time. By making use of novel segmentation algorithms, the method was shown to accurately identify container regions from background regions and find individual containers within the images. The background is removed by subdividing the image into small blocks and removing blocks that are found to be part of the background. The remaining blocks are grouped into regions based on their similarity. In addition the algorithm is implemented in an FPGA which allows for high frame rates. However, the process suffers a small degradation in overall accuracy compared to some previous work due to the extra segmentation that is performed.

The method was shown to have a peek identification rate of 93% while being able to run at 87 fps using images that were 640×480 pixels in size. This was accomplished by dividing the image into 8×8 blocks. A slightly less accurate version (92.9%) was demonstrated to be able to run at 127 fps by using 10×10 blocks An efficient FPGA implementation of the proposed method was also presented. Through heavy logic reuse, only 19% of the available adaptive logic modules of the device was required.

Several factors were examined that affect the operation characteristics of the algorithm. The largest effect was observed when adjusting the size of the blocks that the images were divided into. It was found that the accuracy suffers significantly by adjusting the block size, and only a few values produced usable results. In addition, block size has the largest effect on throughput. By subdividing the image into larger blocks, fewer overall blocks are present and the algorithm can run faster. It was determined that the relationship between throughput and block size is quadratic in nature, with speed increasing with the square of the block size.

The block size also impacts the hardware implementation where larger block sizes require slightly more LEs but significantly less RAM. While the LE usage is approximately constant, the RAM usage heavily depends on the block size. This again is due to the fact that the number of blocks grows quadratically by decreasing the block size.

The number of bins in the histogram was also found to heavily effect the recognition rate. The optimum number of bins was found to be 32 bins, corresponding to using the five most significant bits of the pixel data. Deviation from this value decreases the overall recognition rate. Increasing the bin size significantly increases the FPGA resources required to implement the design, and the opposite is true for reducing the number of bins.

The region merging parameter, k was also examined, though it was found to have a weaker effect on the algorithm. However, the value of k does have to be properly tuned relative to the chosen block size to achieve maximum recognition rate. As the blocks sizes is increased the value of k needs to decrease to stop regions from growing too large. The parameter k was found to have only a slight impact on the frame rate, though not nearly as significant as block size and k has no bearing on the hardware implementation.

5.1 Future Work

Future work will focus on extending this method to the classification of multiple plastic resins types and improving the recognition rate. This work could focus on making more direct measurement of material properties through the application of multispectral imaging. By directly observing the NIR spectra of the imaged material a more accurate segmentation could be obtained. Higher recognition rates would be obtainable by using the improved segmentation and spectra information in the classification process. A multiclass SVM could be trained using by combining the visible light data with NIR spectra to allow the classification of several common plastic resins.

Bibliography

A. Babooram, J. W. (2007). Recycling In Canada.

A. I. Belousov, S. A. Verzakov, J. v. F. (2002). Applicational aspects of support vector machines. *JOURNAL OF CHEMOMETRICS*, **16**, 482–489.

B. Scholkopf, A. S. (2002). *Risk and Loss Functions*, chapter 3, pages 61–86. The MIT Press.

Bruno, E. (2000). Automated Sorting of Plastics for Recycling.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2**(2), 121–167.

Canny, J. (1986). A computational approach to edge detection. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, number 6 in 8, pages 679–698.

Chanda, S., Terrades, O. R., and Pal, U. (2007). Svm based scheme for thai and english script identification. In *Proc. Ninth Int. Conf. Document Analysis and Recognition ICDAR 2007*, volume 1, pages 551–555.

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines.* Software available at urlhttp://www.csie.ntu.edu.tw/cjlin/libsvm.

Chen, D., Bourlard, H., and Thiran, J.-P. (2001). Text identification in complex background using svm. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition CVPR 2001*, volume 2.

D.M. Scott, R. W. (1995). Identification of Plastic Waste Using Spectroscopy and Neural Networks. *Polymer Engineering and Science*, **35**(12), 1011–1015.

