February 2016

Feature Encoding Strategies for Multi-View Image Classification

Kyle Doerr
The University of Western Ontario

Supervisor
Dr. Samarabandu
The University of Western Ontario

Graduate Program in Electrical and Computer Engineering

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Engineering Science

Abstract

Machine vision systems can vary greatly in size and complexity depending on the task at hand. However, the purpose of inspection, quality and reliability remains the same. This work sets out to bridge the gap between traditional machine vision and computer vision. By applying powerful computer vision techniques, we are able to achieve more robust solutions in manufacturing settings. This thesis presents a framework for applying powerful new image classification techniques used for image retrieval in the Bag of Words (BoW) framework. In addition, an exhaustive evaluation of commonly used feature pooling approaches is conducted with results showing that spatial augmentation can outperform mean and max descriptor pooling on an in-house dataset and the CalTech 3D dataset. The results of the experiments contained within, details a framework that performs classification using multiple view points. The results show that the feature encoding method known as Triangulation Embedding outperforms the Vector of Locally Aggregated Descriptors (VLAD) and the standard BoW framework with an accuracy of 99.28%. This improvement is also seen on the public Caltech 3D dataset where the improvement over VLAD and BoW was 5.64% and 12.23% respectively. This proposed multiple view classification system is also robust enough to handle the real world problem of camera failure and still classify with a high reliability. A missing camera input was simulated and showed that using the Triangulation Embedding method, the system could perform classification with a minor reduction in accuracy at 98.89%, compared to the BoW baseline at 96.60% using the same techniques. The presented solution tackles the traditional machine vision problem of object identification and also allows for the training of a machine vision system that can be done without any expert level knowledge.

Keywords: Machine Vision, Manufacturing reliability, Triangulation embedding, Multi-view image classification, Multiple view vision system

Acknowledgements

I would first like to thank my supervisor, Dr. Samarabandu. I have been his student for several years now, and he has been such a positive influence during my time at Western. His patience, expertise and his open-mindedness have made this possible. His willingness to take a chance on me and help foster my education is something I will forever cherish. Thank you Professor.

Next I would like to thank Dr. Wang. He has been friendly and willing to cater to my educational needs. He welcomed me into his research lab to an amazing group of people who have become dear friends to me.

I want to thank the industry sponsor that supported this work, Sightline Innovation Inc. Specifically Wallace (Wally) Trenholm, Maithili Mavinkurve and Mark Alexiuk. Their visionary leadership and expertise have been crucial in solving industrial problems using machine learning.

I would like to thank my parents. Their hard work and belief in the importance of education has provided me with this opportunity. I would not have made it this far without their support. I would also like to my dear friends I have met during my time at Western. Through all of our years together we have watched each other grow. In no particular order, I would like to thank - Paria, Chris Line and S., Fuad, Tania, Guy, Veronik, Ken, Matt, Ashley, Justine, Jeremy and Colin. Thank you.

To my loving parents Mark and Cathy

Contents

Abstract **ii**

List of Figures **viii**

List of Tables **xi**

List of Abbreviations **xii**

1 Introduction **1**

1.1 Motivation 1

1.1.1 Problem Statement 3

1.2 SIFT, SURF, and ORB features 3

1.2.1 SIFT features 3

1.2.2 SURF features 5

1.2.3 ORB features 5

1.3 Thesis Contribution 6

1.4 Organization of thesis 7

1.5 Experiment Summary 8

2 Background and Literature Review **13**

2.1 Machine Learning 13

2.2 Image Processing, Understanding and Computer Vision 13

2.2.1 Image Representation 14

2.3 Feature Extraction 14

2.4 ORB Features 15

2.5 Bag of Words 21

2.5.1 Descriptor Quantization . . . 21
2.5.2 Histogram Encodings . . . 22
2.5.3 Spatial Pyramids and Pooling . . . 23
2.5.4 Support Vector Machines . . . 24
2.5.5 Kernels . . . 27
2.6 First Order Information Encoding . . . 28
2.7 Triangulation Embedding . . . 30
2.8 Power Normalization to Address Bursty Features . . . 33
2.9 Dimensionality Reduction . . . 33

3 Part Classification on a Conveyor Belt Framework 36
3.1 Physical Setup . . . 37
3.2 Object Localization . . . 38
3.3 Extension to Multiple Views . . . 42

4 Experimental Results 47
4.1 Introduction to Datasets Used in Experiments . . . 47
4.1.1 Introduction to In-House Dataset . . . 47
4.1.2 Introduction to Caltech 3D dataset . . . 50
4.2 Evaluation of Single and Multiple Views . . . 51
4.2.1 Classification Performance with a Single View . . . 51
4.2.2 Results . . . 51
4.3 Classification Performance with Multiple Views . . . 54
4.4 Evaluation of Multiple Dictionaries . . . 58
4.4.1 A codebook for each view . . . 58
4.4.2 A single codebook for all the views . . . 58
4.4.3 Summation of visual words . . . 59
4.4.4 Results . . . 59
4.5 Kernel Selection for SVM Optimization . . . 60
4.5.1 Traditional Kernels for use with an SVM . . . 60
4.5.2 Results . . . 61

4.5.3 Histogram Intersection and X^2 kernels . . . 64
Histogram Intersection . . . 64
$\mathbf{X^2}$ kernel . . . 66
4.5.4 Results . . . 66
4.6 Evaluation of Spatial Histograms and Pooling . . . 67
4.6.1 Spatial Histograms . . . 67
4.6.2 Feature Pooling . . . 69
Mean Pooling: . . . 69
Max Pooling . . . 69
Sum Pooling: . . . 69
4.6.3 Results . . . 70
4.7 Evaluation of Descriptor Encoding Strategies . . . 72
4.7.1 Results . . . 73
4.7.2 VLAD and Dimensionality Reduction . . . 75
4.8 Triangulation Embedding . . . 77
4.8.1 Triangulation Embedding Classification Results . . . 78
4.8.2 Triangulation Embedding analysis . . . 78
4.8.3 Triangulation Embedding, recovering from camera downtime . . . 80
4.9 Running Times of Feature Encoding Methods . . . 81

5 Conclusions and Future Work **83**
5.1 Summary . . . 83
5.2 Future Work . . . 85

Bibliography **86**

Curriculum Vitae **91**

List of Figures

1.1 Images of window pillars, guides and their location on a vehicle 3

1.2 Generation of SIFT Descriptors with scale and rotation invariance 4

1.3 The shrinking of the hull by erosion 6

2.1 A representation of a digital image 15

2.2 ORB features found in an image 16

2.3 A visual of the FAST corner test 17

2.4 Keypoints on edges will have little change along the direction of the edge. Gradients along the corner will have a large change in all directions 17

2.5 Features are produced at each level of the pyramid 19

2.6 A sample showing where the centroid is located within an image 20

2.7 k-means data clustering by use of a Voronoi diagram [37] 22

2.8 The process of descriptor encoding 23

2.9 Spatial Pyramid construction on images 24

2.10 A simple diagram of a Support Vector Machine with a separating plane in 2D . 25

2.11 The points lying closest to the separating hyperplane are deemed the support vectors 26

2.12 Data that becomes linearly separable when mapped to a higher dimensional space 27

2.13 VLAD stage 1: Assign a descriptor to a word (centroid) in the visual vocabulary by Nearest Neighbour, as in the traditional BoW. 28

2.14 VLAD stage 2: The residual vector of a single descriptor and its assigned centroid is a sub-vector which will then subsequently be aggregated. 29

2.15 Example of a bursty feature in a histogram 34

2.16 After applying power normalization to the histogram 34
2.17 PCA example on a 2D dataset after projection of the data onto the principal component axes 35

3.1 Positioned cameras focused over a conveyor belt 37
3.2 A diagram showing how different views can look drastically different depending on the view point 38
3.3 Mock conveyor used in lab to gather data. 39
3.4 The type of camera used in our experiments 40
3.5 The object localization process 41
3.6 The shrinking of the hull by erosion 42
3.7 Structuring elements note that element A fits whereas B and C do not. 43
3.8 Pinhole 44
3.9 Projection diagram 44
3.10 Lens used to focus light 45

4.1 Window pillars comparing the two types of lustre: matte and glossy 48
4.2 Window pillars comparing the longer and shorter window pillars for front and back placement 49
4.3 Window guides of varying length 50
4.4 Window pillars with left and right orientation 50
4.5 A diagram showing samples of the CalTech 3D dataset 51
4.6 Classification performance on in-house dataset using the BoW model comparing SIFT, SURF and ORB features against vocabulary size using a single view 52
4.7 Classification performance on CalTech dataset using the BoW model with varying vocabulary size using a single view 53
4.8 Classification performance showing the impacts of using different combinations of views and dictionary size on in-house dataset 54
4.9 Classification performance using multiple views of the CalTech 3D dataset . . 56
4.10 Confusion matrix of the CalTech 3D dataset single view with classification results averaged over 10-folds 57

4.11 Confusion matrix of the CalTech 3D dataset multiple views with classification results averaged over 10-folds 57
4.12 A comparison on using different methods of utilizing the visual vocabulary for image classification using in-house dataset 59
4.13 Heat Map of Grid Search for polynomial degree 2 parameters 62
4.14 Heat Map of Grid Search for polynomial degree 3 parameters 63
4.15 Heat Map of Grid Search for RBF kernel 64
4.16 Generating spatial pyramids [32] 65
4.17 An example of spatial histograms 68
4.18 Spatial histograms and feature pooling applied to in-house dataset 71
4.19 Spatial histograms and feature pooling applied to CalTech 3D dataset 72
4.20 Classification improvements when adding spatial histograms 73
4.21 The results of the in-house dataset using the VLAD encoding approach 74
4.22 Classification of Caltech using the VLAD encoding approach 75
4.23 In-House eigenvalue distribution of PCs on feature vectors using VLAD encodings 76
4.24 CalTech eigenvalue distribution of PCs on feature vectors using VLAD 77
4.25 Similarity scores histogram of Shoe vs. Iron classes using BoW 79
4.26 Similarity score histogram of Shoe vs. Iron classes using Triangulation Embedding 79

List of Tables

4.1 Comparison of classification time [11] 55
4.2 Hyper-parameter search for several kernels and the affects on classification . . . 62
4.3 Comparison of the Histogram Intersection and X^2 kernels 67
4.4 Effects of Dimensionality Reduction when using VLAD encodings 76
4.5 Comparison of the Triangulation Embedding method on the in-house dataset . 78
4.6 Comparison of the Triangulation Embedding method on the CalTech 3D set . . 78
4.7 Comparison of the Triangulation Embedding method on the in-house dataset . 80
4.8 Comparison of the Triangulation Embedding method on the CalTech dataset . 81
4.9 Comparison of the feature encoding rates 81

List of Abbreviations

BoW	Bag of Words
BRIEF	Binary Robust Independent Elementary Features
FAST	Features from Accelerated Segment Test
HIK	Histogram Intersection Kernel
ORB	Oriented FAST and Rotated BRIEF
PCA	Principal Component Analysis
PMK	Pyramid Match Kernel
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
SVD	Singular Value Decomposition
SVM	Support Vector Machine
VLAD	Vector of Locally Aggregated Descriptors

Chapter 1

Introduction

1.1 Motivation

Machine vision is the process of applying image processing and analysis to automate certain tasks. This term is typically applied to automating lower level tasks such as the reading of optical characters, defect inspection, and object counting. Computer vision on the other hand focuses on higher level tasks such as object and person classification, detection, tracking and segmentation. Typically, computer vision tasks use more complex methods and models for representing visual data. Often at that core of computer vision, these tasks are more difficult than machine vision because of variability in the input data and often the data comes from images that the input is not from a controlled environment. Trying to get a computer to interpret and make decisions given a wide variety of complex visual information like a human would is more challenging when the image or video is captured by a human and can contain variable poses and possible occlusions. This would not be the case for machine vision environments in which the camera is often in a fixed position and some knowledge of what to expect in the image would be known beforehand.

Machine vision will often use more low level techniques that have proven themselves to be useful in industrial settings but unable to handle the large variations in input data. Machine vision often requires a clear specified problem that can only be handled in a controlled environment resulting in input data that has less variance. For a product that uses machine vision for object verification and defect detection to have any use in an industrial setting it must provide

very high accuracy and always be more reliable and faster than human inspection. Depending on the application in computer vision detecting object defects can have a wide variability in accuracy depending on the application and technique used. For example, a system that can achieve a 5% misclassification rate may achieve state of the art results for a particular learning task and that may be acceptable, but in an industrial setting that can potentially process thousands of parts a day as say defective, this system may no longer be acceptable at fully automating a task.

Machine vision systems offer high accuracy, but are limited to their environment and often require very careful parameter tuning. Computer vision on the other hand has more freedom in the variability of the data and while often has parameters tuning, typically it is not as stringent with the parameter learning being treated sometimes as a learning task in itself; say using a validation set. Our problem that we try to answer is, if machine vision works well on easier, more well defined problems but not on challenging problems with greater variability in the input and computer vision techniques perform reasonably well on more challenging hard problems, is there a way to bridge the gap between the two and use the strengths of computer vision approaches in industrial applications?

In many manufacturing settings assembly lines use conveyor belts to move parts to different stages of production. In places that manufacture auto parts the products might have minimal variations in size, orientation, texture and luster. This is because each class of part is specific to the location of the car it is on and the part design between car models and the year of manufacture often varies. Many manufacturers employ people to manually inspect these parts but given the level of similarity among the parts moving on a belt the current approach is prone to human error. Correct part classification is essential for quality control and has financial implications for automotive manufacturers when high standards are not met. In this work we propose a low cost framework to address this problem of similar auto part classification using techniques from computer vision.

Our specific problem was to perform classification on window guides and pillars. The functionality of these parts is just that, to guide windows as they move up and down. The vertical parts are called pillars and the horizontal parts are called window guides. Samples of pillars and guides and where they fit on a vehicle can be seen in Figure 1.1. Since the

parts come in different combinations of shapes, sizes and orientations we opted to use multiple cameras angled downwards so that differences among similar parts could be visually captured. More information on these parts can be found in Chapter 3.

1.1.1 Problem Statement

At the time of research, these automotive parts were previously categorized by humans during the manufacturing process. Since there are several stages during the production process, the parts could undergo chemical baths and painting as intermediate processes. Because of these stages, appending simple labels for identification in the form of barcodes is not possible. Using humans to visually categorize these parts and then sort them, resulted in frequent mislabelling that created confusion for the manufacturers during the assembly stage.

Figure 1.1: Images of window pillars, guides[1] and their location on a vehicle

1.2 SIFT, SURF, and ORB features

1.2.1 SIFT features

Scale-Invariant Feature Transform (SIFT)[34], Speeded-Up Robust Features (SURF)[1], and Oriented FAST and Rotated BRIEF (ORB)[43] are all methods for generating a collection of local image features. These local image features are designed to be invariant to translation, scaling and rotation. In other words, these local features can be reliably matched to a new set of image features, even if the new image has undergone one or all of these transformations.

[1] https://www.ecstuning.com

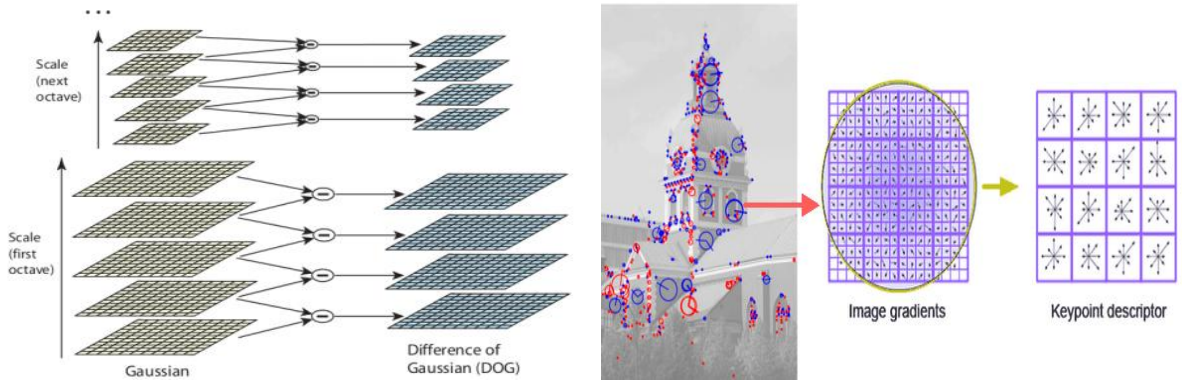
The SIFT algorithm has been one of the most popular and widely used algorithms for detecting and describing image features over the past 15 years. The input image is repeatedly blurred and downsampled to create a pyramid. The differences between successive layers in the pyramid is computed to form a subsequent pyramid known as a Difference of Gaussian (DoG) pyramid. Local extrema at both the space neighbourhood and scale neighbourhood of are marked as keypoints.

The keypoints correspond to an edge or a corner. Keypoints that lie on corners are more desirable as they are more discriminative as explained in Section 2.4. The principal curvatures of the keypoint are encoded by the Hessian matrix:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{1.1}$$

The ratio of the two eigenvalues of the Hessian matrix gives insight as to whether the keypoint resembles more like a corner or edge. The keypoints that are too similar to edges are removed.

The step after rejecting keypoints is to make the local feature rotationally invariant. Around each keypoint, a histogram containing weighted orientations of local gradients within a circular grid is computed. The dominant gradient becomes the canonical orientation in which the patch is rotated respectively as seen in Figure 1.2b



(a) Difference of Gaussian Pyramid[2] (b) SIFT Descriptors from keypoints[3]

Figure 1.2: Generation of SIFT Descriptors with scale and rotation invariance

[2]http://aishack.in/tutorials/sift-scale-invariant-feature-transform-log-approximation
[3]http://www.codeproject.com/Articles/619039/Bag-of-Features-Descriptor-on-SIFT-Features-with-O

1.2.2 SURF features

The SURF algorithm, like SIFT was based on the same principals of generating local image features that are invariant to translation, scaling and rotation. While SIFT features are quite robust, they are also slow to compute because of some computationally intensive tasks such as repeatedly performing convolution with the Gaussian kernel then downsampling to build the DoG pyramid. SURF uses a different approach to detect keypoints by using a Hessian based detector with the Laplace of Gaussian (LoG) approximation, given by:

$$\mathbf{H}(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{1.2}$$

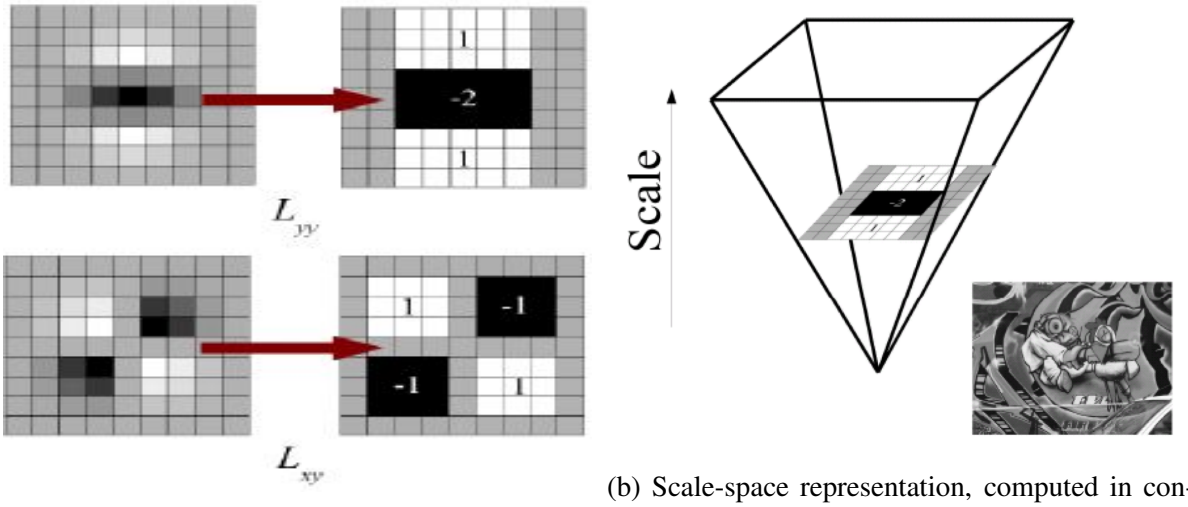
The SURF approach is then to approximate the LoG using box filters. SIFT has a scale space representation by repeatedly performing blurring and subsampling the input image then computing the DoG pyramid. SURF achieves scale invariance by increasing the box filter size at each level of the pyramid. Using the integral image representation initially introduced by Viola and Jones [48], convolution with a box filter can be performed in constant time $O(1)$ regardless of the filter size.

The determinant of Hessian $det(\mathbf{H})$ is computed with non-maximum suppression over the space and neighbouring scales. The resulting maxima that are left are identified as keypoints.

The Haar like filters similar to the scale representation step are used to compute the dominant gradient in the x & y direction in a circular region. The feature descriptor is built using 4x4 square subregions over the interest point. Gradient and magnitude information are used to build a 64 dimensional feature vector.

1.2.3 ORB features

ORB features are based on principles of SIFT and SURF features and are explained in much more detail in Section 2.4. The feature descriptors are built from sampling patterns precomputed to have a large variance. These sampling patterns are oriented by directional information and build a binary string that allows for fast computation and comparison to other ORB features. This brand of binary features are ideal for embedded applications and low powered



(a) Box filters used in feature detection[4]

(b) Scale-space representation, computed in constant time[5]

Figure 1.3: The shrinking of the hull by erosion

devices with limited memory.

1.3 Thesis Contribution

The contributions of this thesis are as follows:

- A comparison of faster, more efficient computer vision features for the purpose of machine vision are evaluated and compared to traditional feature types and the pros and cons of both are evaluated. The results obtained show that binary ORB features are more suitable than SIFT and SURF features for machine vision applications in controlled environments.

- A demonstration that using more advanced feature encoding techniques that contain higher order information, for the purpose of fine-grained image classification with multiple views has an improved classification accuracy for a machine vision dataset. These results also generalize to the publicly available CalTech 3D dataset.

- A new object classification framework using binary ORB features that can classify very similar automotive parts real-time and provide a meaningful, robust solution for

[4]http://docs.opencv.org/3.0-rc1/df/dd2/tutorial_py_surf_intro.html
[5]http://www.fhhyc.com/understanding-surf-features-calculation-process

manufacturing settings.

- A multi view machine vision system that is robust enough that it can handle missing views with a minimal decrease in classification performance. These benefits are demonstrated in the experimental results and show that the multiple view approach can still provide a robust solution despite camera failure.

1.4 Organization of thesis

This thesis is organized as follows:

- Chapter Two: The challenges associated with computer vision are discussed along with a literature review. Later in the chapter a review about one of the most popular image classification and retrieval techniques and its advancements in the past few years. A brief overview of the machine learning techniques used are also discussed and how they can be used for vision systems.

- Chapter Three: This chapter talks specifically about the engineering problem we set out to solve. The chapter then discusses the physical prototyping environment we used to mimic a real world vision system. We also mention the image preprocessing techniques we used prior to classifying the images. We finally discuss the improvement to the vision system by including multiple views.

- Chapter Four: There is discussion in this chapter about the in-house dataset that we created. We also discuss the use of a publicly available multi view dataset. We then provide a series of classification experiments on both datasets. We provide an assessment on using multiple views in a vision system and various classification techniques. Experimental works and results are presented along with a discussion of results.

- Chapter Five: An overall summary of this thesis is provided. Ideas and concepts learned from this work are presented. Areas for future work are also discussed.

1.5 Experiment Summary

All of the experiments conducted were evaluated on an in-house dataset containing automotive parts as introduced in Section 4.1.1 and the publicly available CalTech 3D dataset presented in Section 4.1.2.

The first set of experiments in Section 4.2.1: *Classification Performance with a Single View*, evaluated image classification accuracy by the use of single view images of objects within the Bag of Words (BoW) pipeline. These experiments set the baseline for classification accuracy and are compared against further experiments. The results on the first set of experiments show that for single view image classification, accuracy is generally proportional to dictionary size. This can be seen on the in-house dataset in Figure 4.6 and also on the CalTech dataset in Figure 4.7. The figures show that classification accuracy received a boost in performance for larger dictionary sizes. These two experiments also compared a newer binary descriptor against the traditional SIFT descriptor.

