IN-ORCHARD IMAGING OF PECAN WEEVIL AND EFFICIENT IDENTIFICATION USING ORTHOGONAL POLAR MOMENT DESCRIPTORS

By

Aaron John Franzen

Bachelor of Science in Agricultural Engineering
University of Nebraska–Lincoln
Lincoln, Nebraska
2003

Master of Science in Mechanical and Aerospace Engineering
University of Florida
Gainesville, Florida
2007

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 2015

IN-ORCHARD IMAGING OF PECAN WEEVIL AND EFFICIENT IDENTIFICATION USING ORTHOGONAL POLAR MOMENT DESCRIPTORS

Dissertation Approved:

Dr. Paul Weckler

Co-Committee Chair / Advisor

Dr. Ning Wang

Co-Committee Chair / Advisor

Dr. Carol Jones

Member

Dr. Mike Smith

Outside Member

Name: Aaron John Franzen

Date of Degree: December, 2015

Title of Study: In-Orchard imaging of pecan weevil and efficient identification using orthogonal polar moment descriptors

Major Field: Biosystems & Agricultural Engineering

Abstract:

The pecan weevil is considered the most harmful late season pest of pecan and requires population monitoring in orchards for proper pest management practices. In this research, algorithms for detecting and identifying pecan weevil are developed based on machine vision and pattern recognition components. Instrumented traps were designed and constructed to collect images of insects as they enter a pecan weevil trap. The instrumented traps were fitted to trees in a pecan orchard and left to collect images through a late pecan growing season. These images were processed by computer algorithm to be uniform, then used to test and train an insect detection classifier system that can predict whether an insect is present in the trap for each image.

Images containing insects were further processed to extract only the shape of insect silhouettes. This silhouettes were then used to extract Zernike Moment, Pseudo-Zernike Moment, Fourier-Mellin Moment, and MPEG Angular Rotary Transformation shape description features. The shape descriptors were combined into feature vectors before being used to train classifiers that can discern pecan weevil from non-pecan weevil insects. Pecan weevil identification was shown to be over 98% accuracy depending on the shape descriptors used.

Two algorithms for reducing the number of features in the shape descriptor feature vector were developed based on concepts of Principal Component Analysis and Fisher Multiple Discriminant Analysis. The two algorithms were then used to select the features from the entire set of insect shape descriptors in order to reduce the number of features that need to be stored and used in classification. These two methods were able to reduce the number of shape descriptors from over 1000 to as few as 25 while maintaining classification accuracy above 95%.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS & ABREVIATIONS

API	Application Programming Interface
ART	MPEG-7 Angular Radial Transform
BIC	Bad Image Capture
BIC-BM	BIC-Backlight Malfunction
BIC-BM1	BIC-Backlight Malfunction Type 1
BIC-BM2	BIC-Backlight Malfunction Type 2
BIC-OS	BIC-Over Saturated
BIC-SF	BIC-Shutter Failure
BMPs	Best Management Practices
COTS	Commercial, Off-The-Shelf
FA	Factor Analysis
FMDA	Fisher Multiple Discriminant Analysis
FMM	Fourier-Mellin Moments
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
GPIO	General Purpose Input-Output
GIC	Good Image Capture
GIC-NI	GIC-No Insect
GIC-IP	GIC-Insect Present
ISM	Industrial, Scientific, & Military
KNN	K-Nearest Neighbor
MCC	Matthews Correlation Coefficient
NB	Naive Bayes
OCR	Optical Character Recognition
PCA	Principle Component Analysis
pZM	Pseudo-Zernike Moments
RAM	Random Access Memory
RGB	Red, Green, Blue

RLE	Run-Length Encoding
ROM	Read-Only Memory
RSSI	Received Signal Strength Indicator
RX	Reciever
SVM	Support Vector Machine
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TTR	True Positive Rate
TX	Transmitter
TX-RX	Transmitter<–>Reciever
UART	Universal Asyncronous Reciever Transmitter
UML	Unified Modeling Language
μC	Microcontroller
WSN	Wireless Sensor Network
ZM	Zernike Moments

CHAPTER I

INTRODUCTION

In 2014, United States pecan exports were worth over $500 million with the value of Oklahoma production exceeding $20 million(USDA, 2015). The pecan is consistently in the top three US tree nuts, in both annual harvest size and value. As one of the few tree nuts that is indigenous to the United States, pecans are subject to pressure from pests that have adapted to prefer pecan as food source and reproductive host, akin to the relationship in the date palm-palm weevil and the maize-corn borer host-pest pairs.

The pecan weevil is a striking example of an adversarial indigenous host-pest relationship. The pecan weevil has evolved to possess a long, tube shaped mouth part that it uses to chew through the pecan shuck and shell, eat the nut meat, and deposit eggs inside the nut. Weevil larvae eat what remains of the nut meat before boring a larger hole through the shell and falling to the ground below. While some feeding damage is acceptable for USDA quality grading of pecans, larval damage is considered a serious defect and can result in entire loads of pecans being rejected by a processor. Current year economic losses are limited to feeding and larval damage.