Enviromental Protection Agency (2009). Municipal Solid Waste Generation, Recycling and Disposal in the United States: Facts and Figures for 2008. Obtained may 26th, 2010 from www.epa.gove/wastes/nonhaz/municipal/pubs/msw2008rpt.pdf.

Enviromental Protection Agency (2003). Municipal Solid Waste Generation, Recycling and Disposal in the United States: Facts and Figures for 2002. Obtained may 26th, 2010 from www.epa.gove/wastes/nonhaz/municipal/pubs/msw2003rpt.pdf.

EPIC (2008). A Review of Optical Technology to Sort Plastics and Other Containers. Technical report.

Eureka Recycling (2002). A Comperative Analysis of Applied Recycling Collection Methods in Saint Paul. Technical report.

Felzenszwalb, P. F. and Huttenlocher, D. P. (1998). Image segmentation using local variation. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, pages 98–104.

Goshorn, D., Cho, J., Kastner, R., and Mirzaei, S. (2010). Field programmable gate array implementation of parts-based object detection for real time video applications. In *Proc. Int Field Programmable Logic and Applications (FPL) Conf*, pages 582–587.

House, B., Capson, D., and Schuurman, D. (2011). Towards real time sorting of recyclable goods using an fpga based svm. In *Proc. of the 2011 IEEE International Symposium on Sustainable Systems and Technology (ISSST)*.

Hsu, C. (1997). *Infrared Spectroscopy*, chapter 15, pages 247–284. Prentice Hall.

MacLean, W. J. (2005). An evaluation of the suitability of fpgas for embedded vision systems. In *Proc. CVPR Workshops Computer Vision and Pattern Recognition - Workshops IEEE Computer Society Conf.*

Marshall, J. (2006). Households and the Environment. http://www.statcan.gc.ca/pub/11-526-x/11-526-x2007001-eng.pdf.

Morawski, C. (2002). An Overview of Plastic Bottle Recycling in Canada. Technical report, EPIC.

Morawski, C. (2009). Understanding Economic and Environmental Impacts of Single Stream Collection Systems. Technical report, Container Recycling Institute.

Nawrocky, M., Schuurman, D. C., and Fortuna, J. (2010). Visual sorting of recyclable goods using a support vector machine. In *Proc. 23rd Canadian Conf. Electrical and Computer Engineering (CCECE)*, pages 1–4.

R. Leitner, H. Mairer, K. K. (2003). Real-Time Classification of Polymers with NIR spectral imaging and Blob Anaylsis. *Real-Time Imaging*, **9**, 245–251.

Ramli, S., Mustafa, M. M., Hussain, A., and Wahab, D. A. (2007). Automatic detection of 'rois' for plastic bottle classification. In *Proc. 5th Student Conf. Research and Development SCOReD 2007*, pages 1–5.

Ramli, S., Mustafa, M. M., Hussain, A., and Wahab, D. A. (2008). Histogram of intensity feature extraction for automatic plasitc bottle recycling system using machine vision. *American Journal of Environmental sciences*, **4**, 583–588.

Ramli, S., Mustafa, M. M., Wahab, D. A., and Hussain, A. (2010). Plastic bottle shape classification using partial erosion-based approach. In *Proc. 6th Int Signal Processing and Its Applications (CSPA) Colloquium*, pages 1–4.

Sawyer, D. (2009). New Avenue in Recycling: NIR and Other technologie Sort PET and Bioresin BOttles. Technical report, NatureWorks LLC.

Selke, S. E. (2006). *PLASTICS RECYCLING AND BIODEGRADABLE PLASTICS*, chapter 8, pages 476–586. McGraw-Hill.

Shahbudin, S., Ramli, S., Mustafa, M. M., Hussain, A., and Wahab, D. A. (2010). Support vector machines for automated classification of plastic bottles. In *Signal Processing and Its Applications (CSPA), 2010 6th International Colloquium on*, pages 1–5.

Smith, R. (2009). Human Activity and the Environment:Annual Statistics. http://www.statcan.gc.ca/pub/16-201-x/16-201-x2009000-eng.pdf.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience, 1 edition.