The next section of experiments in Section 4.3: *Classification Performance with Multiple Views*, goes from image classification using a single view to using multiple views on the in-house dataset. The motivation of this experiment was to determine if adding additional views would improve classification performance enough to justify adding more cameras in the framework. First, an evaluation of various view points was performed, with the results presented in Figure 4.8. The results show that using the four view points had a substantial increase in classification accuracy. Another trend that was observed, was that increasing the dictionary size improved performance to 95.62% (C=256), much like the case of using a single view. This is the first time to our knowledge that binary features have been applied to multi view image classification within the standard BoW pipeline. In both the single and multiple view experiments using ORB features resulted in a higher classification accuracy when compared to SIFT or SURF features.

Following this, the same BoW framework was applied to the CalTech dataset to see how previous findings would generalize to more typical computer vision type datasets. Mirroring the previous experiment of multi view image classification with the CalTech dataset, the classification also performed better as function of dictionary size as seen in Figure 4.9. The major

finding from this experiment was that using SIFT features resulted in a higher classification performance compared to ORB features on the CalTech dataset for single and multi view classification tasks, this was not the case on the in-house dataset. One of the major differences between the two datasets, is that the CalTech set had large changes in height and scale, whereas the in-house dataset, the cameras were stationary. It is known that ORB features are fast to compute when compared to SIFT as seen in Table 4.1, but SIFT performs better at matching features at different heights and scales. These results reveal that ORB features, in addition to being much faster to compute, can outperform competing techniques when there are not large changes in height and scale, which for this type of industrial vision system would often be the case.

Using ORB features as the baseline, there were significant improvements in classification performance extending from 1 to 8 views as seen in the confusion matrices shown in Figure 4.10 and Figure 4.11 that led to a mean classification accuracy improvement over the 8 object classes by 9.03%. We were able to show that ORB features are ideal when working with datasets more common in machine vision. This is something that limited prior research has been done, comparing traditional features with binary features beyond the traditional feature matching tasks.

The next set of experiments in Section 4.4: *Evaluation of Multiple Dictionaries*, evaluates how to best represent the centroids that make up the dictionary. One of the natural questions within the BoW pipeline that has been extended to multiple views is how to represent the centroids. This set of experiments compares three different methodologies: A codebook for each view (4.4.1), A single codebook for all the views (4.4.2), and summing the visual words across views (4.4.3). This is an important experiment because how the choice of representing the image as a feature vector is crucial for classification. The results of this experiment are shown in Figure 4.12, and they indicate that using one or four vocabularies for each view were comparable. Using the summation of visual words across views while faster, performed the least favourably. Since the results were nearly the same between the one and four dictionaries we opted to use one dictionary so there would not be the requirement to learn four vocabularies off-line. The results contained within this section show an empirical evaluation of several vocabulary representations.

From the previous experiments we had established the multiple view BoW framework performing classification with a Support Vector Machine (SVM). Classification using a SVM is a powerful tool and is often considered one of the first options when choosing a classifier. The next set of experiments in Section 4.5.1: *Kernel Selection for SVM optimization*, was to optimize the classifier by studying various kernels. The goal was to see if the data was linearly separable and to what degree. If it is not, then using kernels would increase the likelihood of separability in a higher dimensional space. The results presented in Table 4.2 show that the Gaussian kernel could actually perform the best with an accuracy rate of 97.8%. The more recent similarity functions known as the Histogram Intersection and the X^2 which are also functions, were also introduced in Section 4.5.3. The results from these two kernels as presented in Table 4.3 show a considerable increase in classification accuracy of 98.37% and 98.43%. Using the linear kernel as a classification baseline which obtained an accuracy of 95.62%, both of these kernels performed better. In fact, they even performed slightly better then the powerful Gaussian Kernel. It was demonstrated in this section that despite the elegant theory behind kernel methods, that the best performing kernel for a multiple view classification problem turned out to be the X^2 additive kernel. This kernel has two key advantages: it is fast to compute and unlike the traditional SVM kernels there are no hyper parameters that have to be determined thus making it ideal for smaller datasets.

The loss of spatial information that occurs within the BoW framework was addressed using feature pooling techniques is presented in Section 4.6.2. Spatial augmentation and pooling, using: max mean and sum pooling was evaluated and compared to the BoW baseline. The results on both datasets are shown in Figure 4.18 and Figure 4.19. On both datasets using spatial augmentation was the best and outperformed the BoW baseline with an accuracy of 96.08% on the in-house dataset and 93.03 % on the CalTech dataset using the basic linear kernel. The contributions to this experiment is an empirical evaluation of feature pooling and spatial binning techniques applied to a multiple view classification problem.

The next step of experiments was to evaluate descriptor encoding strategies using the feature encoding method Vectors of Locally Aggregated Descriptors (VLAD). In other literature this method of feature encoding has shown to provide excellent retrieval results. This method is evaluated on both datasets presented in Section 4.7.1. The in-house dataset evaluation is

shown in Figure 4.21 and gave a classification accuracy of 98.76% and the CalTech dataset in Figure 4.22 with 88.74%. Both of the results reveal a considerable improvement over the standard BoW baseline. Despite the popularity of VLAD feature encodings, very little work until now has been done to see how well this method works when using multiple views and very limited research has been performed to see how well this technique works outside of the traditional single view classification and retrieval problem.

The final encoding strategy evaluated is presented in Section 2.7: *Triangulation Embedding*. Triangulation Embedding maintains the BoW pipeline, yet on several occasions has given state of the art results for the task of image retrieval. Very little research is available on using this technique outside of the image search domain. Using the in-house dataset, the results of this embedding method can be seen in Table 4.5, which gave the best accuracy over all the other techniques at 99.28%. This considerable increase in classification accuracy also generalized to the CalTech dataset, where the results are presented in Table 4.6, showing an accuracy of 94.38%. There are several contributions contained within this section. Firstly, to the best of our knowledge, Triangulation Embedding has never be applied to a multiple view classification task. The next contribution is we have demonstrated that this powerful method is compatible with binary features, which has the added benefit of being very fast to compute. Thirdly, we were able to demonstrate this powerful technique can be used outside of the computer vision domain and can be used reliably for industrial machine vision applications. By doing this, we have demonstrated a way to help bridge the gap between the two fields of computer vision and machine vision, something this thesis set out to do.

The final set of experiments evaluate the ability of this framework to handle random camera downtime was also evaluated and compared against the original BoW. This section of experiments is located in Section 4.8.3: *Triangulation Embedding, recovering from camera downtime*. When compared to the original BoW, the Triangulation Embedding method is robust enough to withstand a randomly missing input and have a minimal reduction in classification accuracy. Equipment in factory settings is very susceptible to damage or forced to undergo unscheduled repairs because of the harsh conditions that are present in factories. Most industrial vision systems on the market today cannot operate if one of the sensors is damaged, which poses a huge risk for the manufacturer that purchased the vision system, as they run the risk

of having to halt production to keep up quality control. This set of experiments sees how well the proposed system is at classifying, when any one of the several cameras is disabled when capturing an image set. This experiment was carried out by randomly selecting a view for every image set taken and discarding it. For the in-house dataset this method proved to be quite robust and achieved an accuracy of 98.89% as seen in Table 4.7 for the in-house data and an accuracy of 91.94% shown in Table 4.8 for the CalTech dataset. The results demonstrated by this set of experiments, shows that this machine vision system used in conjunction with Triangulation embedding, can provide a robust solution that can still remain operational and useful for a business when one of the camera inputs is unavailable.

Chapter 2

Background and Literature Review

2.1 Machine Learning

Machine learning is a central topic in artificial intelligence and its definition is to have a machine learn from experience of a given task if a given performance measure can be improved from prior experiences [36]. The goal of learning is achieved using concepts from computer science, mathematics and statistics.

There are three main branches in machine learning: reinforcement, unsupervised and supervised learning. Reinforcement learning is when an agent is not told what action to take but learns from the feedback it receives after taking an action. At a high level it can be said that the agent attempts to maximize its rewards after making decisions. Unsupervised learning focuses on finding some structure using unlabelled data. The focus of this work is on supervised learning. In which a learning algorithm is given some labelled input and output data and must recognize a pattern and create a mapping function between the input and output [44]. Among all three learning approaches there is a tremendous amount of applications including image and text classification, medical diagnostics, speech recognition and data mining to name a few.

2.2 Image Processing, Understanding and Computer Vision

Digital image processing as defined by [16] is the activity of processing digital images using a digital computer. Image processing is the field of taking an image and applying algorithms

on the image to return an altered image with some desirable characteristics. Some examples could be performing image enhancement, compression, noise reduction and feature detection to name a few.

Image understanding tries to make sense of what is contained within the image. Image understanding is often seen as a mid-level process that relies on the output of a processed image and is more towards artificial intelligence. Examples of image understanding would be the task of image classification or segmenting an image and labelling the regions.

Computer vision on the other-hand is the closest towards artificial intelligence. The objective being seeing as humans do and having the ability to learn, and make decisions from visual information. An example would be what semantic information can be applied to a series of recognizable objects in an image. While Gonzalez and Woods [16] admits that there is no universally clear definition to distinguish between the three areas they are often thought of as low, mid, and high level processing. The work in this thesis primarily focuses on mid-level processing.

2.2.1 Image Representation

Following the definition of a digital image by Gonzalez and Woods [16], they define a gray scale image as a two dimensional function $f(x, y)$. The values of f are discrete quantities representing the gray scale intensity and x, y corresponding to a spatial location on the image as seen in Figure 2.1. The standard colour image is a vector valued function consisting of three channels: red, green and blue. Each channel holds the intensity of the colour at that particular pixel and when combined forms a colour image.

2.3 Feature Extraction

In the field of computer vision, an image feature is something of interest in a digital image. It is often, but not always a corner or an edge. It stands out as something interesting that is in the digital image. These are often referred to as ‘interest points’. The process of finding these interest points is commonly referred to as feature detection. Once a point or region of interest is found in the image, the next natural step is to find suitable description for this interest

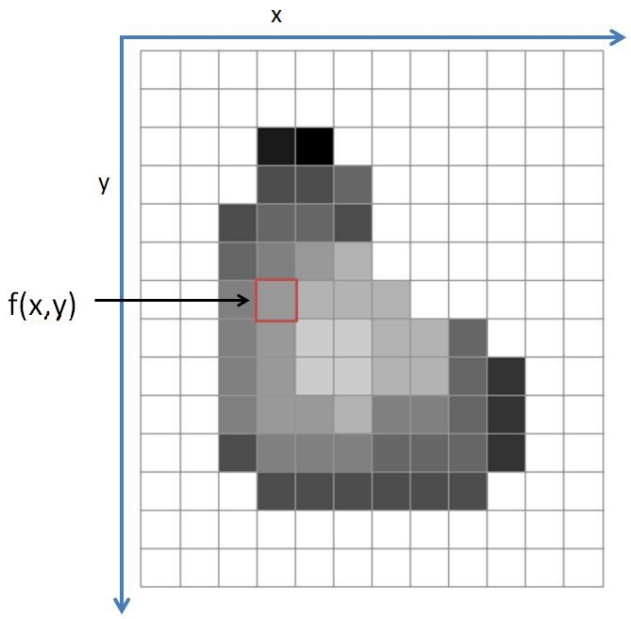


Figure 2.1: A representation of a digital image

point. There are numerous ways to describe these interest points such as colour, orientation, or intensity to name a few. These are referred to as feature descriptors.

2.4 ORB Features

As mentioned earlier the task of feature extraction is two step: detection and description. The most common feature type in computer vision is Scale-Invariant Feature Transform (SIFT) features. SIFT features, developed by David Lowe [34] have many good properties such as scale invariance, but SIFT descriptors are costly to compute. A new trend in the development of feature types that have emerged are known as binary image descriptors because they are faster to compute and match and have smaller memory requirements making them suitable for embedded devices. Rublee et al. [43] developed a method called ORB (Oriented FAST and Rotated BRIEF). The interest point detector uses the FAST (Features from Accelerated Segment Test) method [42] and a modified version of BRIEF as the feature descriptor [4].

The FAST keypoint detector identifies a pixel as being a corner by taking the image intensity at point p and comparing it to a circular ring of n connected pixels. Given the intensity at

Figure 2.2: ORB features found in an image

the candidate pixel I_p, if the n connected pixels are greater or less than a given threshold t the point can be classified as a potential corner. This method exploits the contiguous constraint on the pixel regions to speed up performance by only performing the intensity difference test at the compass directions, thereby avoiding needless computations, if the pixel does not meet the corner criteria as can be seen in Figure 2.3.

The FAST keypoint detector can also identify keypoints that lie along edges in addition to corners. Intuitively, corners have gradients in all directions, whereas an edge will have minimal gradients along the edge direction. It is worth pointing out that having gradients that vary in all directions is a desirable property because it causes the feature to have more variance. This leads to more discriminative features because that feature will respond differently and have a greater response to varying inputs. This can be seen in Figure2.4

The typical approach is to find many potential keypoints, then rank the keypoints based on a measure of how much of a corner it is. This measure is called the Harris corner measure [19]. Given a windowed region w centered at the location of the keypoint as $f(u, v)$ compute the change E given by a shift (x, y) as:

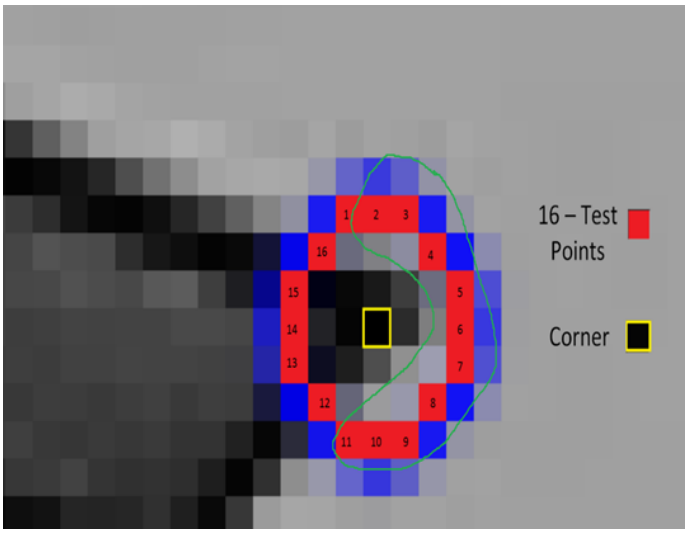


Figure 2.3: A visual of the FAST corner test

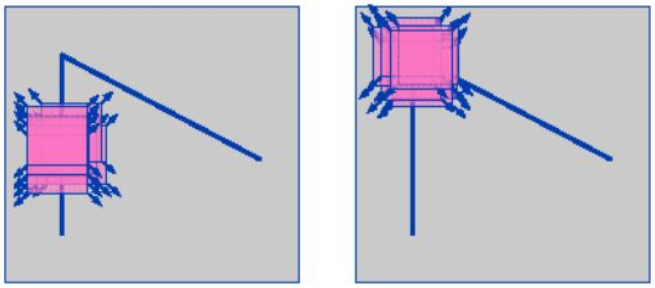


Figure 2.4: Keypoints on edges will have little change along the direction of the edge. Gradients along the corner will have a large change in all directions.[1]

$$E_{x,y} = \sum_{u,v} w_{u,v}(f_{(u+x,v+y)} - f_{(u,v)})^2 \tag{2.1}$$

Where $W \epsilon [0, 1]$ is the window function that is set to 1 for overlap and 0 otherwise.

Using the first order Taylor expansion to approximate a function, let ∇f_x and ∇f_y be the partial derivatives in the x-direction and y-direction, then:

$$f(u + x, v + y) \approx f(u, v) + x\nabla f_x + y\nabla f_y \tag{2.2}$$

[1] http://www.cse.psu.edu/ rtc12/CSE486/

Substituting equation 2.2 into equation 2.1 then:

$$E_{x,y} \approx \sum_{u,v} w_{u,v}(x\nabla f_x + y\nabla f_y)^2 \tag{2.3}$$

Because neighbouring pixels inside the circle may also meet the criteria for being a corner, the authors of ORB use the Harris corner measure [19] to rank the corners. Using 2.3 in matrix form:

$$E_{x,y} \approx (x, y)\, A \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.4}$$

Where A known as the Harris Matrix which is the covariance matrix of the partial derivatives of the windowed region defined as:

$$A = \sum_{u,v} w_{u,v} \begin{bmatrix} \nabla f_x^2 & \nabla f_x \nabla f_y \\ \nabla f_x \nabla f_y & \nabla f_y^2 \end{bmatrix} \tag{2.5}$$

As mentioned earlier corners have gradients that vary in all directions for minor shifts whereas edges are relatively flat along the direction of the edge. The eigenvalues λ_1 and λ_2 of A capture the amount of covariance. If $\lambda_1 > \lambda_2$ would mean that most of the gradient is captured in one direction. If $\lambda_1 \approx \lambda_2$ then the gradient variance is spread along both the x and y axes which follows the definition above that a corner has gradients in all directions for small shifts whereas an edge only has strong gradients on the orthogonal axis against the edge direction - rather most of the gradients can be captured along one axis. The corner response is efficiently calculated based on these two eigenvalues.

An interesting note about keypoints is that inherently no information about their scale is taken into consideration. That means if a feature was found in one image on an object at some scale and then found again in a separate image on an object at a different scale it makes the feature matching process less reliable. In many applications it would be desirable to match features regardless of scale. Moving towards scale invariance the corners are detected at multiple levels using the image pyramid approach. The detected corners and the corresponding scale is

stored and the desired number of keypoints, now ranked are selected.

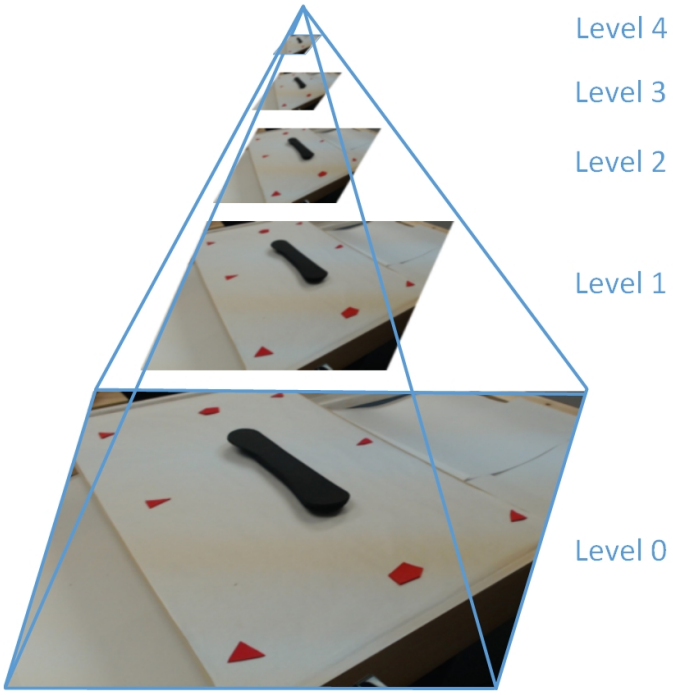


Figure 2.5: Features are produced at each level of the pyramid

Keypoints are commonly described by their orientation. In the case of corner based interest points, they can be thought of as what angle that corner points to. The FAST method alone does not provide this orientation information, but rather computed by calculating the intensity centroid as proposed by Rosin [41]. In this method the moment of an image patch is defined as:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{2.6}$$

The centroid location $C(x,y)$ can then be computed by:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{2.7}$$

By constructing a simple vector from the corner to the centroid the angle θ can be computed as:

$$\theta = tan^{-1}(m_{01}, m_{10}) \tag{2.8}$$

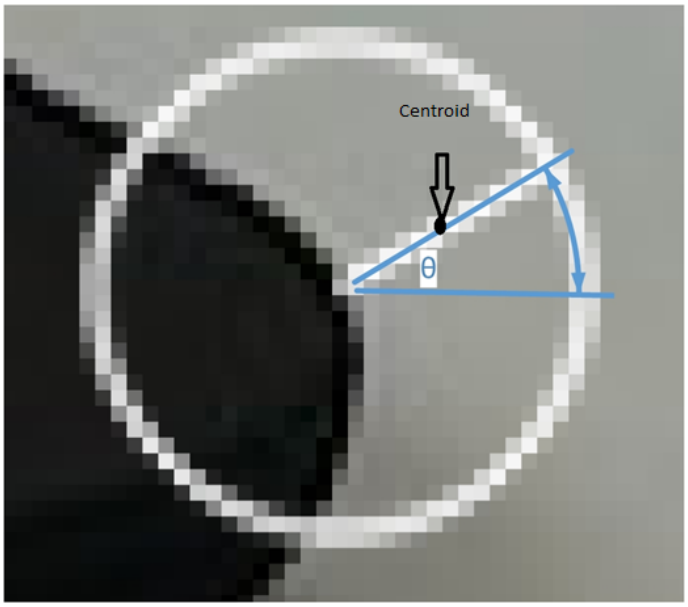


Figure 2.6: A sample showing where the centroid is located within an image

Following this approach the authors of ORB offer a modified version of the BRIEF feature descriptor [4]. The feature descriptor is a bit string on a smoothed image patch $\mathbf{p}$. The bit string is formed by constructing a series of binary tests τ defined by:

$$\tau(\mathbf{p}; \mathbf{x}, \mathbf{y}) = \begin{cases} 1 : \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 : \mathbf{p}(\mathbf{x}) \geq \mathbf{p}(\mathbf{y}) \end{cases} \tag{2.9}$$

The test results are defined by intensity of point $\mathbf{p}$ at $\mathbf{x}$. The final descriptor then is built using:

$$f_n(\mathbf{p}) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(\mathbf{p}; \mathbf{x}_i, \mathbf{y}_i) \tag{2.10}$$

Calonder et al. [4] observed that the selection of points falling under a Gaussian distribution around the center of the patch to perform the best. Rublee et al. [43] opted to go with this Gaussian sampling of points and a vector length of $n = 256$. Smoothing of the image patch is done prior to the sampling. Test points by default are 5 x 5 sub-window of a 31 x 31 pixel patch. At this point, the binary tests do not take into account the orientation information nor are they capable of being invariant to in-plane rotations. The binary tests are rotated based on the

orientation of the computed intensity centroid. This is done by taking a selection of n binary tests at location (x_i, y_i) defining a 2 x n matrix:

$$\mathbf{S} = \begin{pmatrix} x_1, ..., x_n \\ y_1, ..., y_n \end{pmatrix} \tag{2.11}$$

A rotation matrix in $\mathbb{R}^2$ is created from the patch orientation θ. The orientation is discretized by $\frac{2\pi}{30}$ increments. The rotation matrix is defined as:

$$\mathbf{R}_\theta = \begin{bmatrix} cos\ \theta & -sin\ \theta \\ sin\ \theta & cos\ \theta \end{bmatrix} \tag{2.12}$$

The rotation matrix is used to rotate the location of the binary patch tests in 2.10. The series of binary tests is now:

$$\mathbf{S}_\theta = \mathbf{R}_\theta \mathbf{S} \tag{2.13}$$

The rotated BRIEF operator is then defined as:

$$g_n(\mathbf{p}, \theta) := f_n(\mathbf{p}) | (x_i, y_i) \epsilon\ \mathbf{S}_\theta \tag{2.14}$$

2.5 Bag of Words

2.5.1 Descriptor Quantization

Descriptor quantization is common among all BoW and extended BoW models. This process is often done through clustering a set of feature descriptors to form what is known as the *visual words*. All the visual words or clusters are known as the *visual vocabulary*. The most popular clustering algorithm is known as *k-means* which was first proposed by James MacQueen [35]. One of the earliest mentions of constructing the visual vocabulary using the k-means approach for object categorization is given by Csurka et al.[9]. A set of n feature descriptors can be described as: $x_1, ..., x_n \epsilon\ \mathbb{R}^D$. In building the visual vocabulary the descriptor space is partitioned using k-means with k vectors being the centroids, described as: $\mu_1, ..., \mu_k \epsilon\ \mathbb{R}^D$. Each descriptor

has a data to means assignments: $q_1, ..., q_n \ \epsilon \ \{1, ..., K\}$. The centroids are found using the objective function, effectively minimizing the l^2 error:

$$q_{ki} = \underset{k}{\text{argmin}} \ \|x_i - \mu_k\|^2 \tag{2.15}$$

Unfortunately with this method the quantization of words is done by vector quantization (or hard quantization) there is no obvious choice for the number of visual words (centroids) and often needs to be determined empirically.