However, failure to control weevils in the current year results in larger populations of weevils in subsequent years.

The pecan weevil is considered one of the most destructive insect pest of pecans, potentially causing complete crop loss in years with large pest populations in untreated orchards. Unlike many other pests of pecans, the pecan weevil emerges and does its damage during the late growing season, after much of the variable costs of production have already been incurred. While many production costs have been incurred by the late season, growers also have more management information at their disposal, including yield and market price estimates for the season, as well as knowledge of the overall balance sheet. Armed with the aforementioned information, growers are able to make an informed decision on whether to treat for pecan weevils at all, or how many pesticide applications should be applied.

Current Best Management Practices (BMPs) suggest that pecan growers should place traps throughout an orchard and monitor for adult pecan weevil emergence from the time the nuts reach the gel stage through the shuck splitting stage (Mulder et al., 2012). Pesticide application is prescribed when certain loss thresholds are reached, depending on the intended market, yield estimates, and market values. Although trapping weevils is not labor intensive for small growers or backyard producers, larger orchards may require an inordinate amount of time and labor for this task. An automated system for monitoring pecan weevil emergence would free that time and labor resources for other tasks.

The long term goal of this project is to move toward such an automated system. As electronics technology continues to get smaller, more powerful, and more power efficient, the thought of a wireless monitoring system is becoming closer to reality. Previous work at OSU has developed image processing techniques that are capable of discerning the pecan weevil from other insects that occur in the orchard

(Ashaghathra, 2008). The combination of such image processing capabilities and the now available electronic components allow for a system to be constructed that can capture images of insects in traps, process the images, and report the results over a wireless link to a base station or even to the growers home. But, just as users complain about short battery life on their mobile devices, growers will complain about poor battery life in a weevil counting system, making it important that such a system be designed such that one doesn't spend as much time changing batteries as would be spent counting weevils.

1.1 RESEARCH OBJECTIVES

Past research and literature suggests that the pecan weevil can be discerned from other insects through image processing techniques in a laboratory environment with dead, manually positioned insects, whereas the performance of such identification techniques has never been evaluated using living insects in natural poses. Image or region histograms have been used to give a partial description of the scene at hand, with the potential to rule out the presence of an insect in the image frame. Orthogonal polar moment features have also been shown as a very promising feature for shape identification and classification, including insect shape and optical character recognition. Combinations of features comprised of different descriptors have been successfully combined into a single feature vector to successfully recognize patterns in many data-mining and bio-informatics applications. Subsequently, these feature vectors have been reduced in dimension without incurring a significant reduction in classification performance through techniques such as Principle Component Analysis (PCA) and Fisher Multiple Discriminant Analysis (FMDA). Furthermore, clustering and classification techniques such as K-Nearest Neighbor (KNN),

Naive Bayes (NB), and Support Vector Machine (SVM) methods have been proven successful as methods to identify group or class membership based on diverse combinations of features in many research areas.

This study aims to combine advances in image processing, feature space reduction, and automatic membership classification in order to provide accurate, fast identification of pecan weevils for future electronic monitoring applications. The objectives of the study can be summarized as:

A. To evaluate the performance of image histograms as the feature-space for insect presence detection in field conditions

B. To propose a new shape-description and identification kernel that is suitable for in-situ insect shape description and identification

C. To evaluate Principle Component Analysis (PCA) and Fisher Multiple Discriminant Analysis (FMDA) dimensional reduction techniques with the aforementioned shape-description kernel (Objective B) for the potential of reducing computation needs in insect identification

D. To apply Naive Bayes (NB) and Support Vector Machine (SVM) classification methods to both homogeneous and mixed feature set classification of insects using the shape description kernel.

1.2 RESEARCH HYPOTHESIS

The hypotheses to be tested in this study follow from the research objectives, and follow as:

A. Images collected with both ambient and back-lighting contain enough information in their intensity histograms to determine if insects are present

B. Uniform processing of insect images, combined with orthogonal moment shape descriptors, can be used to identify pecan weevils among other insects

C. Zernike Moments (ZM), Pseudo-Zernike Moments (pZM), Fourier-Mellin Moments (FMM), and MPEG-7 Angular Radial Transform (ART) can be used in reduced dimensional form to identify a pecan weevil with a reduction in required computation using different classification methods.

CHAPTER II

REVIEW OF LITERATURE

2.1 PECAN WEEVIL BIOLOGY AND MANAGEMENT

2.1.1 *Pecan Weevil Lifecycle*

The pecan weevil, *Curculio caryae* (Horn's taxonomy), of the order *Coleoptera*, is a small beetle that is an obligate feeder of tree nuts. Pecan weevils are known to attack *Carya* species including hickory and the preferred host of pecan, *Carya illinoinensis*. Pecan weevils have also been observed to attack the Persian walnut, *Juglans regia* (Mulder et al., 2012).