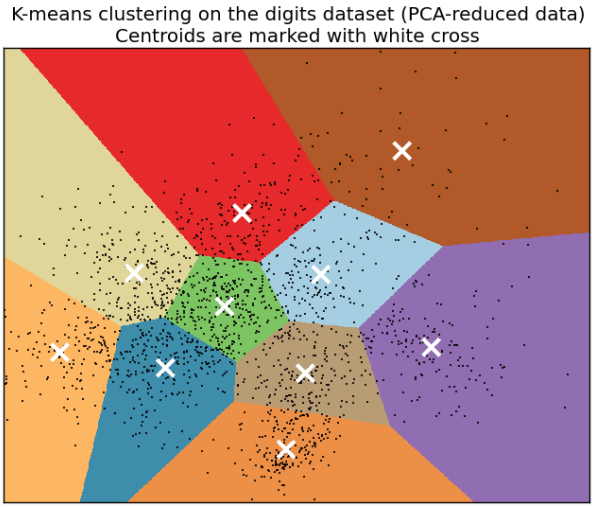


Figure 2.7: k-means data clustering by use of a Voronoi diagram [37]

2.5.2 Histogram Encodings

One of the natural issues that is brought up in the image retrieval or classification task is how to meaningfully represent the image in such a way that it can be compared to other images. One idea of representing the images is to use a global histogram of intensity values. The concept of using histograms of words faces some challenges that histograms of intensity values may not give any information about what is contained in the image. Images that share no relationships may have very similar histograms. One approach to circumvent this problem is to use a histogram of quantized local descriptors from a previously learned visual vocabulary. Chatfield [6] gives an excellent overview of this approach. The idea is to build the visual

vocabulary as a preprocessing step. Afterwards on each new query or training image recompute the image descriptors $x \epsilon \mathcal{X}$ then assign each descriptor to a visual word by Equation 2.15. The resulting query image is then represented as a $1 \times n$ vector with n being the number of centroids learned by k-means. This histogram representation is a type of feature vector which can then be used for retrieval or image classification purposes.

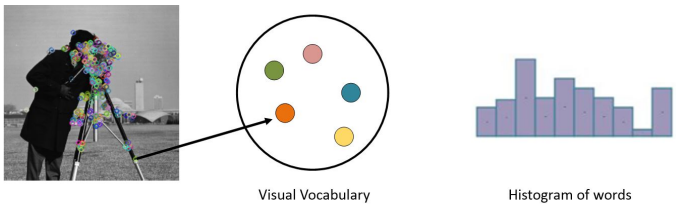


Figure 2.8: The process of descriptor encoding

2.5.3 Spatial Pyramids and Pooling

One of the limitations of the BoW model is that it is an orderless representation of encoded keypoints found on a digital image. The geometric and spatial information can reveal a lot about the scene especially when it comes to image classification. This information is lost. One method for overcoming this proposed by Lazebnik et al. [32] is to partition the image into increasing fine sub-regions and computing a histogram per region. This work was adapted from Grauman and Darrell [18] in which the authors proposed the *pyramid match kernel* (PMK). The PMK is a function that can compare histograms of image descriptors at increasingly coarser grids as seen in Figure 2.9. Each successive level in the pyramid has an increased weight applied to the histogram I^l defined as:

$$I^l = \frac{1}{2^{L-l}} \quad for\ l = 0, ..., L-1 \tag{2.16}$$

It was suggested by Graunman and Darrell [18] to use the histogram intersection function to compare histograms, as they showed it was a Mercer kernel and thus suitable for use in a support vector machine (described later). In the works done by [32] and [18] it is worth noting that increasing the number of levels in the pyramid improves recognition performance up to

a point and then starts to decrease. There are also variations in the construction of the spatial grids of *m*x*m* but *m*x*n*.

Conversely a more simple approach to incorporate weak geometry is to simply concatenate all the histograms derived from each spatial region. The one obvious problem is that simple concatenation of each histogram per spatial region causes the feature vector to become quite large. One way to circumvent this is to perform sum, max or average pooling. In which each bin of the histogram per level of the pyramid is aggregated by the sum, max or average response.

Figure 2.9: Spatial Pyramid construction on images

Choosing the number of levels in the pyramid or the partition layout remains an open question and is often found on a validation set. One thing to keep in mind is that classification performance will plateau then start to decrease as the number of grids grows simply because there will be fewer descriptors found in each cell. This will start to cause sparsity in the feature vector and corrupt the image similarity metric.

2.5.4 Support Vector Machines

Support vector machines (SVMs) are a very useful tool in machine learning. Developed by Vapnik and Cortes [7] from statistical learning theory, it is a powerful method of supervised learning. Starting off with discriminant analysis, assuming a set of data is linearly separable; that is, a hyperplane can be drawn that separates data given a set of inputs $x = [x_1, x_2, ..., x_n]$ a set of weights $w = [w_1, w_2, ..., w_n]$ can be established to form a discriminant:

$$g(x) = w^T x + w_0 \tag{2.17}$$

To find a decision rule:

$$g(x) = \begin{cases} Class\ 1,\ g(x) > 0 \\ Class\ 2,\ g(x) < 0 \end{cases} \tag{2.18}$$

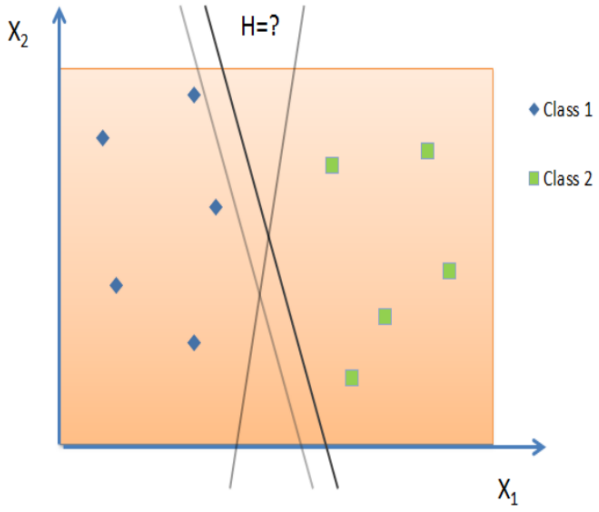


Figure 2.10: A simple diagram of a Support Vector Machine with a separating plane in 2D

In Figure 2.10 the example is clearly linearly separable with the separating hyperplane said to be the discriminant. The problem arises in finding the ideal location of the discriminant function that will generalize well on new data by finding the ideal weights with respect to the given training data.

The idea is to write the discriminant as a linear combination of the data that lies closest to the separating hyperplane. These data points closest to the discriminant are known as support vectors. Given a training set of data for a binary classification problem:

$$y_i = \begin{cases} -1,\ Class\ 1 \\ +1,\ Class\ 2 \end{cases} \tag{2.19}$$

With the condition:

$$y_i(w^T x + w_0) \geq b,\ \ b > 0 \tag{2.20}$$

Constructing two canonical hyperplanes passing through the support vectors on both sides

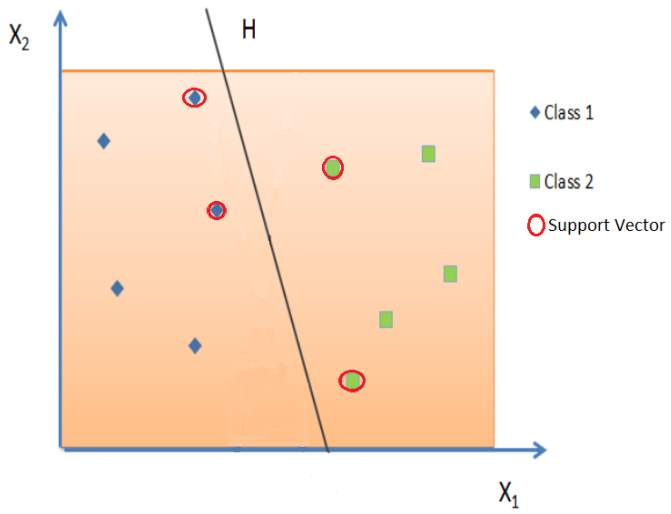


Figure 2.11: The points lying closest to the separating hyperplane are deemed the support vectors

of the discriminant defines the margin m. The margin can be written as $m = \frac{2}{|W|}$. The problem can be reformulated as:

$$\begin{aligned} &Minimize\ |w|\ subject\ to: \\ &y_i(w^T x + w_0) \geq b \end{aligned} \tag{2.21}$$

In other words, minimizing the $|w|$ translates to maximizing the margin separating the support vectors. This can be modelled as a constrained quadratic optimization problem [20].

In many cases, the data may not be completely separable and a slack variable ξ is used to provide a penalty for training examples that lie on incorrect sides of the discriminant. Using optimization techniques the SVM is computed by:

$$\begin{aligned} &Minimize\ |w|\ subject\ to: \\ &\frac{1}{2}w^2 + C\sum_i \xi_i, \\ &subject\ to: \\ &y_i(w^T x_i + b) \geq 1 - \xi_i,\ \ where\ \xi_i \geq 0\ \forall i \end{aligned} \tag{2.22}$$

The weight vector is given by W and bias b. This method can be solved analytically by the sequential minimal optimization technique developed by Platts [39] with complexity $O(n^3)$.

2.5.5 Kernels

Most real world data is not linearly separable. In 1965 Thomas Cover proved that nonlinearly separable data is more likely to be linearly separable when cast into a higher dimensional space. This is known as Cover's theorem [8]. By raising the dimensionality of the input space by the use of a kernel function allows the feature space to remain linear while increasing the likelihood of data separability. Selecting a kernel that corresponds to a dot product in a higher dimensional space is known as the *kernel trick*. A kernel function that is Positive Semidefinite (PSD) satisfies what is known as Mercer's condition [3]. An example of raising the dimensionality of a binary class problem to a higher dimension making it linearly separable is given Figure 2.12 below by using the kernel function $K(x, x^2)$:

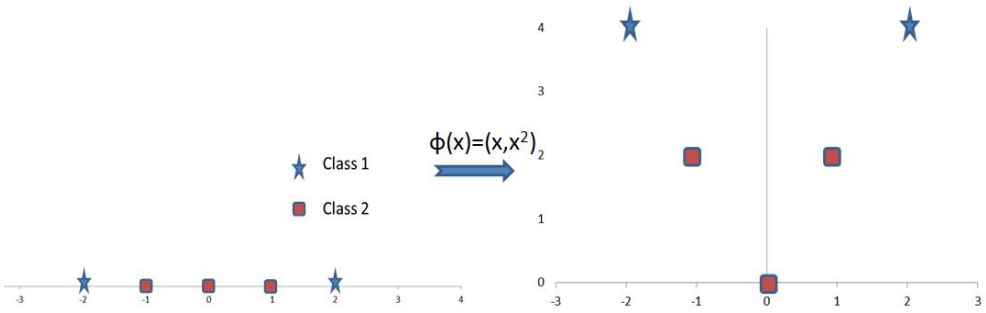


Figure 2.12: Data that becomes linearly separable when mapped to a higher dimensional space

Kernel methods are also an approach to compare the similarity between input data. In many applications two image representations can be compared to one another by using their respective feature vectors as input into a kernel function. As mentioned above the kernel trick can give a higher dimensional representation without even knowing an explicit mapping function but rather computing the dot product of two inputs. Generating the discriminant function of a SVM requires creating a Gram Matrix of feature vectors as input and compared using a kernel function. This linear kernel corresponds to the dot product, but any kernel can be used, ide-

ally one that satisfies Mercer's condition. This allows for higher dimensional mappings of the data to take place with a simple modification to the comparison function that has an increased chance of being linearly separable.

2.6 First Order Information Encoding

Despite the success of the *Bag of Words* (BoW) representation it has remained relatively unchanged in retrieval and classification systems. One proposed method to help recover from the problem of quantization loss is to use first order information known by *Vector of Locally Aggregated Descriptors* (VLAD). This method proposed by Jegou et al. [25] is to extend the encoding of building a histogram of visual word occurrences found within an image. The proposed approach assigns each descriptor $x \in \mathcal{X}$ to a centroid (using Nearest Neighbour) of a previously learned visual vocabulary as in Equation 2.15. After the assignment stage, the difference between each descriptor and assigned centroid is computed forming a set of *residual vectors*. These two stages can be seen in Figure 2.13 and Figure 2.14.

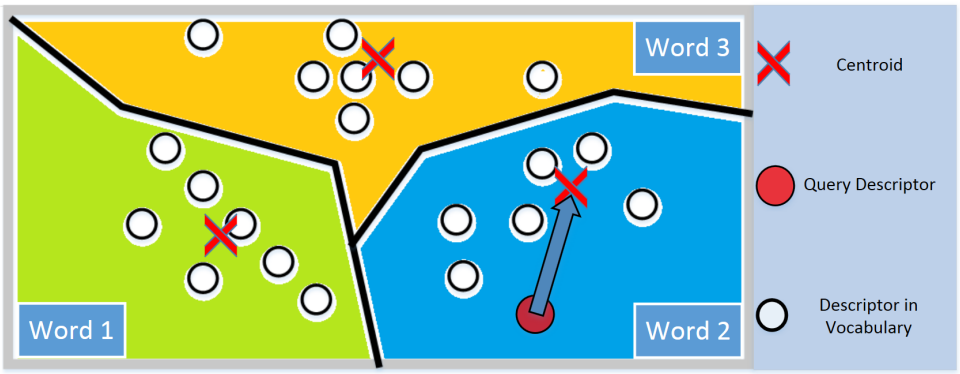


Figure 2.13: VLAD stage 1: Assign a descriptor to a word (centroid) in the visual vocabulary by Nearest Neighbour, as in the traditional BoW.

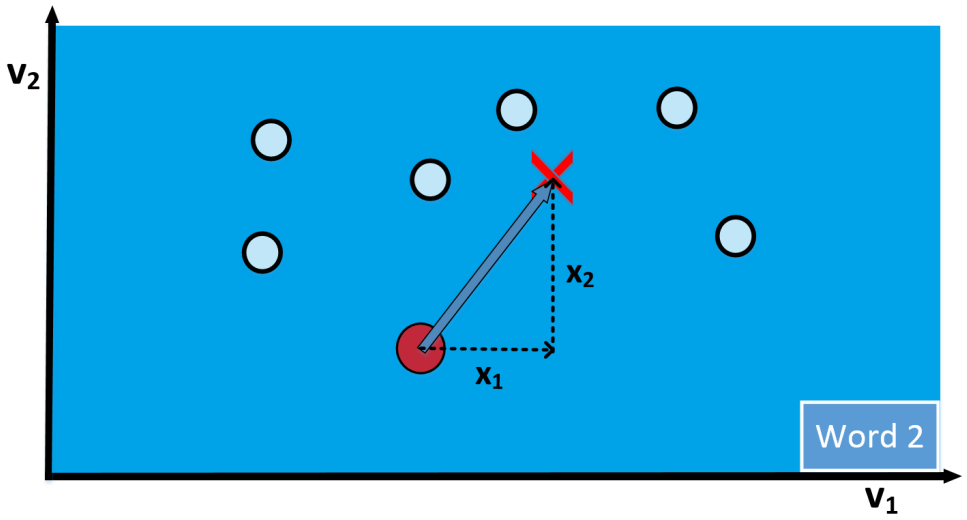


Figure 2.14: VLAD stage 2: The residual vector of a single descriptor and its assigned centroid is a sub-vector which will then subsequently be aggregated.

The formal definition of the VLAD calculation:

$$v_{i,j} = \sum_{NearestN(x)=c_i} \mathbf{x}_j - \mathbf{c}_{i,j} \tag{2.23}$$

With x_j and $c_{i,j}$ being the difference between each of the j^{th} components of the centroid c_i and the descriptor that is assigned to it. The resulting feature vector then has dimensionality $D = k \times d$. Where d is the length of the descriptor and k is the number of centroids.

Recent extensions to the VLAD encoding method with hopes of increasing its discriminative capability have been proposed. One method by Eggert et al. [12] involves creating a secondary dictionary by clustering the data within each Voroni cell in the dictionary, similar to using a vocabulary tree approach. Another technique by Liu et al. [33] involves creating a secondary dictionary offline at a finer resolution by computing a set of residual vectors on the original dictionary and then clustering by k-means over the residual set. The encoding would then be the aggregation of the primary VLAD vector along with a secondary VLAD vector at a finer scale using the clustered residuals. Further enhancements were also shown by Tak-Eun and Ho [28] involve reweighing the residuals prior to the aggregation stage of VLAD based on the salient regions of the corresponding local descriptor.

2.7 Triangulation Embedding

So far, the BoW and VLAD encoding methods have been introduced. Triangulation embedding uses a different type of encoding method that operates on a per descriptor basis and is the core work of this thesis.

Two main approaches for determining the position of a point are: trilateration which finds points by a distance measure and the other is triangulation. Triangulation as recently described by Jegou and Zisserman [27] uses angles and will be our point of focus. In the proposed triangulation embedding strategy, a set of image descriptors $\mathcal{X}$ are clustered using the traditional k-means approach. Once this is completed, a new set of image descriptors are then matched to the recently formed centroids. In the VLAD approach only the residuals of the query descriptors matched to the nearest centroid are computed. This approach deviates from VLAD because it does not use nearest neighbour assignment, but rather computes the residuals for each centroid.

There are two stages for this feature encoding method: embedding and aggregation. Their proposed embedding step $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$ $(d < D)$ maps each descriptor $x \in \mathcal{X}$ as:

$$x \mapsto \phi(x) \tag{2.24}$$

The aggregation stage is defined as a summation over all the mappings:

$$\psi(\mathcal{X}) = \sum_{x \in \mathcal{X}} \phi(x) \tag{2.25}$$

Given a set of visual words formed by k-means, $C = \{c_1, ..., c_{|_c|}\}_i,\ c_i \epsilon \mathbb{R}^D\ of\ |C|$. The first step to defining the triangulation embedding function $\phi_\triangle$ is computing a set of normalized residual vectors from the descriptors as:

$$r_j(x) = \left\{ \frac{x - c_j}{||x - c_j||} \right\},\ \text{for } j = 1...|C|,\ x \neq c_j \tag{2.26}$$

The set $r_j(x)$ are then concatenated to form R(x). Another difference here is that these residuals are unit normed thus describing their relationship to the visual vocabulary based on their angles, hence the term triangulation. The normalization happens on a per descriptor

basis, whereas in the VLAD case, unit normalization is performed after the residual vectors are summed. This approach is also known as *secant manifolds* [21]. Other papers that use this technique outside of the computer vision area refer to these visual words learned by k-means as anchor points.

For each descriptor $x \in \mathcal{X}$. It is important to point out that the unit ball is formed by quantizing residuals of image descriptors matched to the original dictionary on a validation set. For each subsequent descriptor to be embedded and later used for indexing or retrieval, the authors describe the triangulation embedding function, as:

$$\phi_{\triangle}(x) = \Sigma^{-\frac{1}{2}}(\mathrm{R}(x) - \mathrm{R}_0) \tag{2.27}$$

With the $\mathrm{R}_0 = \mathbb{E}_X[\mathrm{R}(X)]$ and Σ belonging to the covariance matrix from a training set. This is typically referred to as *whitening the vector*. Where whitening is the process of centering, rotating and scaling the data defined as:

$$\phi_{\triangle}(x)' = diag(\lambda_1^{-\frac{1}{2}}, ..., \lambda_D^{-\frac{1}{2}}) \;\; P^T \; \phi(x) \tag{2.28}$$

With λ being the largest eigenvalues and their associated eigenvectors from matrix $P \in \mathbb{R}^{D \times D}$. The above equation shown by Jegou and Chum [23] can improve image retrieval and research in [27] and [10] shows that the discriminative ability of the embedding is improved. It is also worth noting that selecting the first n eigenvectors such that $n < D$ jointly performs dimensionality reduction, later to be discussed in Section 2.9.

As the focus of this thesis is on image classification using multiple views, we proposed a method to aggregate the features from different views. For every view i in the image set, the features are whitened as described in Equation 2.28, using the same projection matrix P that is learned off-line across all views. The whitened feature vectors for each view are then concatenated to represent the image set:

$$\Phi_{\triangle}(x) := \phi_{\triangle i}(x)' \;\; for \; i = 1...n \tag{2.29}$$

Quite recently further enhancements to the Triangulation Embedding method done by Do et al. [10] have further improved the mean average precision (mAP) by 1.6% on the Holidays

and 3.4% on the Oxford5k datasets. The authors provide a more rigorous explanation of the embedding step 2.24 and use a higher order representation leading to a weighted quadratic at the anchor points. In their publication they also prove that VLAD encoding is related to another very successful encoding method known as *Local Tangent-Based Coding* (LTC) as introduced by Yu and Zhang [49]. LTC is based on the idea that features and codewords construct a smooth manifold. Through the use of anchor points, a nonlinear function $f(x)$ on $\mathbb{R}^d$, that embeds points onto the manifold can be approximated by a linear function on $\mathbb{R}^D$, where $d < D$ given by Equation 2.28. The linear approximation of the nonlinear $f(x)$ is given by $f(x) \approx P^T\phi(x)$, where P^T is found by PCA.

All of these encoding strategies are derived from a common representation known as *Super Vector Encoding*. The identifying characteristic of this set of encoding methods, is that all of these encoding methods use local information such as the residuals between the descriptor and nearest centroid (residual distance) to capture higher order information. The resulting feature vector has a dimensionality that is a factor of the dimensionality of the feature descriptor. Both the VLAD and Triangulation Embedding are examples of a Super-Vectors where both encoding methods capture local information between the descriptor and codeword. The original BoW representation that uses histogram encoding does not contain local information and is not considered a Super-Vector because the feature vector depends on the number of centroids, not the dimensionality of the descriptor type.

Encoding strategies that are derived from a Super-Vector, have shown superior performance as demonstrated by Peng et al. [38]. The authors conducted a comparative study on how the aggregation of different feature types performs with different encoding methods, for the task of action recognition based on multiple views from video sources. Another comprehensive evaluation was carried out by Yongzhen et al. [22] on several common datasets for scene classification and object recognition. Once again, encoding methods based on the Super Vector Encoding showed superior performance.

2.8 Power Normalization to Address Bursty Features

One of the phenomenons of feature detection is that once a feature is identified in an image there is a greater chance of a nearby feature (in the spatial sense) of also being detected with a high similarity score. This phenomenon is known as *feature bursts* as analyzed by [24]. This can often result in large peaks in the feature vector which can corrupt the similarity metric that is in many kernel based learning approaches. The variance in the feature vector can be stabilized by applying power-law normalization defined as:

$$f(z) = sign(z)\,|z|^{\alpha} \qquad 0 \leq \alpha \leq 1 \tag{2.30}$$

Power normalization can be applied on each encoded descriptor individually or the feature vector. It is worth noting that when applied to the feature vector when $\alpha = 0.5$ its known as the popular Hellinger Kernel.

Its worth noting that the peaks in the above images can occur from both the original BoW histogram component or they can occur from the vector residuals.

2.9 Dimensionality Reduction

One of the on going challenges in computer vision is that much of the data is inherently high dimensional and redundant. This raises the challenge of running time for many algorithms. Another problem is that as the dimensionality increases the distance between neighbouring data points grows exponentially. This is known as *the curse of dimensionality*. This problem is addressed by mapping the high dimensional data to a lower dimensional space based on the assumption that high-dimensional data lies near a lower-dimensional manifold.

One of the most common approaches to perform dimensionality reduction is to use Principal Component Analysis (PCA). PCA attempts to find an orthogonal projection of the data onto a lower dimensional linear subspace from n dimensions to d such that $n \geq d$ while capturing the greatest variance in the data.

There are several approaches to performing PCA. The sample mean of n samples of x observations defined as:

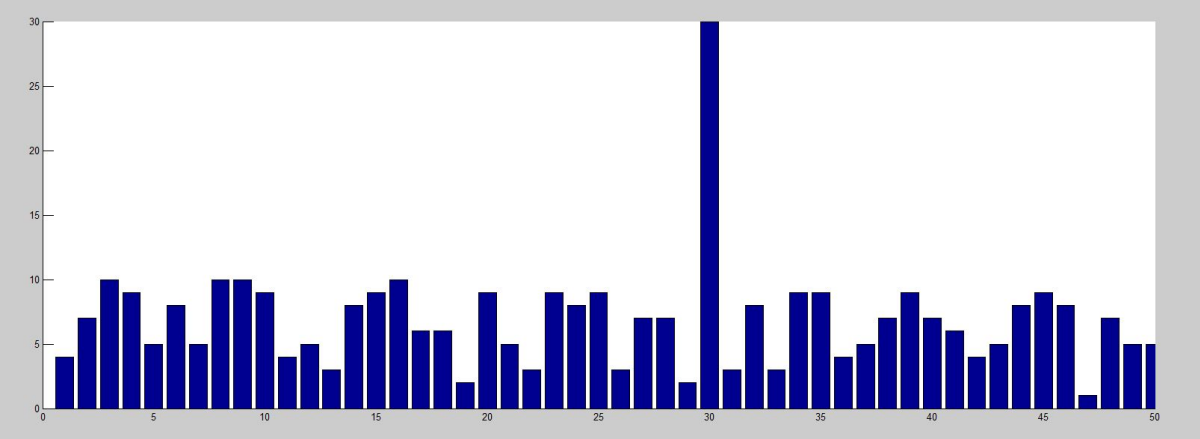


Figure 2.15: Example of a bursty feature in a histogram

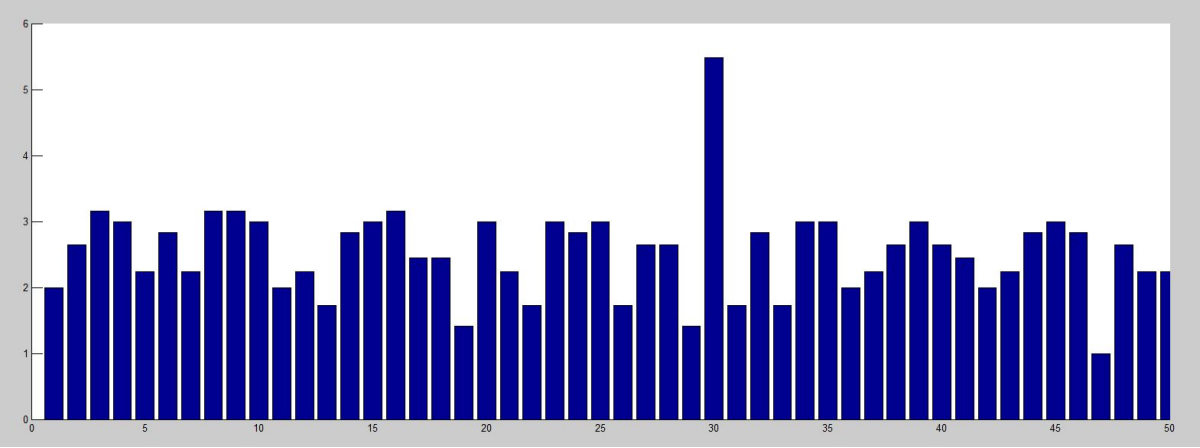


Figure 2.16: After applying power normalization to the histogram

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{2.31}$$

With the mean-centered covariance defined as:

$$\sum = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T \tag{2.32}$$

For simplicity and keeping with common notation we will refer to the covariance matrix with mean subtraction along the columns as $\sum$ as XX^T.

Singular Value Decomposition (SVD) is defined as taking a matrix X of size m x n and factoring it such that:

$$X = USV^T \tag{2.33}$$

Where U is a m x m orthogonal matrix, V is a n x n orthogonal matrix and S is a diagonal matrix whose diagonal elements are in decreasing order.

If the provided matrix to be decomposed is a square and symmetric matrix as in the case of XX^T then SVD is equivalent diagonalization of XX^T by U resulting in each column vector of u in U being eigenvectors. The numerical method to solving the SVD problem in this work was the use of the Jacobi Method.

The covariance matrix $\sum$ is symmetric of size nxn and thus can always be diagonalized and diagonalized by an orthogonal matrix. Two matrices A and D are said to be similar $A\tilde{D}$ if an invertible matrix P exists such that $A = P^{-1}DP$

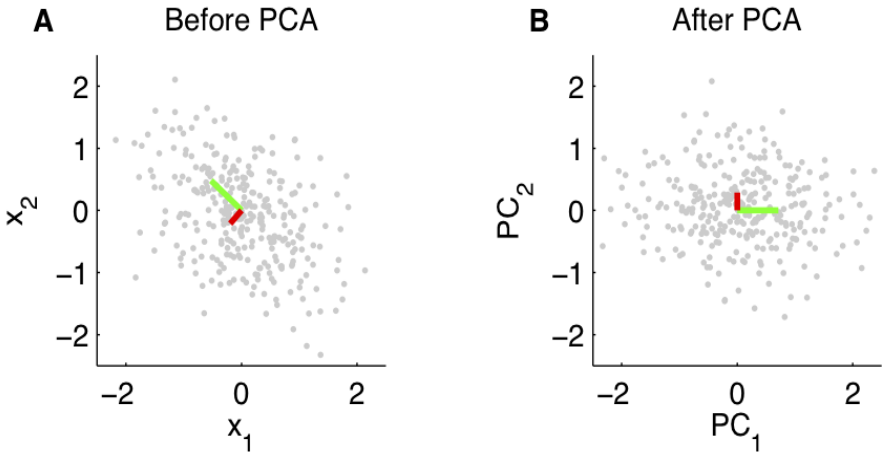


Figure 2.17: PCA example on a 2D dataset after projection of the data onto the principal component axes[2]

The principal components refer to the orthogonal vectors forming the new coordinate system spanned by the dominant eigenvectors of the data covariance matrix [15]. The eigenvectors capture the directions that maximize the variance in the data with a corresponding eigenvalue that captures the magnitude of variance. This example can be seen in Figure 2.17.

[2]http://www.user.tu-berlin.de/felix.biessmann/mmreview/

Chapter 3

Part Classification on a Conveyor Belt Framework

There has been previous research in using multiple view techniques for object recognition and retrieval tasks [14, 47, 40, 31, 30, 46]. It has already been demonstrated that the addition of multiple views within the BoW framework can improve accuracy for object classification as demonstrated by Savarese and Lei [45] and Fu et al. [14] on the same dataset. On other datasets, the task of object classification can be improved using multiple views as seen in the works of methods that rely on high quality reconstruction for classification, but 3D reconstruction is heavily affected by noise resulting in holes in the model in addition to being computationally more expensive. In many machine vision applications that require rapid classification using methods of 3D model generation are simply not practical when using standard cameras.

Many existing systems require the manufacturer of the vision system to create new templates, or modify some parameters as most current systems use low level vision techniques. Our goal was to use computer vision techniques that could learn an additional object class during normal manufacturing production that could be performed with minimal supervision. This is often at the cost to the customer and possibly cause system downtime. Another real concern for manufacturers that do buy these vision systems is that computer hardware can break at any time without warning, in many cases halting production. The goal here was to create a setup that could address both of these problems.

3.1 Physical Setup

The purposed method of performing classification on a conveyor belt was to use multiple cameras to view the object from four sides. The motivation behind this was to make use of additional information that comes from multiple views and possible occlusions of discernible regions that may take place on the part. The occlusion of discriminating features on the object is very much a real case for the in-house dataset that was constructed. The hopes being that this approach would boost classification performance.

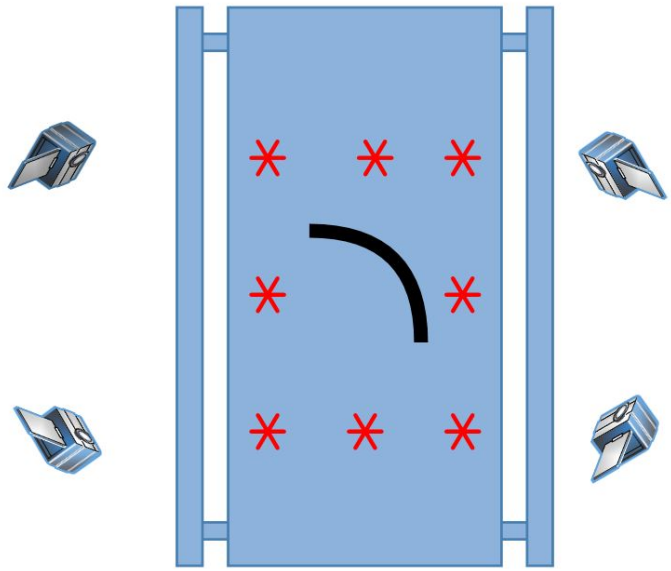


Figure 3.1: Positioned cameras focused over a conveyor belt

A diagram of the envisioned system can be seen in Figure 3.1 showing how the cameras were positioned around the conveyor belt to capture from the four corners. Naturally the four captured images provide a different view point of the object to be classified as can be seen in Figure3.2.

A mock conveyor belt that had a moving board with wheels would sit on top of a wooden frame. The moving board would simulate the belt portion of a conveyor and allowed for one directional object movement. Allowing for multiple image captures of a moving part without accidental displacement of the part and provide a more authentic simulation. The constructed conveyor can be seen in Figure 3.3.

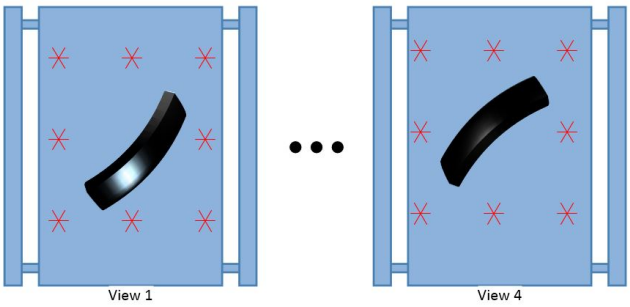


Figure 3.2: A diagram showing how different views can look drastically different depending on the view point

The cameras were positioned about 1.4 meters above the belt and angled at approximately 45° down towards the object. The spacing of the mounted cameras was arbitrarily selected to ensure the conveyor portion of the belt was always viewable. For consistency once positioned, the cameras were not moved.

The actual cameras themselves were simple, low cost, commercially available USB 2.0 web cameras. These cameras are able to record video in 1080p; however, all captured frames were processed and stored at a resolution of 1280x720. Many other works relating to multi view images require more specialized and costly cameras. The goal here being that the selection of cameras need not be expensive and provide a cost efficient solution. The type of camera used can be seen in Figure3.4.

3.2 Object Localization

Given that the cameras are positioned above the conveyor belt at an angle, there are regions outside of the belt that are captured. Since the regions off of the belt provide no additional information about the part red patches as seen in Figure 3.3 were placed on the belt surrounding the object. As the focus of this work is on the classification aspect of the problem and not image segmentation the red markers served to provide a simple boundary of the object.

We aimed to overcome this problem by finding a generalized approach that avoids advanced image segmentation techniques. The boundary surrounding the markers are evaluated to see, if they fall within a certain range and a boolean value is returned. The set of boolean values

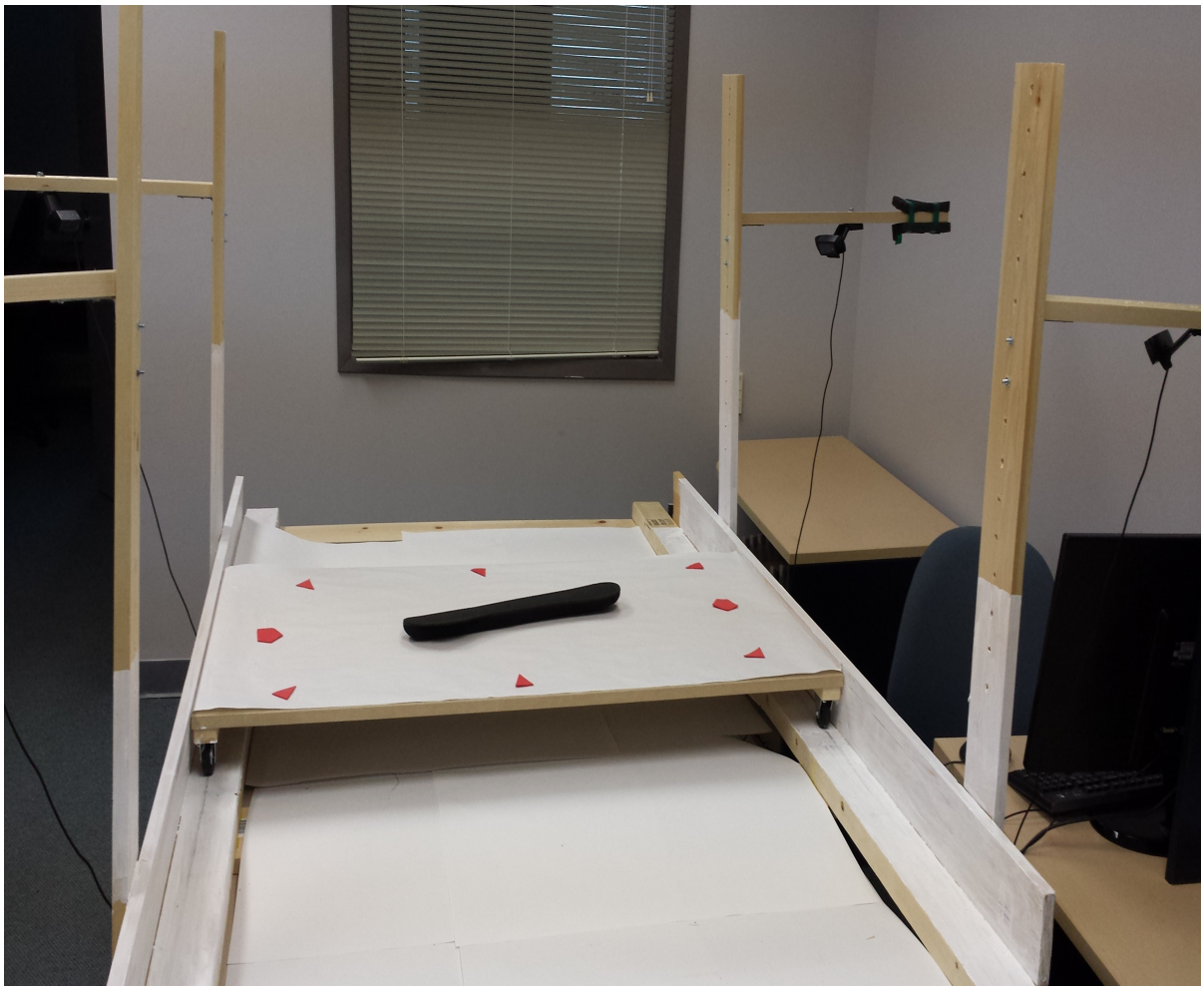


Figure 3.3: Mock conveyor used in lab to gather data.

creates the binary mask. Given three discrete sets $\Theta = \{\theta_1, \theta_2, \theta_3\}$ representing each colour channel, segmentation by colour thresholding is defined as:

$$g(x,y) = \begin{cases} 1 : f(x,y) \subseteq \Theta \\ 0 : f(x,y) \nsubseteq \Theta \end{cases} \tag{3.1}$$

To reduce the number of spurious pixels being identified as the markers the basic noise reduction technique of median filtering was applied as a preprocessing step. One of the masks generated can be seen in Figure 3.5b.

The final stage of enclosing the object and thus separating the object from the background was to connect the identified regions in the mask. This is done by applying a convex hull

Figure 3.4: The type of camera used in our experiments [1]

as seen in 3.6a. Given a set of n points corresponding to pixels for which $g(x, y) = 1$ the computational complexity of finding the hull is $O(n \log n)$ based on the Graham method [17].

At this stage the current mask contains the image but also may contain some markers that fall within the hull. We remedied this problem by applying the morphological operation of erosion. Since the hull could take on various shapes depending on the image captured we needed to shrink the size of the mask while preserving its shape. The convex hull could vary widely among views as it relates to the position of the belt and shrinking the mask by a set amount of pixels runs the risk of masking out some of the auto part in the image. This approach helps to minimize the impact of this problem. At this point only features within the masked region will be detected shown in Figure 3.5d.

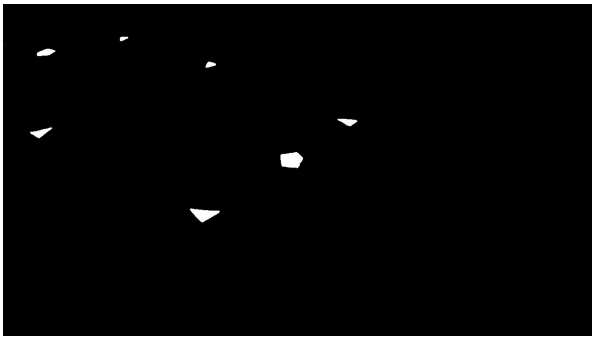
Using the definition from Woods and Gonzalez [16], erosion stems from set theory and is defined given two sets A and B in Z^2 the erosion of A by B denoted $A \ominus B$ is given by:

$$A \ominus B = \{ z \mid (B)_z \subseteq A \} \tag{3.2}$$

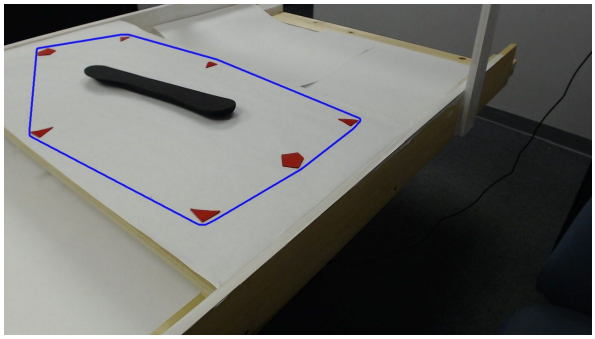
Set B is commonly called a structuring element. In the context of digital images B is a mxn

[1] http://www.logitech.com/en-us/product/hd-pro-webcam-c920

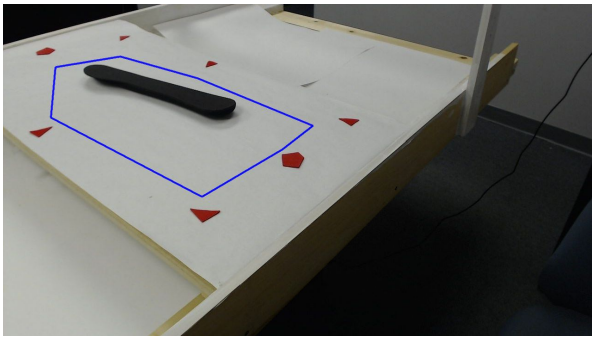
(a) Markers identified by colour thresholding



(b) Binary mask on which to apply the convex hull



(c) Object enclosed by the hull shown in blue



(d) Reduced area to only enclose the part

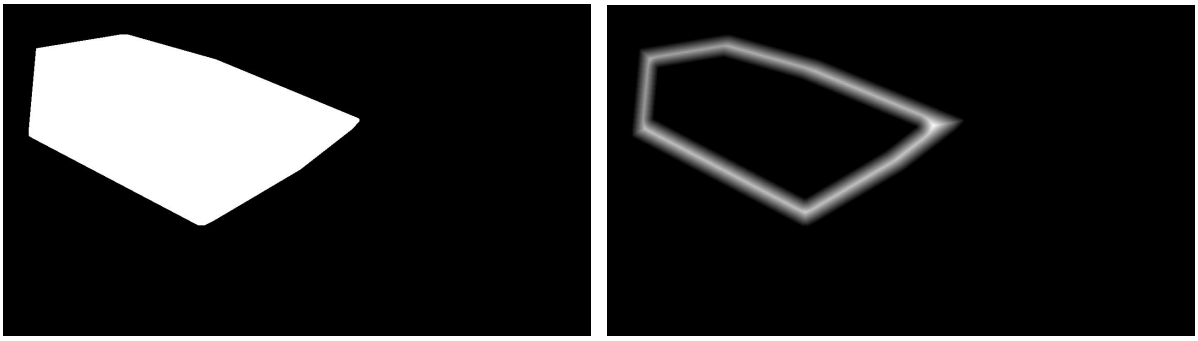
Figure 3.5: The object localization process

matrix and is the set of all points, that when translated by z is contained in A.

Using the definition of a digital image from Section 2.2.1 given image A and treating it as a function corresponding to (x, y) pixels the result of performing erosion over A with structuring element B returns a mask g given by:

$$g(x,y) = \begin{cases} 1 : \text{if B fits A} \\ 0 : \text{if } B \text{ does not fit A} \end{cases} \tag{3.3}$$

A visual representation can be seen in 3.7. It is worth noting that structuring elements can have varying shapes. Morphological operations can also be useful for many image processing operations such as hole filling or perimeter extraction. The formal definition of the particular operation of erosion 3.2 from set theory also makes it clear how this can be applied on non-binary images and extended to multiple channels or volume images.



(a) Convex hull (b) Distance map of the hull after erosion

Figure 3.6: The shrinking of the hull by erosion

3.3 Extension to Multiple Views

Historically, most problems in computer vision involving the use of stereo or multi view images try to compensate from the fact that we live in a 3D world and when taking images 3D points are mapped to a 2D plane, for the standard camera model resulting in a loss of depth information.

The basics of how a camera works come from the pinhole camera. The same concept takes place when working with digital images. Multiple Rays of light emanates from a single point known as the point source. The concept of a pinhole camera is to capture only a single ray from each point on the object. The model lets rays of light into a small opening in a light proof box that inverts the image. The opening of the box that allows light in is called the aperture. An ideal pinhole camera has an infinitely small aperture that will only allow one ray to pass through. Realistic cameras based on this model are not ideal and have more than one ray per point entering the aperture. Small apertures allow less rays in resulting in a darker but more focused image. Larger apertures allow for a brighter but more blurry image.

The pinhole camera model does allow for an approximation of a perspective projection for generating an image. The mapping of the object to the image plane can be see in Figure 3.9. Its worth noting that in this diagram the image plane is actually in front of the camera.

The mathematics of projection remains the same except now the image is not flipped upside down. Using homogeneous coordinates this model can be described in matrix form by:

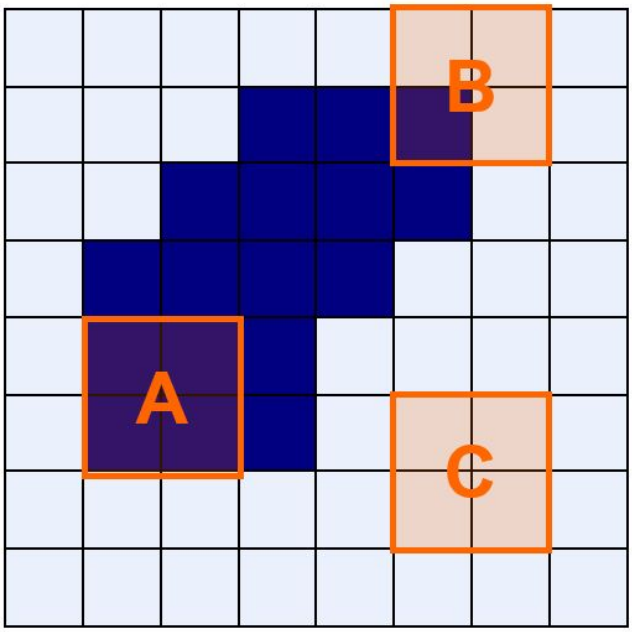


Figure 3.7: Structuring elements note that element A fits whereas B and C do not.[2]

$$\begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.4}$$

The above formulation can be rewritten as:

$$z\mathbf{m} = P\mathbf{M} \tag{3.5}$$

Most cameras do not use an actual pinhole to filter rays of light but make use of a lens for the purpose of focusing the rays of light as seen in Figure 3.10

The matrix $\mathbf{m}$ can simply be rewritten as $\mathbf{m} = (fx/z, fy/z, 1)^T$. The drawbacks regarding this formulation is it lacks camera pose and internal geometry. Through matrix decomposition of matrix P the decomposition can give intrinsic and extrinsic parameters. The intrinsic matrix contains internal parameters such as focal length. The extrinsic matrix contains parameters used to define motion about a static scene relating the world coordinates to the camera

[2] http://www.slideshare.net/shkulathilake/morphological-image-processing-43465879

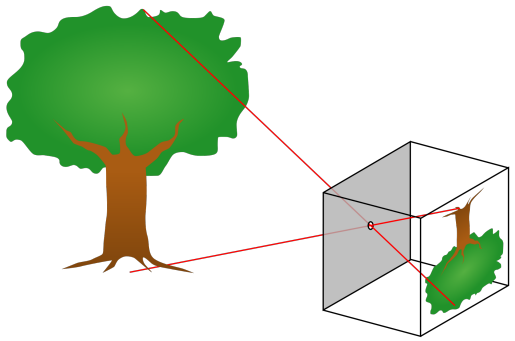


Figure 3.8: Pinhole

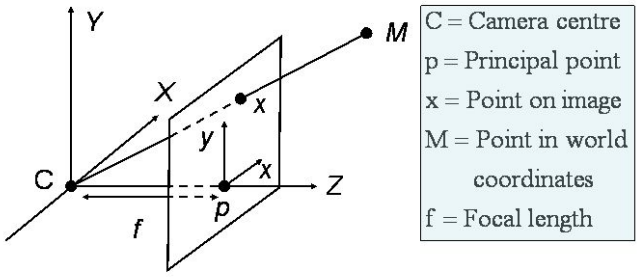


Figure 3.9: Projection diagram

coordinates.

These camera parameters can be found through a process known as camera calibration. The distortion coefficients can be identified by both sets of parameters and corrected for. Glossing over the details of camera calibration it is important to point out that this plays a very important role in recovering depth. Most of the techniques above would require the use of calibration for whatever multi view information they deal with.

This work focuses on the use of multiple views of the image in which to capture features on. Our belief being that using multiple views of an object can improve the classification performance. This has been seen by the works given by [14] and [45], using the BoW model for object classification is improved when the number of views is increased without having to handle with depth information. The belief that different views of the object contain additional information was something of interest. For our particular dataset that is introduced in Section 4.1.1, the actual objects could not always be distinguishable from one view as the differences

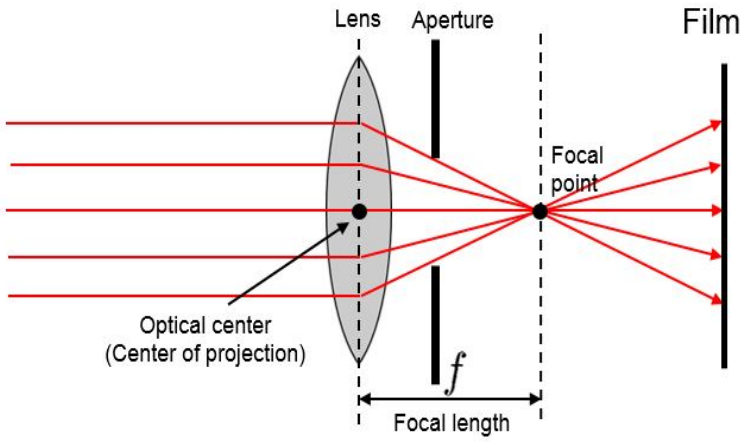


Figure 3.10: Lens used to focus light[3]

between some of the part classes could only be identified when looking at the object from multiple view points.

This is where this work deviates from many of the above approaches. Many techniques for generating 3D models require many views of the part. This is evident by the large number of views of the same object in publicly available datasets. Many stereo imaging techniques have two cameras very close together. The objects in-house dataset can only be distinguished at a visual level when the views are angled far apart as in our setup. Other feature based techniques to work on the correspondence problem between sets of features across multiple views relies on strong discriminative features. Unfortunately our objects for classification are all black and have some level of luster. Reliable feature matching techniques are not feasible as with most reconstruction methods using correspondence when there are not enough features matched consistently across the frames with high confidences they are dropped. The end result is an extremely sparse point cloud representation with very few points and certainly not enough points to regenerate the full model. There is of course the technique of deflectometry [29] uses very specialized equipment for paint defects after performing surface reconstruction. The main issue with that approach is the object needs to be stationary. Even if the common 3D approach for image classification could be reliably performed our system requires classification of the

[3]http://www.cse.unr.edu/ bebis/CS485/

object on a moving belt near real time.

Chapter 4

Experimental Results

4.1 Introduction to Datasets Used in Experiments

4.1.1 Introduction to In-House Dataset

As mentioned earlier the object would sit atop the board with a white background with red marking points. All of the parts used were automotive parts. The parts are named window pillars and guides as shown in Figure 1.1. On each of the parts there is a white label appended, which due to a confidentially agreement with the manufacturer the label is hidden from this publication. Where noted we have selected publicly available images to give the reader a sense of the type of parts used. The stock images which were taken from the popular automotive website ECS Tuning. The stock images were selected to show the levels of similarity among the parts and better illustrate the problem. For all experiments, conducted we used our dataset and no stock images were used. Any non-stock image displayed within this publication were used in the actual dataset. Any stock image will contain the ECS Tuning watermark. It is worth noting that all of the parts were new and their condition was the same as used as in the factory.

One of the challenging aspects of this dataset is the high level of similarity between parts that serve the same function but differ in some physical characteristic. Parts that serve the same function but can vary in any orientation, length and luster are all considered distinct classes because they have their own identification number that the manufacturer uses.

The automotive parts could vary in luster. During the painting process all of the parts are

painted black. From the black paint there are two types of paint that could be applied. One of the paints gives a matte finish whereas the other one gives the part a glossy finish. This can be seen in Figure 4.1

Figure 4.1: Window pillars comparing the two types of lustre: matte and glossy

Different models of vehicles often require the same part, but just at a different length as vehicles are made in different sizes. In most cases there are multiple parts needed for the same vehicle but differ in required size. This can be seen in Figure 4.2.

The parts could also be different based on the intended location of install. For example a part that has the same functionality but is designed to be installed on the left side of the vehicle will be different than the part that is designed on the right side of the vehicle. These parts are denoted to belong to a different orientation as seen in Figure 4.4.

Parts that had the same functionality/purpose but could vary in the ways mentioned above are referred to as belonging to the same *family*. For this dataset there are 39 classes. Since each view provides a snapshot of the object at four different angles is named an *Image Set*. Of the 39 classes there were 35 images sets per class for a total of 1365 images .

Figure 4.2: Window pillars comparing the longer and shorter window pillars for front and back placement

The images were captured by moving the part along the belt. To simulate a real world environment, after each image is taken, the belt is moved and the part's location sitting on the board is moved, rotated or both. During all movements, the belt remained in view of the cameras while the part remained inside the markers used for tracking, and the rotation was no greater than approximately ± 45°.

There was only one part per class to generate all the image sets. A natural concern was the fact that the label was fixed and would bias the dataset since the label contained numerical characters. To address this obvious threat to experimental validity in each of the 5460 images the four corners of the label region were manually marked and stored as a mask. This label mask was used during experimentation to stop features from being found within or on the edge of the label.

Figure 4.3: Window guides of varying length[1]

Figure 4.4: Window pillars with left and right orientation[1]

4.1.2 Introduction to Caltech 3D dataset

As mentioned earlier the in-house dataset is under a confidentiality agreement and thus cannot be publicly released. To show that the contributions contained within this thesis is not just valid but can also be extended to other multi view datasets. The publicly available multi view dataset known as the CalTech 3D object categories [45] was used to as benchmark. This dataset contained 10 individual object instances from 8 categories. Each instance had 8 viewing angles, 3 heights and 3 scales. See Figure 4.5. One exception was the car dataset which only had two heights instead of three. The images were 400 x 300 pixels and the dataset contained a mask

[1] https://www.ecstuning.com

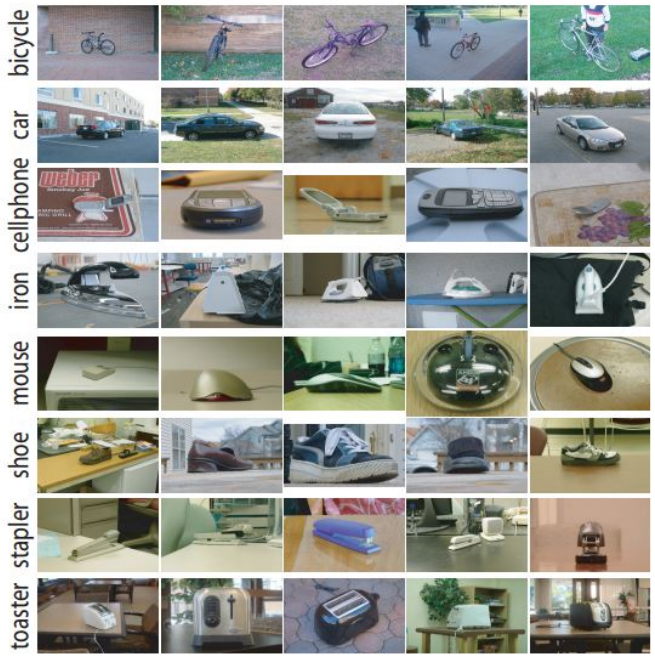


Figure 4.5: A diagram showing samples of the CalTech 3D dataset

of each object.

4.2 Evaluation of Single and Multiple Views

4.2.1 Classification Performance with a Single View

The following two experiments evaluate classification performance using different features, on both the in-house dataset and the CalTech 3D set. In both cases, only the first view point was used throughout the entire BoW pipeline. This reduces the problem to image classification from a single view.

4.2.2 Results

The classification results for the in-house dataset are shown in Figure 4.6. ORB features had a greater classification accuracy over SIFT and SURF features. For ORB and SIFT features, once the dictionary size reaches 256, the classification accuracy as a function of dictionary size starts to level off and the improvement in classification is quite minimal and does not justify

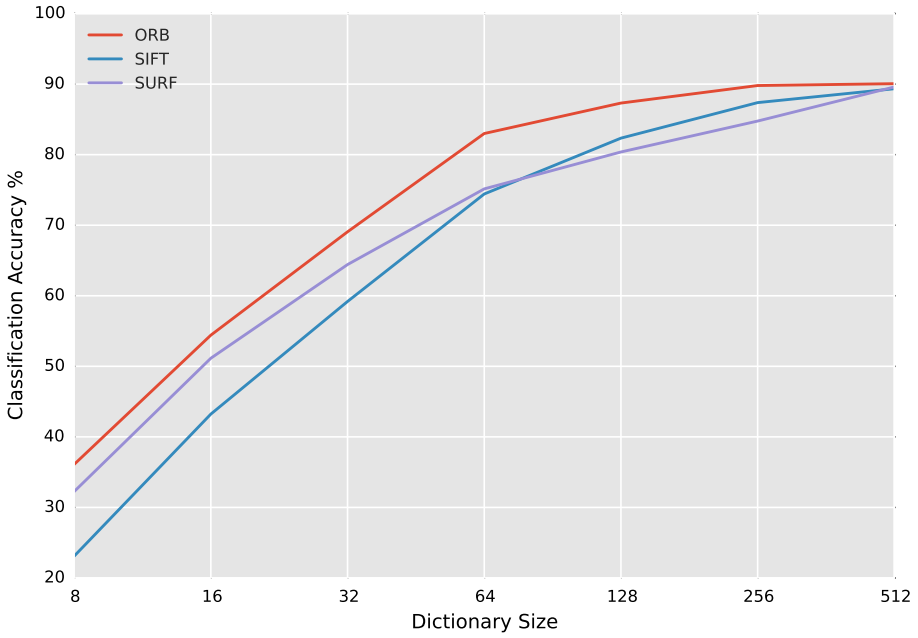


Figure 4.6: Classification performance on in-house dataset using the BoW model comparing SIFT, SURF and ORB features against vocabulary size using a single view

the increased computational time.

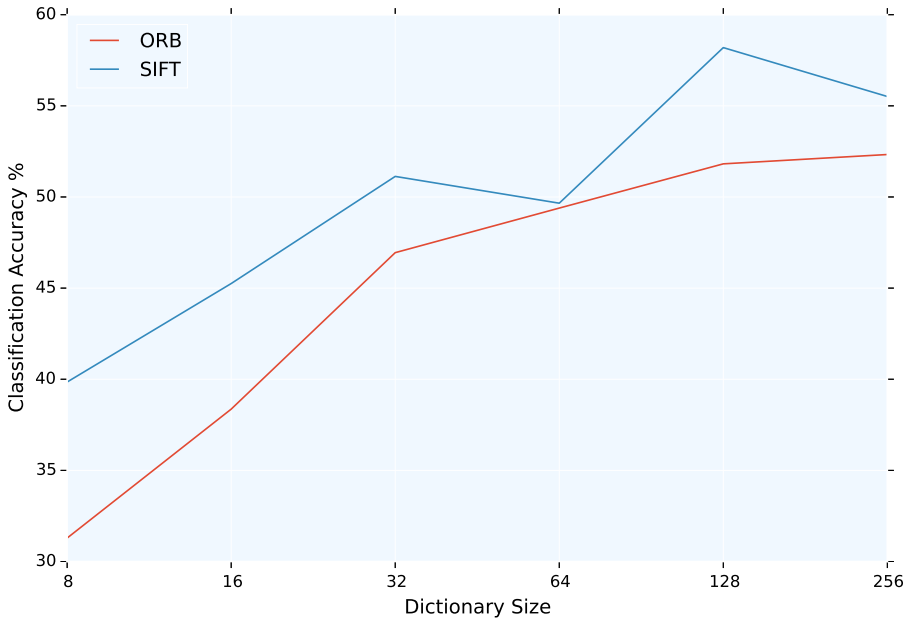


Figure 4.7: Classification performance on CalTech dataset using the BoW model with varying vocabulary size using a single view

The results shown in Figure 4.7 shows classification accuracy by feature type as a function of dictionary size using a single view of the CalTech dataset. Unlike the prior experiment, SIFT features outperformed ORB features by a fair margin. That being said, looking at the results for SIFT features, it is concerning to see these fluctuations in accuracy as seen when the dictionary size is 64 and 256.

4.3 Classification Performance with Multiple Views

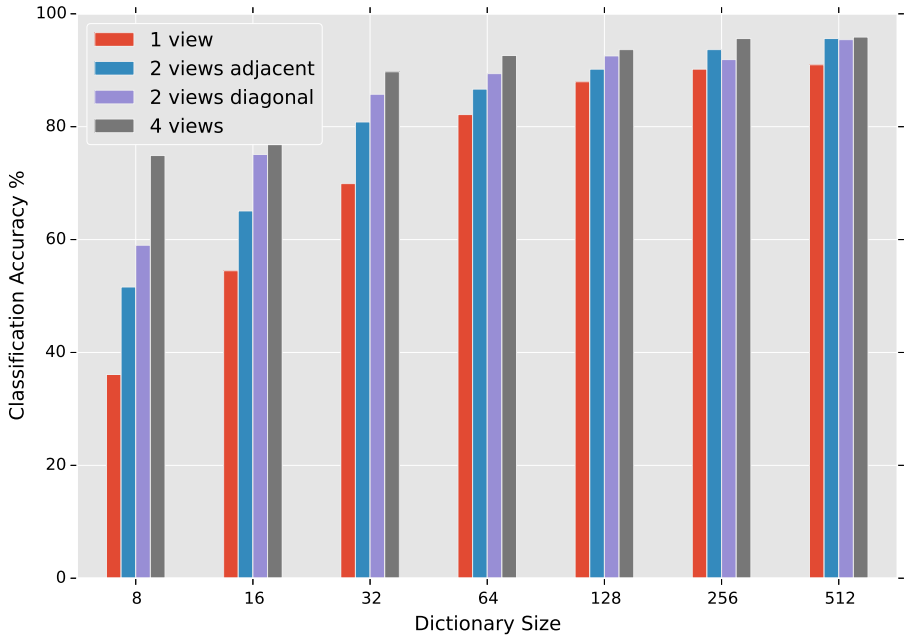


Figure 4.8: Classification performance showing the impacts of using different combinations of views and dictionary size on in-house dataset

The results for varying the number of views and dictionary size for the in-house dataset are presented in Figure 4.8. The results within this figure indicate two things. Firstly, increasing the dictionary size has a positive impact on classification performance. In fact the general trend of improved performance by increasing the dictionary size using only one view as seen in 4.6 also extends to multiple views. This improvement can likely be attributed to a reduction in quantization loss as the number of centroids is increased. Secondly, the takeaway from

this is the consistent improvement of classification performance when the dictionary size is increased. As discussed earlier multiple views of an object can increase the information that can be extracted from the object for classification. This is especially true for the case when objects appear the same from a particular view but have clear visible differences from other views. These distinctive differences are often occluded by the object itself. This is the case with our in-house dataset.

Method	Classification Time (ms)
SIFT	2325
SURF	1210
ORB	350

Table 4.1: Comparison of classification time [11]

The classification times are listed in Table 4.1. As mentioned earlier, many applications require near real-time classification. As is the case for our application. The full set of algorithms required for descriptor extraction is a core component in running time. As mentioned by the authors Rublee et al. [43] ORB descriptors are faster to compute compared to SIFT and SURF descriptors. Their comparison is focused on descriptor matching and not necessarily the image classification task.

In Table 4.1 we evaluate the time it takes from reading an image set, detecting keypoints, computing descriptors, constructing a feature vector and finally performing classification using all four views. For this evaluation, the number of detected features was set to 300 features per view and the dictionary size was 1000 centroids. The dataset used was a subset of the in-house dataset containing 25 classes. The classification time shows that ORB has a speed increase of over 6x compared to SIFT and 3x compared to SURF. With the multiple view framework that was proposed SIFT and SURF are not ideal for near real-time classification.

The same approach of assessing classification accuracy as a function of the number of centroids used to construct the dictionary was carried out on the Caltech dataset. For a fair comparison between the results obtained in Figure 4.8 the same experimental setup, and the use of 10-fold cross validation was used. Here the results show that SIFT descriptors perform better then ORB. This deviation could be attributed to a few things. Firstly, the dataset used

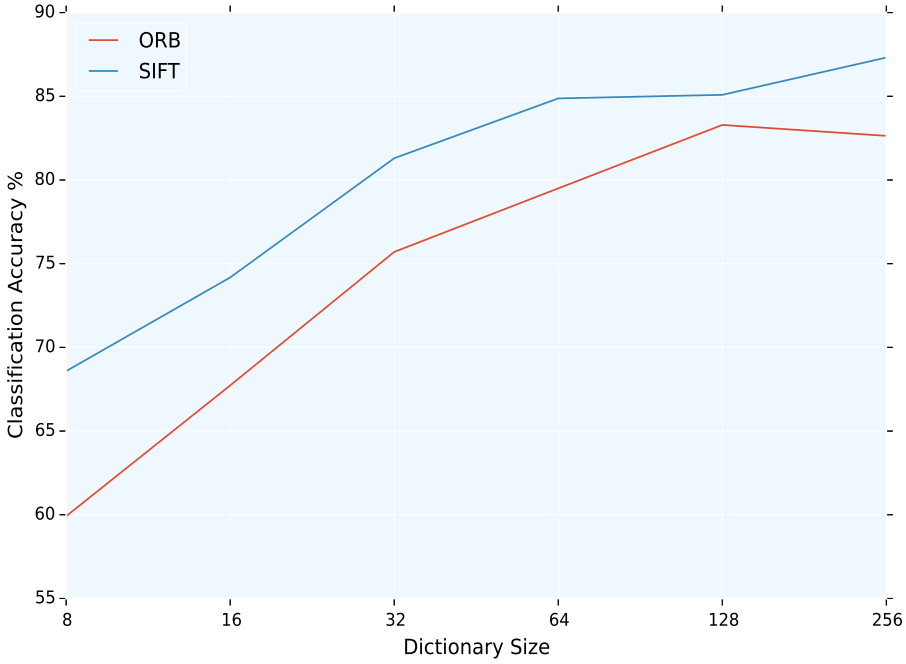


Figure 4.9: Classification performance using multiple views of the CalTech 3D dataset

objects that were captured from 3 different heights and 3 different scales. In our setup, have such a great variation in height and scale as the object is captured once it enters the field of view from the four cameras. Secondly the cameras were at a set height. The take away from this would be SIFT descriptors may be better suited for image classification when the object can appear at drastically different heights and scales. ORB descriptors handle scale by the simple image pyramid approach. SIFT descriptors handle scale by the more elegant approach of feature description at multiple scales by using the Difference of Gaussian, stemming from scale space theory.

Figure 4.8 showed that for our in-house dataset, increasing the number of views consistently improved classification performance regardless of the dictionary size used. This would lead to the conclusion that increasing the number of views improves classification performance under the assumption that the features found in additional views would provide more discriminative information to assist with the classification task. This also holds true for the CalTech 3D set as seen comparing the confusion matrices generated by an average accuracy over the 10-fold

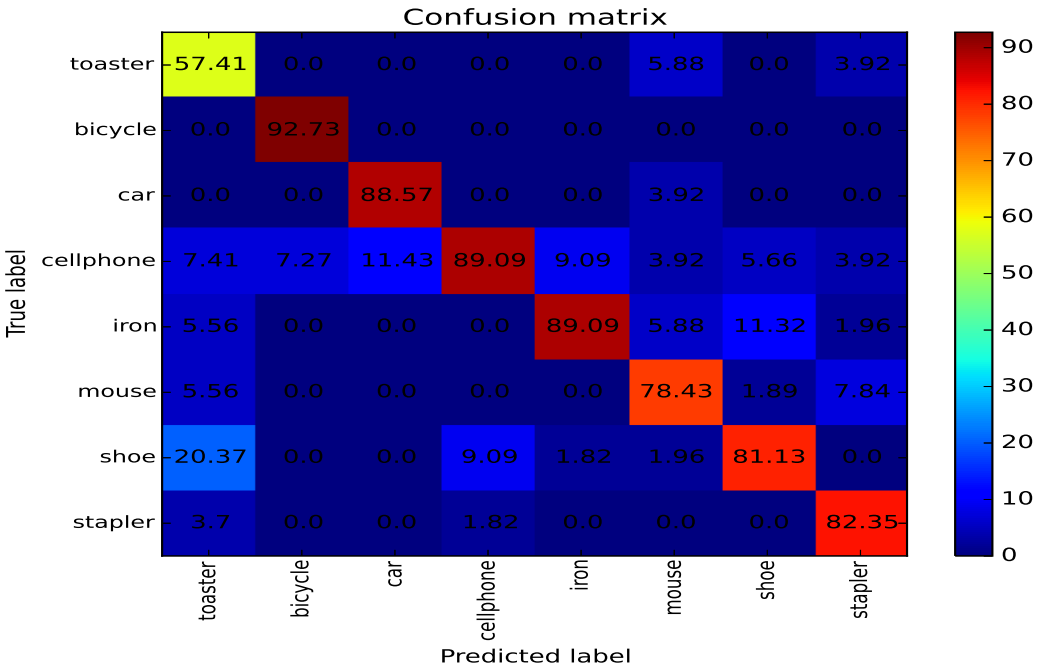


Figure 4.10: Confusion matrix of the CalTech 3D dataset single view with classification results averaged over 10-folds

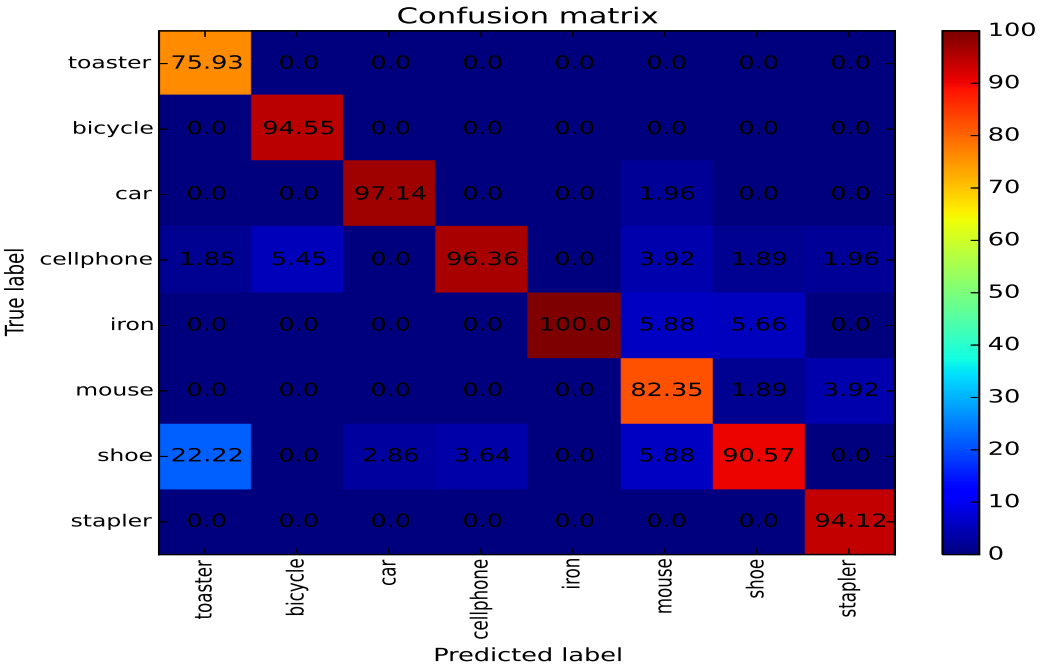


Figure 4.11: Confusion matrix of the CalTech 3D dataset multiple views with classification results averaged over 10-folds

cross validation. The two matrices can be seen in Figure 4.10 and Figure 4.11. Both sets of results used 256 centroids for the visual vocabulary and 350 ORB features. It is clear just by visual inspection that using the multiple view setup considerably improved classification accuracy. For example the toaster object has a mean accuracy of being correctly classified at 57.41% but extending this to using all 8 views, the mean classification accuracy jumped to 75.93%. For everyone of the 8 classes in this dataset classification accuracy improved. The mean classification accuracy improvement over the 8 classes was 9.03%

4.4 Evaluation of Multiple Dictionaries

This section evaluates different strategies on how to utilize the visual vocabulary most effectively. The method in which the visual vocabulary is used ultimately affects the resulting feature vector that is fed to a classifier. In this work three approaches are evaluated. The following three subsections use the following definition for a set of image descriptors:

Every image contains one or more features x belonging to a set of features X.

4.4.1 A codebook for each view

Every image belongs to a view i and has a corresponding visual vocabulary/codebook (CB) i containing n centroids. Multiple histograms are created by associating X_i to CB_i forming the histogram H_i. The feature vector is then defined as $fv := H_i$ for views $i = 1, ..., k$. The resulting feature vector has dimensionality $1 \times ni$

4.4.2 A single codebook for all the views

A second approach is a bit more simple. For every $x \in X_i$ assign x to the single codebook CB_1 of dimensionality n. The feature vector is $fv := H_i$. The resulting feature vector has dimensionality $1 \times ni$

4.4.3 Summation of visual words

The final approach is to sum the histograms H corresponding to each view. For every $x \in X_i$ multiple histograms H_i are created using one CB the same way as in Section 4.4.2. The feature vector is defined as $fv = \Sigma_{i=0}^{k} H_i$. The resulting feature vector has dimensionality $1 \times n$. The above approach simply requires the indexes for each of the histograms to be consistent.

4.4.4 Results

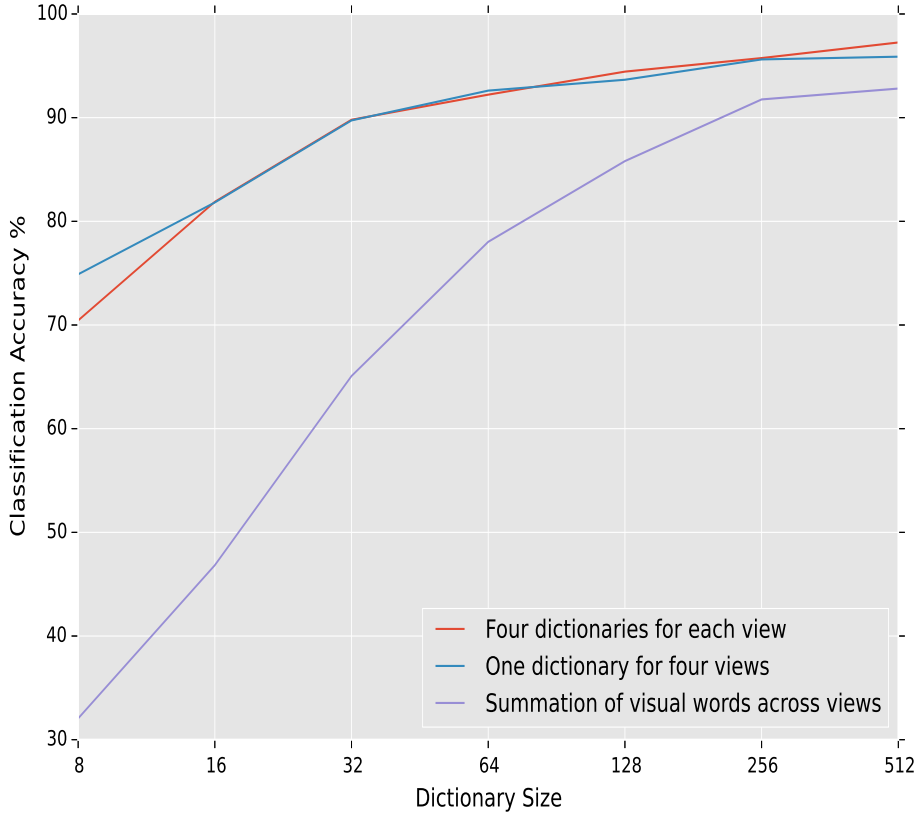


Figure 4.12: A comparison on using different methods of utilizing the visual vocabulary for image classification using in-house dataset

The results using the three approaches can be seen in Figure 4.12. It is clear that the summation strategy performs the least favourably. There is however a speedup in classification using this approach as the length of the feature vector does not depend on the number of views.

Unfortunately the results show that the summation method performs much more poorly for small codebook sizes but does catchup with the other approaches for large codebook sizes.

The results also indicate two things. The first being that classification accuracy is comparable for either approach but using the multiple dictionary approach performs slightly better than one dictionary when the number of centroids is larger. For this particular dataset, this effect can be seen when the number of centroids is 256. For all of the experiments, the second strategy of using one dictionary for i views is used where applicable.

4.5 Kernel Selection for SVM Optimization

In subsection 2.5.5, the concept of kernels was introduced. Kernel methods can be a useful tool for improving classification by operating in an implicit feature space by using a kernel function without having to map the input data itself by use of the *kernel trick*. This implicit feature space can be of a greater dimension in which data is more likely to be separable. The only requirement is that the selected kernel meets the requirements for being positive semi-definite, also known as a Mercer Kernel.

4.5.1 Traditional Kernels for use with an SVM

The following kernels are some of the most commonly used kernels for the purpose of implicitly mapping the feature space to a higher dimension with the exception of the linear kernel. Assuming the following are column vectors:

Linear Kernel:

$$k(x, x') = x^T x' \tag{4.1}$$

Polynomial Kernel:

$$k(x, x') = (\gamma x^T x' + r)^d, \quad r > 0 \tag{4.2}$$

Gaussian Kernel

$$k(x, x') = \exp\left(-\gamma \|x - x'\|^2\right) \tag{4.3}$$

For the polynomial and Gaussian kernel there are two hyper-parameters that normally have

to be learned, γ and cost. The cost parameter is to penalize the model for training samples that are misclassified. Normally a grid search is employed to search the parameter space using a logarithmic scale.

4.5.2 Results

The grid search for all experiments below are for finding the best γ and Cost C. The range of values used are: [-10,-9, ... , 9, 10]. These values in the set are the exponents and here we used a base 10. So the γ values are: $[\gamma^{-10}, \gamma^{-9}, ..., \gamma^{9}, \gamma^{10}]$. The same is true for the cost C

The grid search performs 10-fold cross validation against both parameters and the value inside the heat map is the mean accuracy.

This experiment referenced in Figure 4.13 evaluated the polynomial kernel 4.2 with a degree of 2. Very little can be said for this figure as the accuracy was either very poor or quite high with the greatest mean accuracy of 97.45% found by using the full grid search.

This experiment from Figure 4.14 evaluated the polynomial kernel with a degree of 3. Since the degree of this kernel was greater then the previous experiment carried out in Figure 4.13. Cover's theorem, suggests the data should be more likely to be linearly separable and thus perform better, this was not the cause. perhaps the data used for the most part, was already linearly separable and thus classification did not benefit from a kernel mapping with an increased dimensionality. The mean accuracy of 96.86% found by using the full grid search.

The radial basis function was the final kernel that was evaluated. This kernel takes the input and using the kernel trick to map the data to an infinite dimensional feature space. The heat map for this grid search can be seen in Figure 4.15. This heat map looks considerably different then the other two for the same set of hyper parameters. One can easily see the triangular region in the heat map and how the closer to the center of the region the greater the accuracy. The best accuracy found was 97.8%.

The Table 4.2 shows the best hyper-parameters searched and the corresponding parameter values that performed the best. As seen on some of the heat maps some of the largest accuracy values are repeated. The first set of hyper parameters to maximize the accuracy was recorded. The results show that the RBF kernel had a higher classification performance compared to the

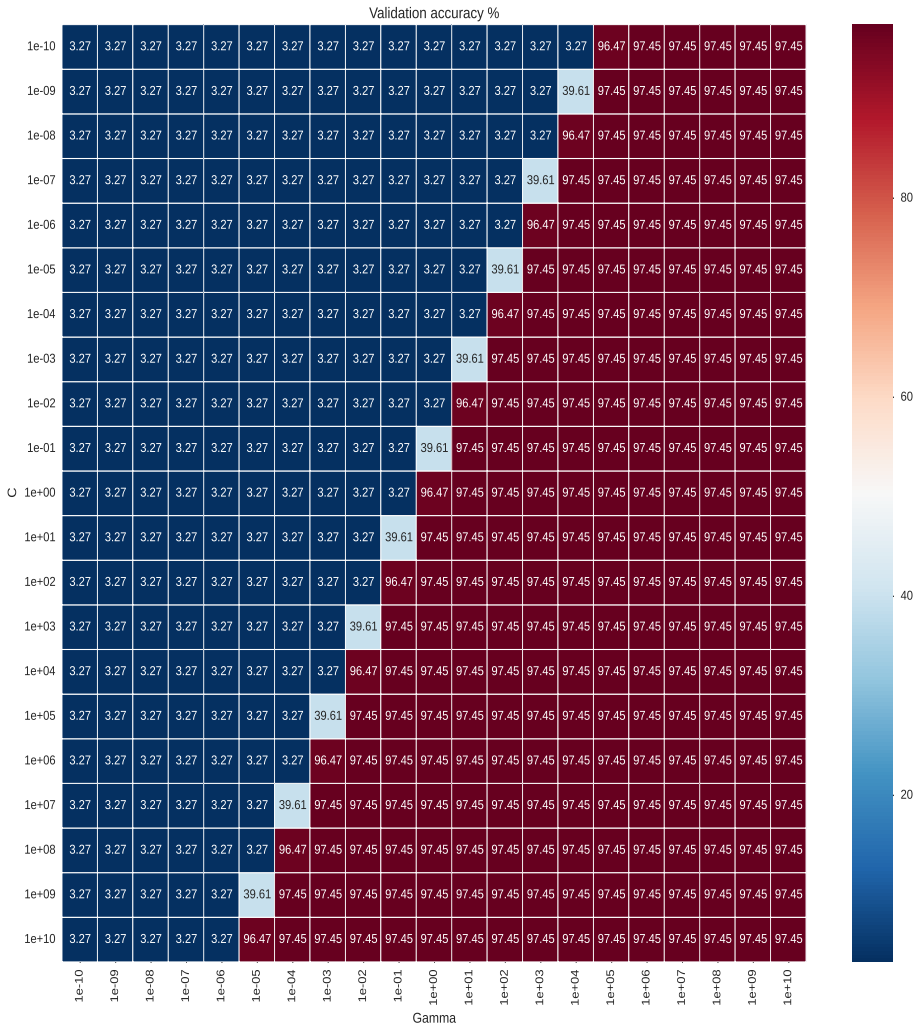


Figure 4.13: Heat Map of Grid Search for polynomial degree 2 parameters

Kernel	Log2C	Log2y	Best Log2C	Best Log2y	Accuracy (%)
Linear	-10,1,10	NA	10	NA	97.1
Poly. (Degree=2)	-10,1,10	-10,1,10	1×10^{-10}	1×10^{6}	97.5
Poly. (Degree=3)	-10,1,10	-10,1,10	1×10^{-10}	1×10^{4}	96.9
Gaussian Kernel	-10,1,10	-10,1,10	100	0.1	97.8

Table 4.2: Hyper-parameter search for several kernels and the affects on classification

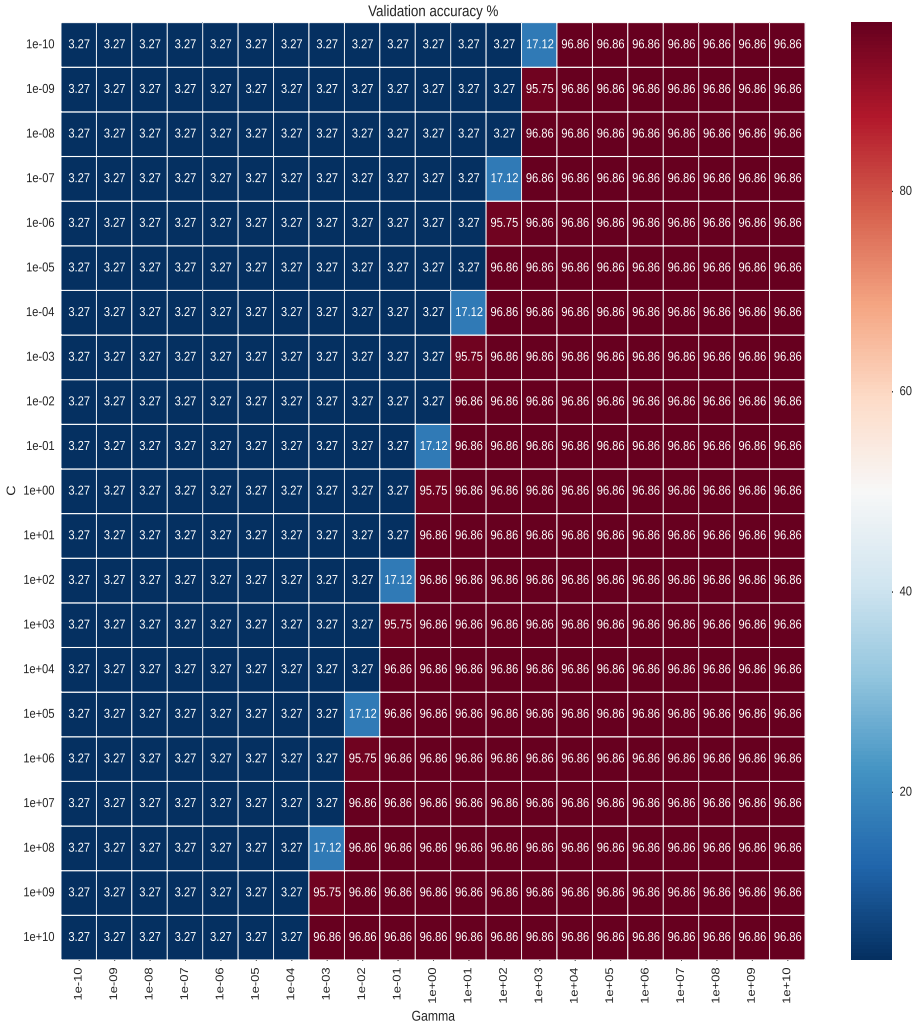


Figure 4.14: Heat Map of Grid Search for polynomial degree 3 parameters

linear and both polynomial kernels of degree 2 and degree 3.

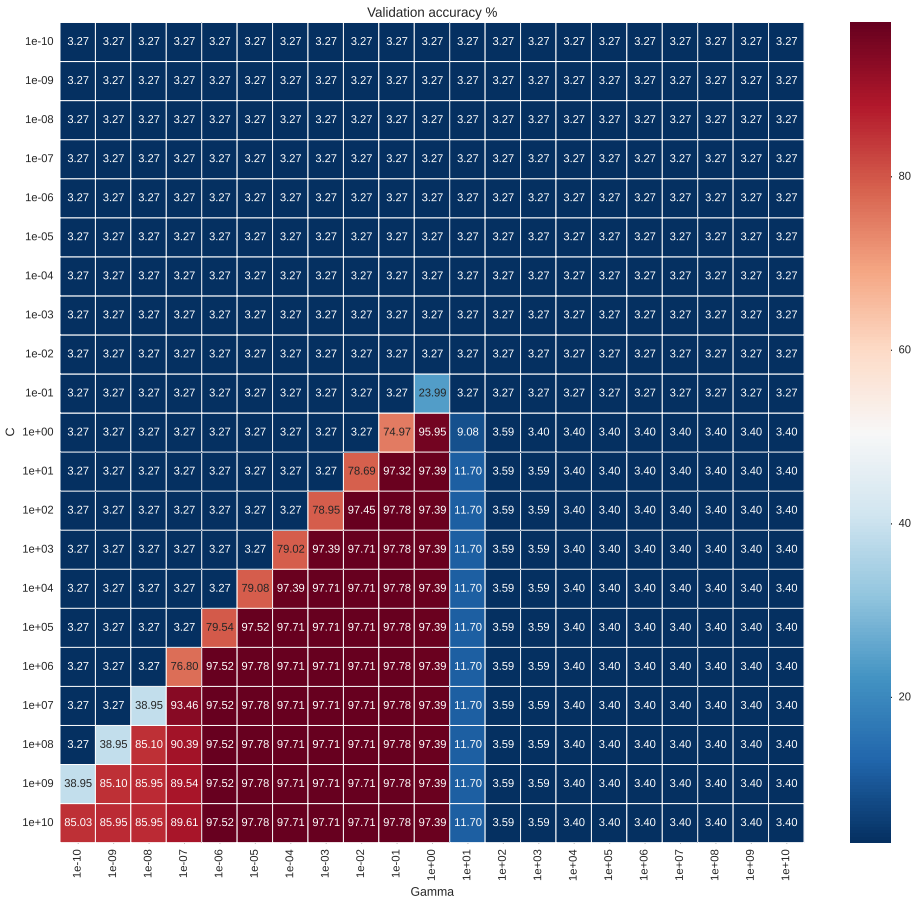


Figure 4.15: Heat Map of Grid Search for RBF kernel

4.5.3 Histogram Intersection and X^2 kernels

Histogram Intersection

A well known problem with the traditional BoW is the lack of spatial information. That is, using a histogram based approach, spatial relationships among features are ignored and the geometry of the scene is lost.

The original approach as proposed by Grauman and Darrell [18] is that given two sets of vectors X and Y, the correspondence between the two sets is computed. Grids over the feature space are computed with an increasing amount of coarseness at each level l which contains 2^l

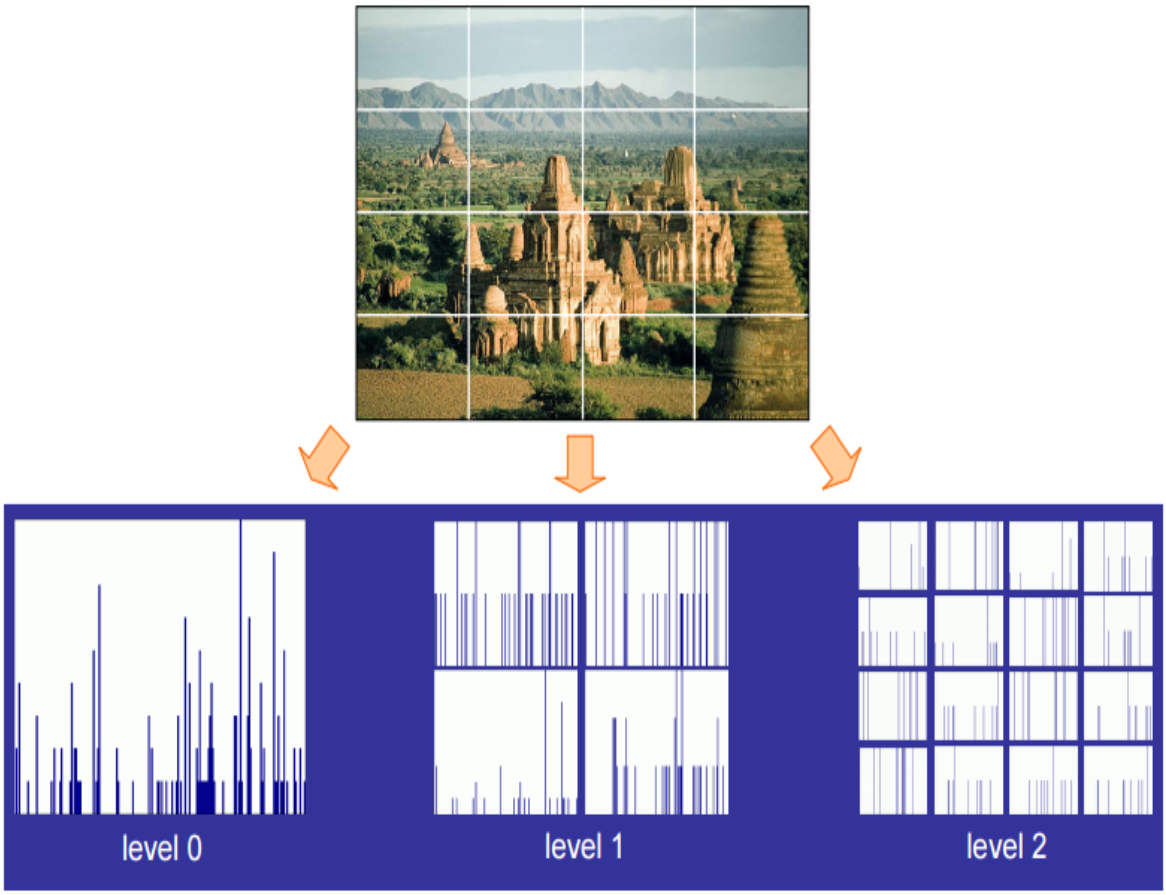


Figure 4.16: Generating spatial pyramids [32]

cells for each resolution $0, ..., L$ for a total of $D = 2^{dl}$ cells. Two points are said to match if they fall into the same cell of the grid. Matches that occur at finer grid levels have a higher weight. The histograms H_X^l and H_Y^l are the points from X and Y that fall into the ith cell of the grid at level l. The weighted sum of matches is given by the histogram intersection function:

$$\mathcal{I}(H_X^l, H_Y^l) = \sum_{i=1}^{D} min(H_X^l, H_Y^l) \tag{4.4}$$

This work was further developed by Lazebnik [32]. The author deviated from the previous work by construction grids on the images and not the entire feature space and thus incorporate weak geometry.

The image is divided into fine sub-regions This process can be see in Figure 4.16. Local histograms within each cell are computed and weighted based on what level of the pyramid the

grid belongs to.

Using the same notation given by Svetlana et al. [32] $\mathcal{I}(H_X^l, H_Y^l)$ is abbreviated to $\mathcal{I}^l$.

$$k^L(X, Y) = \mathcal{I}^L + \sum_{l=0}^{L-1} \frac{1}{2^{L-l}} (\mathcal{I}^l - \mathcal{I}^{l+1}) \tag{4.5}$$

$$= \frac{1}{2^L} \mathcal{I}^0 + \sum_{l=1}^{L} \frac{1}{2^{L-l}} (\mathcal{I}^l - \mathcal{I}^{l+1}) \tag{4.6}$$

The associated weight at level l is set to $\frac{1}{2^{L-l}}$, thus the associated features and respective histograms found at finer resolutions of the pyramid have more weight. Since the histograms at higher levels are given more weight when intersections take place as defined by 4.6, there is a larger similarity score. This results in a scheme that favours matches at higher levels returning a larger score if the intersection takes place in the same grid which introduces *weak geometry*.

$\mathbf{X^2}$ kernel

Another kernel function of interest is the X^2 additive kernel from the Chi-Squared distribution. This kernel function is given by:

$$k(x, x') = -\sum_i \frac{(x_i - y_i)^2}{(x_i + y_i)} \tag{4.7}$$

It is worth noting that while the Histogram Intersection is positive definite (Mercer kernel). The X^2 additive kernel in 4.7 is only conditionally positive definite. Kernel selections that are not positive definite can still be used, and often work well in practice [5]. Not being positive definite means there is no geometrical interpretation and the hyperplane that maximizes the margin discussed in subsection 2.5.4, is no longer guaranteed to be optimal.

4.5.4 Results

The results presented in Table 4.3 show the classification performance using the in-house dataset with all four views. The linear kernel classification accuracy came from Section 4.3. The linear kernel accuracy value is the baseline to compare the other two kernels against as they

Kernel	**Accuracy (%)**
Linear Kernel	95.62
Histogram Intersection Kernel	98.37
X^2 (additive)	98.43

Table 4.3: Comparison of the Histogram Intersection and X^2 kernels

all had the same number of features and same number of words (256 centroids) in the visual vocabulary. The results show that the histogram intersection kernel and the X^2 both perform better than the standard approach that lacks any geometry. The discriminative ability of the HIK approach is greater than the linear kernel which is consistent with [32]. Our work shows that this improvement using weak geometry can be applied to a multiple view system.

The X^2 kernel performed slightly better than the HIK. It is surprisingly as the process of generating the spatial pyramids with increasing weights was **not** used for the X^2 test. This result is quite surprisingly because the only change that took place was swapping the linear kernel for the X^2 kernel and the problem with the lack of spatial information is still present.

4.6 Evaluation of Spatial Histograms and Pooling

There are several methods for improving the standard histogram from the bag of words representation. The most common way is to incorporate weak geometry by using spatial histograms as shown by Grauman & Darrell [18] and Lazebnik et al. [32] as looked at in the previous section. Another technique when using spatial binning is to pool the local encoded features across the spatial regions. An exhaustive evaluation of these different techniques can be found in the work done by Chatfield et al [6].

4.6.1 Spatial Histograms

As introduced in Section 4.5.3 the loss of spatial information is detrimental to the discriminative ability for the image classification task. Weak geometry can be incorporated by the use of spatial histograms by partitioning the image into grids. Once done the local features in each grid area encoded, for example by creating a histogram. An example of this process can be

seen in Figure 4.17 by partitioning the image into a 2×2 grid.

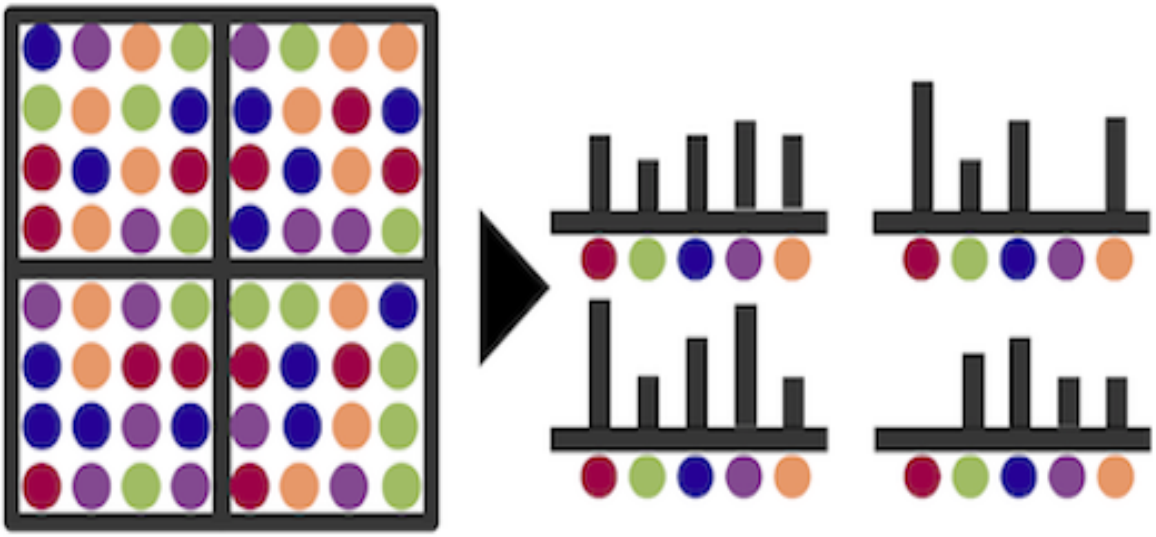


Figure 4.17: An example of spatial histograms [2]

It is really important to point out there are many different flavours to this approach. For example the grid shape need not be $n \times n$. Often vertical or horizontal strips i.e. 3×1 are used or some combination of grid shapes. The most suitable grid shape is often determined empirically. In the case of the HIK using the method described by Lazebnik et al [32] the grid size increased at a constant weight the the histograms were re-weighted based on what level the cell belongs to. This also does not have to be the case. The histograms do not need to be re-weighted by a scale factor determined by the level the cell belongs to. That approach also uses the histogram intersection kernel to find the intersection between sets. This is also not a requirement as any valid kernel can be used. In fact, the classifier is not restricted to be a SVM.

The encoded features in each cell can then be concatenated and used in classification. Using this approach it is often beneficial to normalize the encoded features by each individual cell using the l^1 or l^2 norm and then concatenating each feature encoding to build the full feature vector and possibly normalizing again on the entire feature vector prior to classification. It is also worth pointing out that increasing the number of spatial histograms too much can cause over fitting and harm classification.

[2] SPATIALHISTIM: http://clic.cimec.unitn.it/vsem/content/bovw.html

4.6.2 Feature Pooling

When using the spatial histogram approach the encoded features can be concatenated to form the full feature vector. Another approach known as feature pooling combines the feature encodings in each spatial cell by some operation. The benefit to this is that simple concatenation of encoded features can greatly increase the dimensionality of the representation by a factor of n where n is the number of cells. According to Boureau et al [2], "Pooling is used to achieve invariance to image transformations, more compact representations, and better robustness to noise and clutter."

Some of the most frequently used pooling operations include mean, max and sum pooling. Given an image that is partitioned into M cells with each cell containing encoded features with the BoW histogram model. Each cell would then contain a $1 \times j$ feature vector F, where j is the number of centroids in the visual vocabulary. The following three pooling techniques assume that for every $m \epsilon M$ at least one feature exists in m and thus the required encoding stage can take place.

Mean Pooling:

The mean pooling across M cells is defined as:

$$F_j = \frac{1}{M} \sum_{m=1}^{M} u_{mj} \tag{4.8}$$

Max Pooling

The max pooling across M cells is defined as:

$$F_j = \max\{u_{1j},\ u_{2j},\ \ldots,\ u_{Mj}\} \tag{4.9}$$

Sum Pooling:

The sum pooling across M cells is defined as:

$$F_j = \sum_{m=1}^{M} u_{mj} \tag{4.10}$$

4.6.3 Results

For all of the experiments conducted in this section, any technique that requires spatial histograms or pooling was done by taking the image and partitioning the image into 2×2 grids for level, $L = 1$. The experiments conducted do not exceed beyond the first level into finer grids, as it actually harms classification performance. All of the experiment involving spatial histogram and pooling were used along size the original BoW model. All of the feature vectors are composed of the image at level $L = 0$ and concatenated with the pooling or spatial histogram approach at level $L = 1$. Prior to applying any pooling operation, the feature vector of each cell in the grid is l^2 normalized

Keeping consistent with previous experiments, a dictionary size of 256 centroids was used. This means that the dimensionality of the feature vector using any of the pooling approaches is $1 \times 512n$, where n is the number of views. The spatial histogram which contains four cells at $L = 1$ is concatenated with the original BoW representation at $L = 0$. The resulting feature vector dimensionality for this technique is $1 \times 1280n$

The first experiment performed evaluates the impact of using spatial histograms and feature pooling techniques on the in-house dataset as seen in Figure 4.18. There was no positive results on performance caused by the various pooling approaches. The only accuracy improvement noted, was from using the spatial augmentation/pyramid approach in which there was a slight increase in classification accuracy for the in-house dataset.

The following experiment was repeated using the 8 view CalTech 3D set as seen in Figure 4.19. The results show that the pooling techniques perform slightly better but the greatest improvement in classification performance came from the spatial histogram. Using the spatial histogram approach had a considerable boost in classification performance by over 11%. The downside to the spatial histogram approach in particular with this dataset is that it consists of 8 views and thus the size of the feature vector that has a size of 10240

As the spatial histogram had such a considerable improvement in classification accuracy as seen in Figure 4.19 a comparison showing improvements to classification at the object level was performed as seen in Figure 4.20. Classification accuracy for each object in the dataset was improved. The greatest improvement came from the accuracy of the toaster class. Classifi-

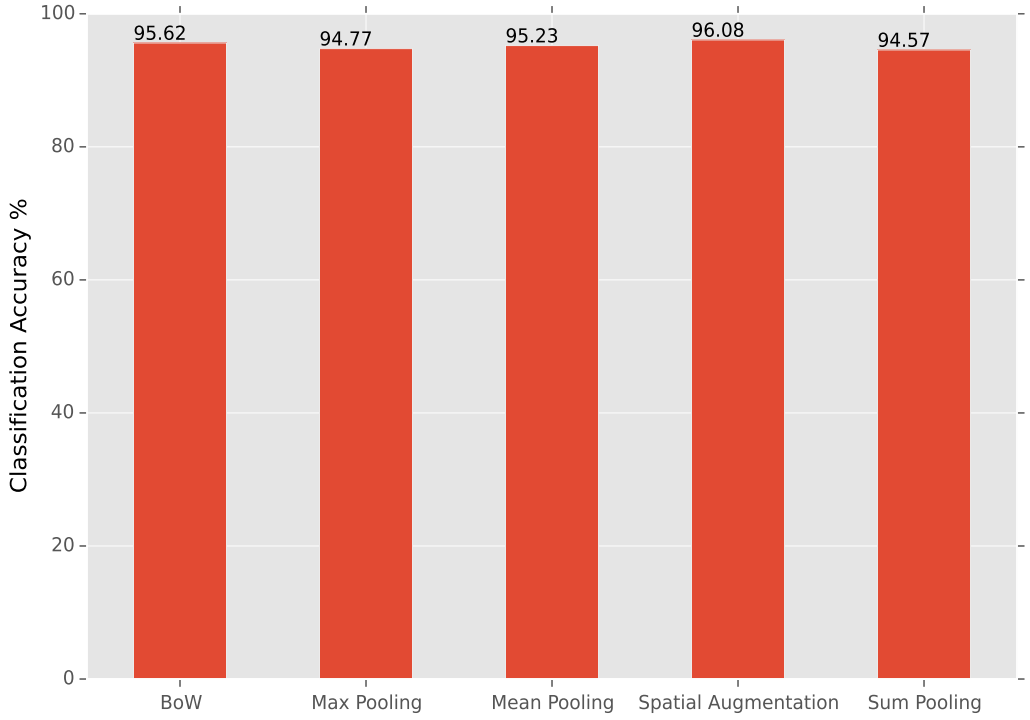


Figure 4.18: Spatial histograms and feature pooling applied to in-house dataset

cation of this object was considerably worse then the other object categories in the dataset and as a result was really lowering the overall classification mean. The use of spatial histograms improved classification of the toaster object from 57.41% to 75.93%.

In this chapter an evaluation of using different techniques with the goal of improving classification by incorporating weak geometry and using a more robust image representation by feature pooling was evaluated. Most works that use these techniques focus on improving classification using a single view. This chapter showed that these techniques that can improve classification with a single view could be incorporated to a multiple view framework and further improve classification. While most of these techniques are often applied to computer vision tasks, the improvements can generalize to machine vision tasks.

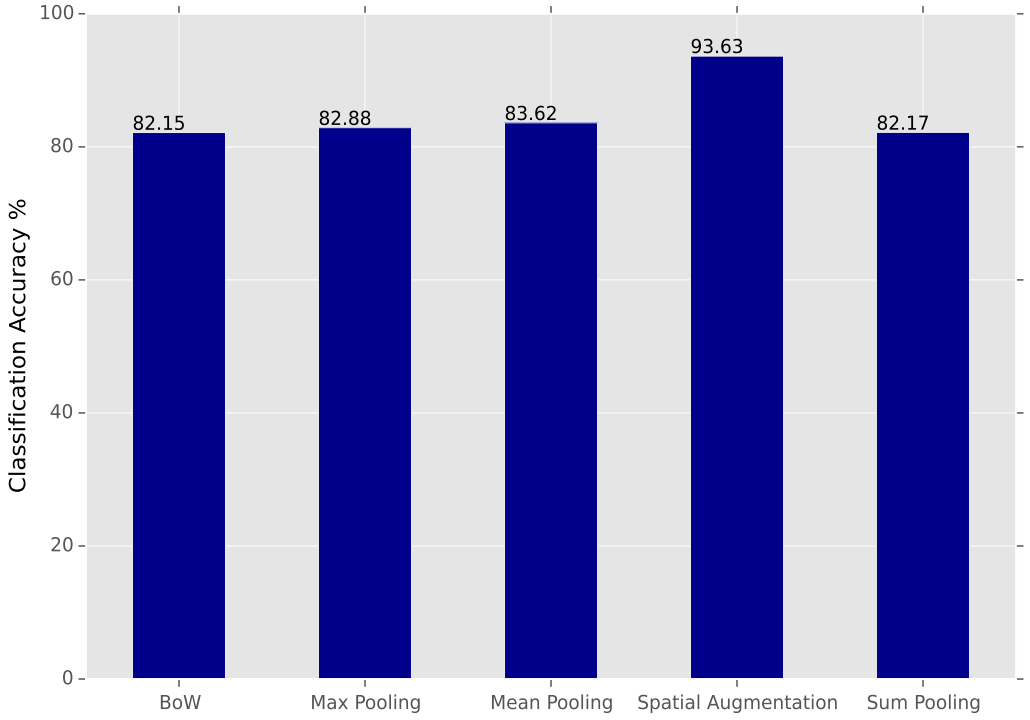


Figure 4.19: Spatial histograms and feature pooling applied to CalTech 3D dataset

4.7 Evaluation of Descriptor Encoding Strategies

In the past few years there have been advancements in representing an image. The traditional BoW model encodes the descriptors by matching the features to the visual vocabulary and the final image representation is a histogram. A more modern approach of descriptor encoding known as a Vector of Locally Aggregated Descriptors (VLAD) was first introduced by Jegou et al. [26]. The VLAD representation replaces the histogram representation by aggregating the difference between a descriptor assigned to a centroid by Nearest Neighbour (NN) assignment along all of its dimensions. A more detailed explanation of this approach can be found in Section 2.6. Feature encoding using the VLAD method is commonly refereed to in literature as containing first order image statistics. Whereas the traditional BoW with a histogram is said to contain zero order image statistics.

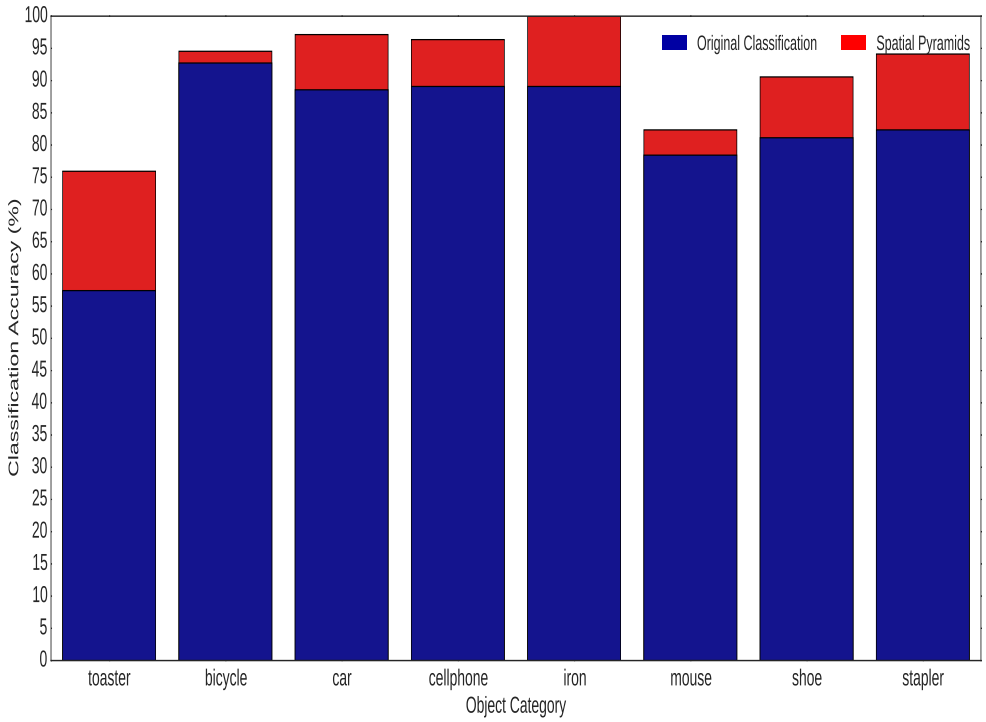


Figure 4.20: Classification improvements when adding spatial histograms

4.7.1 Results

The results shown in Figure 4.21 evaluates the VLAD encoding classification results as a function of dictionary size. This experiment also evaluates the results of different normalization strategies. The results for this experiment show that without any normalization the classification is quite good. In fact at the smallest dictionary size it outperforms the baseline BoW approach. The one notable issue is the shape of the classification line is jagged. Unlike almost all of the experiments presented the trend of increasing the dictionary size has a positive impact on performance. This makes it hard to predict the effects of the number of centroids on classification.

The next two evaluations compare the use of the l^2 normalization and power normalization followed by l^2 normalization before training. The results show that applying power normalization at $\alpha = 0.5$ followed by l^2 normalization performs the best. It also happens that this series

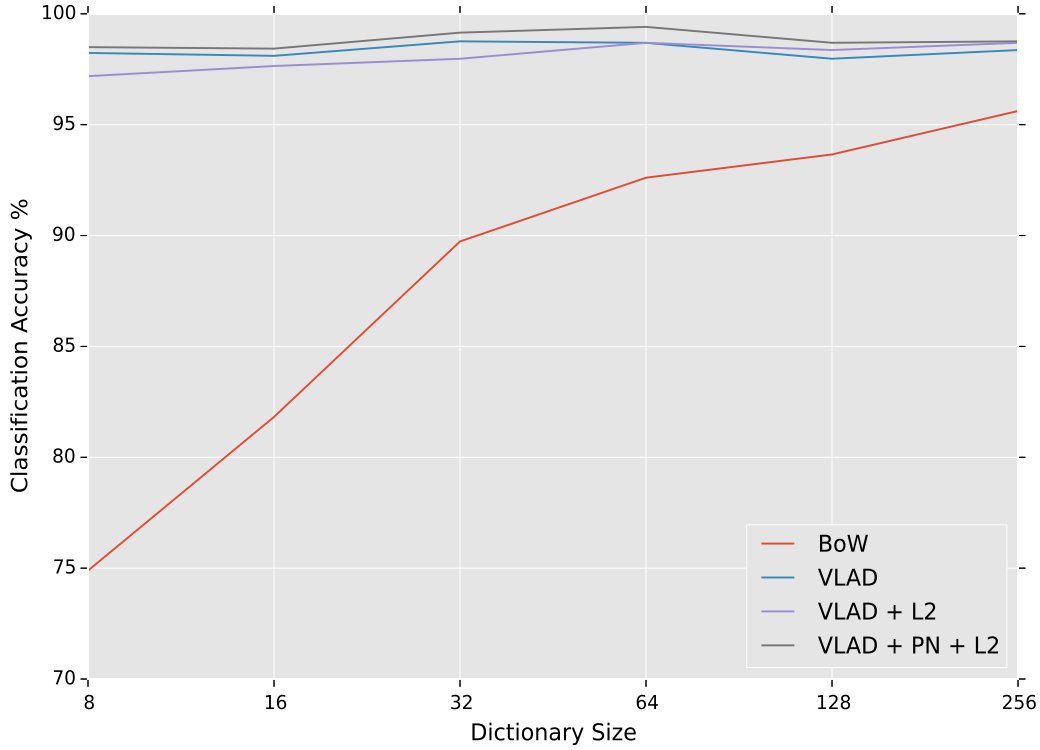


Figure 4.21: The results of the in-house dataset using the VLAD encoding approach

of normalizations appears to be the most stable with the greatest accuracy of 99.4 % using 64 centroids when building the visual vocabulary. All of the VLAD normalization strategies perform better then the BoW for a comparable dictionary size.

One of that natural drawbacks to this method is the size of the feature vector can easily grow to the point where it is unreasonable to work with. The dimensionality of VLAD using this multiple view framework is that using ORB features which have a dimensionality of 32, the resulting feature vector will be of length $1 \times 32ni$ where n is the number of centroids and i is the number of views. Sticking with our baseline for our in-house dataset which has 4 views, with a dictionary size of 256 the length fo the feature vector is 1×32768

For the purpose of comparison, the same experiment was carried out on the CalTech dataset as seen in Figure 4.22. There are some similarities between these results and the results obtained earlier from Figure 4.21. Firstly, the VLAD approach is superior then the regular histogram based BoW. It appears that this experiment shows the importance of normalization prior

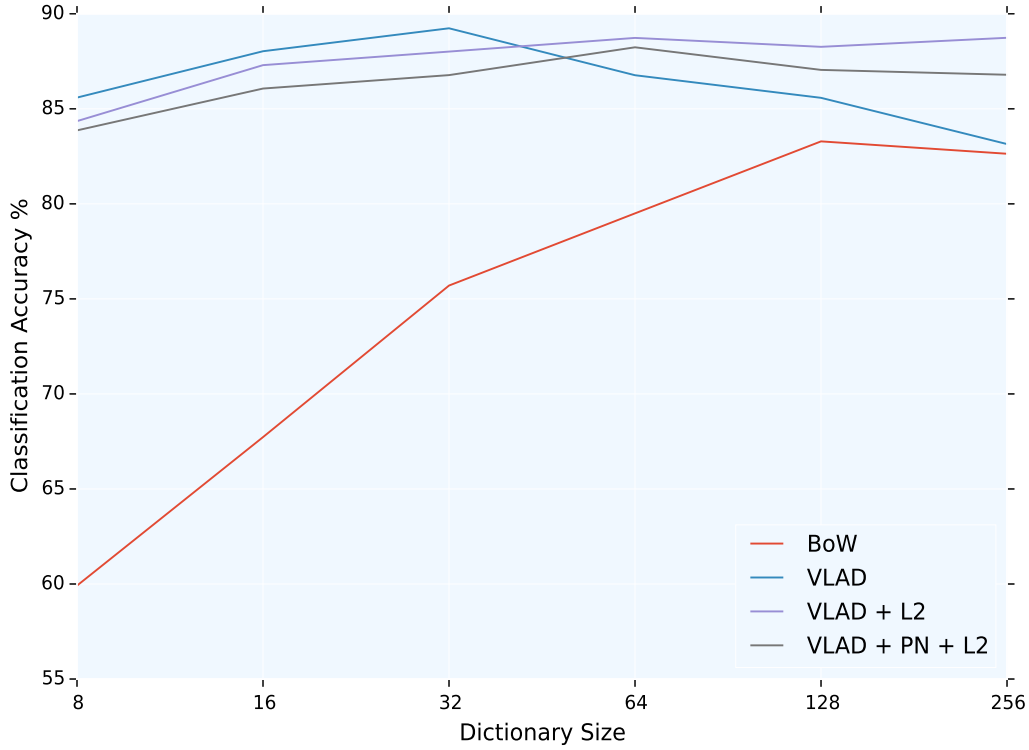


Figure 4.22: Classification of Caltech using the VLAD encoding approach

to classification. While the best results happen to appear for unnormalized feature vectors, at dictionary size 32, it is the most unstable of the two normalized VLAD encodings, and appears to have a fair reduction in classification accuracy as the dictionary size grows.

For the Caltech dataset, the best normalization approach appeared to be l^2 whereas in the in-house set power normalization followed by l^2 appeared to work the best. The results are too similar to make any generalization of the best approach, though it could be argued that the variance in the classification results as a function of the dictionary size it is too unstable.

4.7.2 VLAD and Dimensionality Reduction

Following this the next set of experiments evaluates the effects of dimensionality reduction as discussed in Section 2.9. The VLAD encoding method drastically increasing the dimensionality of the feature vector by a factor of n, where n is the dimensionality of the descriptor,

resulting in an increase in the training and testing time.

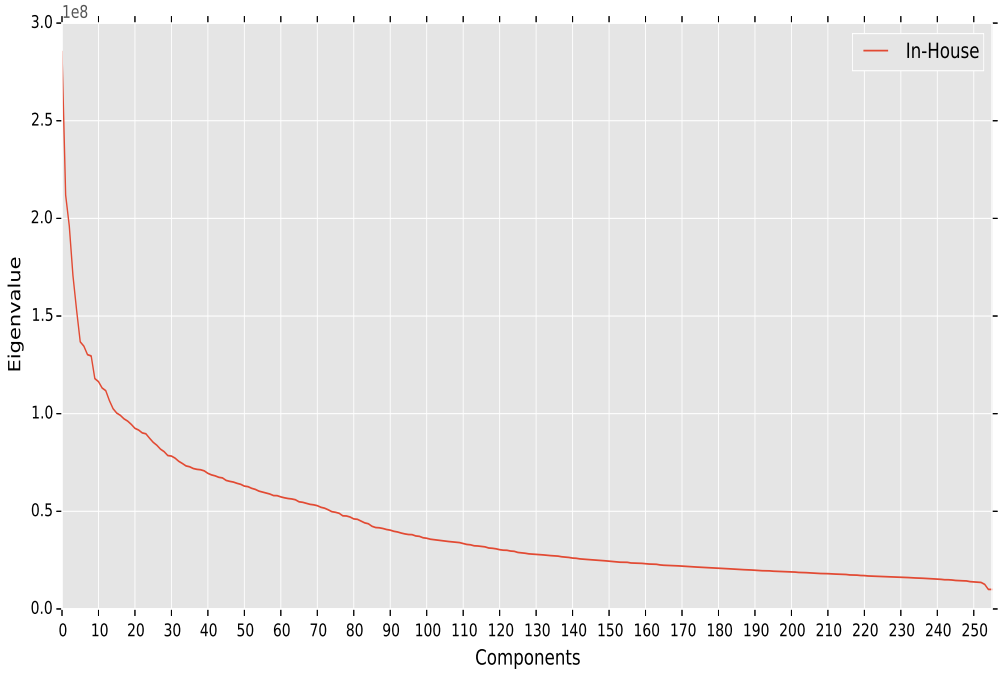


Figure 4.23: In-House eigenvalue distribution of PCs on feature vectors using VLAD encodings

Both the figures 4.23 and 4.24 show the eigenvalues as a function of the number of Principal Components (PCs). The eigenvalues and corresponding eigenvectors are sorted in descending order. The eigenvalue explains how much of the variance in contained within each principal component. Looking at the In-House eigenvalue distribution in 4.23 the majority of variance is contained within the first 100 components in which case the eigenvalues start to level off. This indicates that the image representation contains a lot of redundant information. The CalTech

Dataset	**Dimensionality Reduction D to → D'**	**Accuracy (%)**
In-House	-	98.76
In-House	32 768 → 256	96.53
CalTech	-	88.74
CalTech	65 536 → 256	64.85

Table 4.4: Effects of Dimensionality Reduction when using VLAD encodings

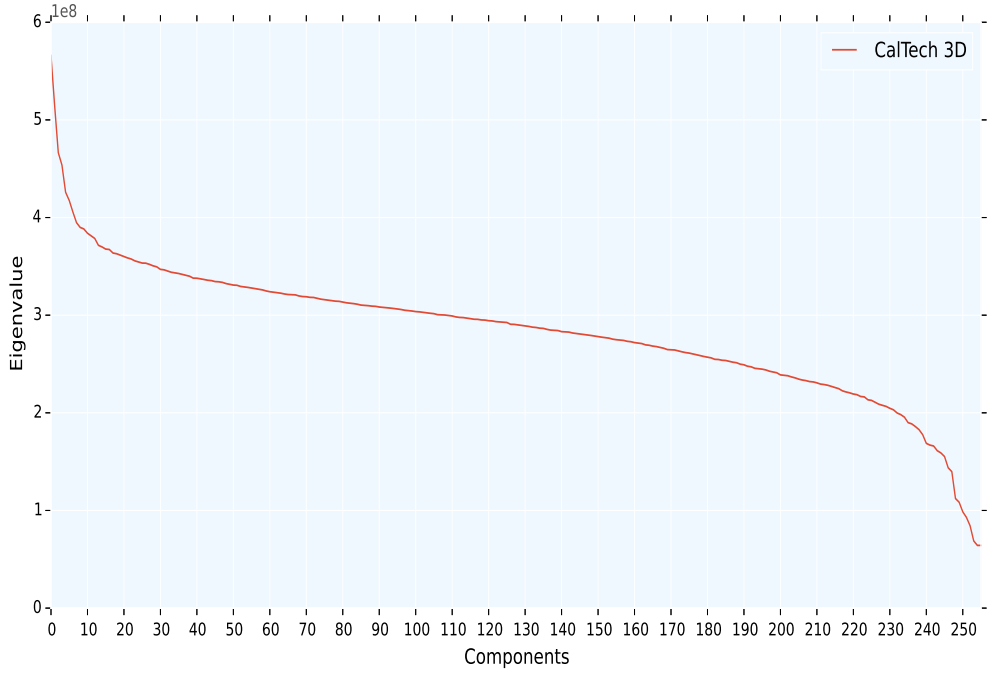


Figure 4.24: CalTech eigenvalue distribution of PCs on feature vectors using VLAD

dataset on other hand in 4.24 does not level off quickly and the data distribution cannot be well explained by first i PCs.

Judging by the two eigenvalue distribution figures it would likely appear that dimensionality reduction would have a greater negative result on classification performance for the CalTech dataset as compared to the in-house dataset. This is confirmed in Table 4.4

4.8 Triangulation Embedding

Triangulation Embedding is a powerful vector representation of an image introduced by Jegou and Zisserman [27] for the image search task. This method is discussed in Section 2.7. Their work is adapted for the image classification task within a multiple view framework.

4.8.1 Triangulation Embedding Classification Results

The first set of classification results compares classification accuracy using Triangulation Embedding and other methods discussed throughout this thesis, shown in Table 4.5. The classification accuracy using this method has an accuracy of 99.28% on the in-house dataset and an accuracy of 94.38% on the CalTech dataset. On both datasets this method outperforms all the other approaches.

Approach	**Accuracy (%)**
BoW + l^2	95.62
Histogram Intersection Kernel	98.37
X^2 (additive)	98.43
Spatial Pyramids	96.08
VLAD + l^2	98.76
$\Sigma R(x)$	96.08
$\Phi_{\triangle}$	99.28

Table 4.5: Comparison of the Triangulation Embedding method on the in-house dataset

Approach	**Accuracy (%)**
BoW +l^2	82.15
Spatial Pyramids	93.63
VLAD + l^2	88.74
$\Sigma R(x)$	83.53
$\Phi_{\triangle}$	94.38

Table 4.6: Comparison of the Triangulation Embedding method on the CalTech 3D set

4.8.2 Triangulation Embedding analysis

In computer vision, images are represented by feature vectors. These representations can be compared to one another by using a similarity metric. These types of metrics take two vectors as inputs and returns a scalar value. For example, within an SVM framework, image similarity is evaluated using a kernel function that also outputs a scalar. One of the most common similarity metrics is the cosine similarity also defined as a linear kernel (assuming l^2 normed inputs) mentioned in equation 4.1. The similarity between two feature vectors x_1 and x_2 should

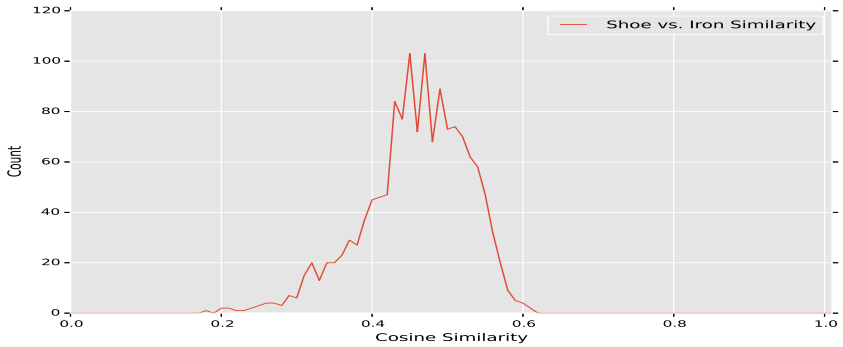


Figure 4.25: Similarity scores histogram of Shoe vs. Iron classes using BoW

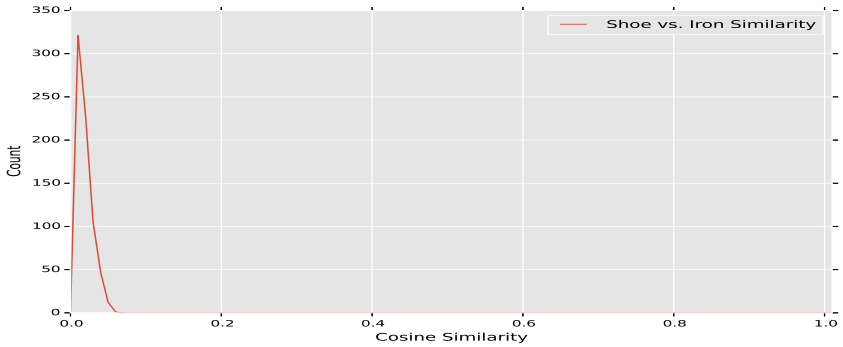


Figure 4.26: Similarity score histogram of Shoe vs. Iron classes using Triangulation Embedding

be close to unity when the images are similar. Conversely, a lower similarity value would be returned when the two feature vectors are dissimilar, which would likely be the case when the feature vectors belong to different classes. Focusing on the classification problem, this means that feature vectors belonging to the the same class should, in general return a larger similarity score when they belong to the same class and lower scores when they belong to different classes.

Looking at the results presented in Section 4.8.1, the triangulation embedding method consistently outperforms the original BoW approach. A more detailed comparison on where the classification fails can be seen in the confusion matrices shown in Figure 4.10 and Figure 4.11.

Focusing on the Shoe vs. Iron classes, by looking at their confusion matrices and improvement in classification accuracy can be seen. Taking the feature vectors of both classes, the cosine similarity among Shoe vs. Iron classes was computed and a histogram of these scores

was created. The first histogram in Figure 4.25 using the BoW has many scores with relatively larger similarity scores compared to the Triangulation Embedding method in Figure 4.26. The feature vectors compared belong to different classes, yet the BoW method shoes there is a relatively large amount of similarity even though the feature vectors belong to different classes. This is not an ideal situation. The histogram shown in Figure 4.26 using the Triangulation Embedding method has many of the scores being much lower, implying the feature vectors belonging to the two different classes are more dissimilar. The takeaway from this analysis is that the Triangulation Embedding method produces a more discriminative feature vector as shown by the similarity scores that are frequently used in image classification and retrieval tasks.

4.8.3 Triangulation Embedding, recovering from camera downtime

A major issue that plagues many vision systems is unexpected downtime. Any piece of computer hardware can fail to function unexpectedly. The likelihood of hardware failure is greatly increased in industrial settings as the hardware may be exposed to physical damage, extreme temperatures and dirty environments to name but a few. This poses a major risk to the manufacturer as they may have to halt production.

Approach	**Accuracy (%)**
BoW	94.51
BoW + l^2	93.72
BoW + PN	96.60
$\Phi_{\triangle}$	98.56
$\Phi_{\triangle}$ + l^2	98.56
$\Phi_{\triangle}$ + PN	98.89

Table 4.7: Comparison of the Triangulation Embedding method on the in-house dataset

To evaluate how robust the system was at handling random camera downtime was simulated. The final feature vector in our framework is the concatenation of n sub-feature vectors derived from the n views.vector is derived To represent a missing camera input one of the sub-feature vectors was replaced with a zero, effectively reducing the system to $n - 1$ views. Its worth noting this was not done on the learning set to generate the visual vocabulary.

The results evaluated in this section use a somewhat different form of a SVM. The software

Approach	Accuracy (%)
BoW	75.59
BoW + l^2	75.57
BoW + PN	74.87
$\Phi_\triangle$	91.69
$\Phi_\triangle$ + l^2	91.21
$\Phi_\triangle$ + PN	91.94

Table 4.8: Comparison of the Triangulation Embedding method on the CalTech dataset

package is known as LibLinear [13]. As the name suggests, this SVM implementation only supports linear kernels but has the advantage of being very fast as it scales almost linearly based on the number of samples.

The results presented in Table 4.7 and Table 4.8 show two things: Firstly, On both datasets the drop in calcification accuracy was reduced by using the Triangulation Embedding method over the BoW method. Secondly, applying Power Normalization (PN) with $\alpha = 0.5$ in lieu of l^2 normalization further improves classification accuracy.

4.9 Running Times of Feature Encoding Methods

There were three main feature encoding methods discussed in this thesis: Histogram encoding for the standard BoW, the Vectors of Locally Aggregated Descriptors (VLAD) and the Triangulation Embedding method. The experiments were conducted on an Intel i7 desktop with 32 GB of RAM. The implementation was done in Python 3.4 using the OpenCV library and the NumPy module for many of the calculations.

Approach	Encoding Rate (images/sec)
BoW	18.64
VLAD	17.78
$\Phi_\triangle$	12.90

Table 4.9: Comparison of the feature encoding rates

The encoding rates for these methods are presented in Table 4.9. These rates also include the time it takes to extract the ORB descriptors. By including the feature extraction times to

the encoding rate, it gives a more meaningful sense of how fast feature vectors can be built for the purpose of running through a classifier. The Triangulation Embedding performed the best in terms of classification accuracy and also has an encoding rate that is fast enough to be useful within the proposed framework. Using the proposed four view setup with each image having a resolution of 1280×720, it takes approximately 0.31 seconds to generate a feature vector.

The processing times could be improved by moving from Python which is a scripting language to using a pure C/C++ implementation. Further optimization can also be achieved using parallel processing and GPU computing, something that will be left to future work.

Chapter 5

Conclusions and Future Work

5.1 Summary

The work contained within this thesis is a framework for creating a machine vision system for correct product classification of very similar items. The type of system designed is for the purpose of error-proofing. In the field of machine vision this type of system to reduce errors is commonly referred to as Poka-yoke, which the direct Japanese to English translation is “mistake proofing.”

This work studies and evaluates complex computer vision approaches and applies them to machine vision problems in an effort to bridge the gap between the two fields. The methodology contained within this framework uses advances in binary features and feature encoding, where most earlier literature focuses on feature matching with binary features, or feature encoding using traditional non-binary features. Most prior works are focused on the task of image retrieval not image classification. The solution presented within this framework can allow for on-line learning of new object classes during regular factory operation and avoids using physical measurements or expert level knowledge, as is the case with most machine vision systems.

The feature pooling method of spatial augmentation performed the best, while being the most common in literature. The newer methods of mean and max pooling, fell short with respect to improvements in classification. The spatial augmentation approach performed the best.

The results of VLAD encoding, while relatively new and impressive in the literature, did

not prove to be very robust for binary descriptors. In fact, reading literature one would assume the VLAD approach was an easy out of the box solution for improving the BoW model, yet this was not immediately realized as normalization is an essential requirement for performance and there was no uniform normalization that was consistent within the literature. Having these requirements that are not overtly clear makes one wonder the generalization capabilities of the method between datasets.

The in-house dataset itself, while we believe in its validity, perhaps more exciting challenge would come from additional object classes. If more parts were available or additional parts that serve a different purpose on the vehicle could further demonstrate the usefulness of this framework.

While various image encoding strategies using 2D images were studied there was also some experimentation with using depth information. Mentioned earlier in this thesis, the difficulty of using depth information for 3D representation. Experimentation with depth reconstruction using feature matching with calibrated cameras and the popular Structure From Motion (SFM) technique performed poorly. It was likely the result that the object itself sits on a white background, and the features detected on the part were unlikely to match across multiple frames with a high enough confidence. Without strongly matched features across multiple frames, the ability to generate an acceptable point cloud was compromised.

The embedding stage of the triangulation embedding approach, while explained in limited detail by the original authors had room for exploration of various dimensionality reduction techniques. The one resulting in the best accuracy was found using PCA through the Singular Value Decomposition (SVD) approach. There was also a considerable improvement using descriptor whitening when compared to simple summation aggregation $\sum R(x)$ for triangulation embedding. This leads to the question on further improving on the benefits of descriptor whitening and the influence feature descriptors have on each other.

Finally, the results show that the system is robust enough to handle camera failure with a minimal decrease in accuracy. A vision system that can handle unexpected camera failures provides a great benefit for users, as production will not have to stop in effort to maintain quality control.

5.2 Future Work

Granted this framework is robust enough to provide a meaningful low cost industrial solution, but further improvements remain. The integration of other types of sensors to capture other features, such as a depth sensor would be of particular interest. Another area of interest that is left to future work would be the use of Convolutional Neural Networks. As of late they have exceeded many previous benchmarks for image classification and retrieval tasks. Despite the difficulty in network architecture and the large quantity of training images needed, it would be interesting to see what classification improvements, if any, could be provided.

Bibliography

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[2] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118, 2010.

[3] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[4] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010*, pages 778–792. Springer, 2010.

[5] Olivier Chapelle, Patrick Haffner, and Vladimir N Vapnik. Support vector machines for histogram-based image classification. *Neural Networks, IEEE Transactions on*, 10(5):1055–1064, 1999.

[6] Ken Chatfield, Victor Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. 2011.

[7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[8] Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *Electronic Computers, IEEE Transactions on*, (3):326–334, 1965.

[9] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[10] Thanh-Toan Do, Quang D Tran, and Ngai-Man Cheung. Faemb: a function approximation-based embedding method for image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3556–3564, 2015.

[11] Kyle Doerr, Jagath Samarabandu, and Xianbin Wang. Efficient object classification using multiple views in manufacturing environments. In *Information and Automation for Sustainability (ICIAfS), 2014 7th International Conference on*, pages 1–5. IEEE, 2014.

[12] C. Eggert, S. Romberg, and R. Lienhart. Improving vlad: Hierarchical coding and a refined local coordinate system. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 3018–3022, Oct 2014.

[13] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[14] Xiang Fu, S. Purushotham, D. Xu, Jian Li, and C.-C.J. Kuo. Hierarchical bag-of-words model for joint multi-view object representation and classification. In *Signal Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pages 1–6, Dec 2012.

[15] Ali Ghodsi. Dimensionality reduction a short tutorial. *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada*, 2006.

[16] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[17] Ronald L Graham and F Frances Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324–331, 1983.

[18] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1458–1465. IEEE, 2005.

[19] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[20] Marti A. Hearst, ST Dumais, E Osman, John Platt, and Bernhard Scholkopf. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18–28, 1998.

[21] Chinmay Hegde, Aswin C Sankaranarayanan, Wotao Yin, and Richard G Baraniuk. Numax: A convex approach for learning near-isometric linear embeddings. *IEEE Transactions on Signal Processing*, 2015.

[22] Yongzhen Huang, Zifeng Wu, Liang Wang, and Tieniu Tan. Feature coding in image classification: A comprehensive study. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(3):493–506, 2014.

[23] Hervé Jégou and Ondřej Chum. Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening. In *Computer Vision–ECCV 2012*, pages 774–787. Springer, 2012.

[24] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1169–1176. IEEE, 2009.

[25] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE, 2010.

[26] Hervé Jégou, Florent Perronnin, Matthijs Douze, Javier Sanchez, Pablo Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1704–1716, 2012.

[27] Hervé Jégou and Andrew Zisserman. Triangulation embedding and democratic aggregation for image search. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3310–3317. IEEE, 2014.

[28] Tak-Eun Kim and Myoung Ho Kim. Improving the search accuracy of the vlad through weighted aggregation of local descriptors. *Journal of Visual Communication and Image Representation*, 31:237–252, 2015.

[29] Markus C Knauer, Jurgen Kaminski, and Gerd Hausler. Phase measuring deflectometry: a new approach to measure specular free-form surfaces. In *Photonics Europe*, pages 366–376. International Society for Optics and Photonics, 2004.

[30] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. Hough transform and 3d surf for robust three dimensional classification. In *Computer Vision–ECCV 2010*, pages 589–602. Springer, 2010.

[31] Guillaume Lavoué. Bag of Words and Local Spectral Descriptor for 3D Partial Shape Retrieval. In Eurographics, editor, *Eurographics Workshop on 3D Object Retrieval (3DOR)*, May 2011.

[32] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.

[33] Ziqiong Liu, Shengjin Wang, and Qi Tian. Fine-residual vlad for image retrieval. *Neurocomputing*, 173:1183–1191, 2016.

[34] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[35] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[36] T Mitchell. *Machine Learning*. McGraw Hill, 1st edition, 1997.

[37] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.

[38] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *arXiv preprint arXiv:1405.4506*, 2014.

[39] John C Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods*, pages 185–208. MIT press, 1999.

[40] Erik Rodner, Doaa Hegazy, and Joachim Denzler. Multiple kernel gaussian process classification for generic 3d object recognition. In *Image and Vision Computing New Zealand (IVCNZ), 2010 25th International Conference of*, pages 1–8. IEEE, 2010.

[41] Paul L Rosin. Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291–307, 1999.

[42] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430–443. Springer, 2006.

[43] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[44] S Russel and P Norvig. *Machine Learning*. McGraw Hill, 1st edition, 1997.

[45] Silvio Savarese and Li Fei-Fei. 3d generic object categorization, localization and pose estimation. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[46] Roberto Toldo, Umberto Castellani, and Andrea Fusiello. A bag of words approach for 3d object categorization. In *Computer Vision/Computer Graphics CollaborationTechniques*, pages 116–127. Springer, 2009.

[47] Yutaka Usui and Katsuya Kondo. 3d object recognition based on confidence lut of sift feature distance. In *Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on*, pages 293–297. IEEE, 2010.

[48] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[49] Kai Yu and Tong Zhang. Improved local coordinate coding using local tangents. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1215–1222, 2010.

Curriculum Vitae

Name:	Kyle Doerr
Post-Secondary Education and Degrees:	2012-Present M.E.Sc. Software Engineering Electrical and Computer Engineering University of Western Ontario London, Ontario, Canada 2012-2013 Masters of Engineering (coursework) Software Engineering Electrical and Computer Engineering University of Western Ontario Graduated 2012 B.E.Sc. Software Engineering with Professional Internship Electrical and Computer Engineering University of Western Ontario
Honours and Awards:	UWO Graduate Scholarship
Related Work Experience:	Teaching Assistant 2013-2014 The University of Western Ontario SE 2250 - Software Construction SE 2205 - Algorithms and Data Structures for Object-Oriented Design

Publications:

Doerr, Kyle, Jagath Samarabandu, and Xianbin Wang."Efficient object classification using multiple views in manufacturing environments." Information and Automation for Sustainability (ICIAfS), 2014 7th International Conference on. IEEE, 2014.

Doerr, Kyle, Jagath Samarabandu, and Xianbin Wang."Improving multi-view image classification using higher order information and triangulation embedding." Information and Automation for Sustainability (ICIAfS), 2014 7th International Conference on. IEEE, 2014.