Pecan weevils reach 9-13mm in body length as adults. They have a long mouth part, or proboscis, that ranges from 3/4 to 5/4 of the body length. Females typically possess a longer proboscis. Adult pecan weevils are typically light brown in color. Adult pecan weevils are shown in Figure 2.1 (Mulder et al., 2012).

Pecan weevils in the larval stage are a milky yellow, legless grub with a reddish-brown ring on the mouth end. Pecan weevil larvae feed on nut meat during the late summer and early autumn before boring a hole through the shell and falling to the ground below. Upon larval emergence from the nut, the pecan weevil grub has reached an approximate length of 15 mm. The hole created in the side of the

(a) Male pecan weevil.

(b) Female with longer probiscus.

Figure 2.1: Pecan weevils on pecan nuts.

nut is the signature of larval damage, and is typically 3mm in diameter (Mulder et al., 2012).

The pecan weevil life cycle lasts from one to three years when including all life stages, from egg placement (oviposition) to death as an adult. Beginning approximately five days after adult emergence, female pecan weevils bore small (less than 1mm diameter) holes in nut shells and lay eggs inside the shells starting at early water stage and continuing through shuck split. The female pecan weevil typically lays three to four eggs in each nut and will avoid nuts in which another pecan weevil has previously oviposited. Smith and Mulder (2009) found that in areas of heavy infestation, eggs were present at the same time as third-instar larva.

Eggs hatch inside the nuts and the larvae mature through four instars of development. The first three instars last around 13 days each with feeding done throughout. The fourth instar includes approximately five days of feeding, but continues for one or two years without feeding the larvae pupates. After the first five days of fourth instar feeding, the pecan weevil larvae exits through the previously mentioned borehole, drops to the ground, and prepares for overwintering. Pecan weevil larvae enter

the soil under the pecan canopy and form a semi-impermeable earthen pupal cell. The larvae enter a state of diapause in this pupal shell where they will remain for one or two years before pupating. In the diapause state, the larvae exhibit nearly zero respiration or metabolism and are well protected in the earthen cell. After pupation and moulting, the adult weevil often remains in the ground for an additional year before emerging to mate and feed (Mulder and Grantham, 2007; Mulder et al., 2012; Cottrell and Wood, 2003, 2008).

Upon emergence as adults, pecan weevils enter the canopy to feed. Research has shown that, although weevils can fly, the majority crawl to the tree trunk and climb into the canopy. After emergence, pecan weevils normally live for 15-30 days with females living longer than males. In improved pecan cultivation with large nuts, females have been observed to survive up to 56 days. Pecan weevils use their mouth parts to chew through the shuck and feed on the nut kernel. Feeding injury, with a shallow depth of insertion, can occur on single nuts multiple times. Additionally, there appears to be no inhibition of further feeding due to said shallow insertion. Contrarily, deep insertion and oviposition with testa penetration does inhibit further deep penetration for oviposition (Schraer et al., 1998).

Pecan weevils begin to feed and lay eggs from the onset of the nut's water stage through shuck split. Nuts with feeding damage prior to the gel stage are aborted by the tree, falling to the ground before ripening. Pecan trees also abort nuts with eggs and larva deposited before the gel stage. In this case, the absence of further kernel development prevents the larvae from developing and ultimately causes larval death. Nuts that have larval damage often remain on the tree well beyond shuck split, taking on a dark brown or black color.

(a) Feeding damage on gel stage pecan.

(b) Larval exit holes on pecans at shell hardening stage.

(c) Mature pecans with larval damage.

Figure 2.2: Nuts damaged by pecan weevil.

Both feeding and larval damage cause loss in the pecan crop value in orchards that have been controlled for pecan weevil in the past, although larval damage is typically much greater. Nuts with feeding damage have small spots on the shell (Figure 2.2(a)), and often have dark spots with a bitter taste in the kernel. Feeding damage does not cause extreme loss in value, but will result in nuts grading out lower on the USDA scale, thus reducing prices paid. Most nuts with larval damage are worthless, as there is very little nut meat remaining (Figure 2.2(c)). Nuts with larval damage that do have nut meat remaining are still unsaleable (Figure 2.2(b)). Additionally, the presence of nuts with larval damage may result in bulk rejection of a truck or bag of nuts (Mulder et al., 2012).

2.1.2 *Pest Management for Pecan Weevil*

Control of pecan weevils has become an economical imperative for farmers since the advent of monoculture orchards of improved variety pecans, including intensive production in the natural range of both the pecan tree and by extension, the pecan weevil. Pecan weevils have not been observed to migrate long distances

from the soil from which they emerge as adults. As such, the pest has yet to become a significant problem in many areas of production outside of the native pecan range, including much of New Mexico (recently declared eradicated), Texas, and Mexico. Conversely, areas inside the native range of pecans typically require treatment for weevils, especially in improved orchards in the southeastern United States, including Oklahoma (Mulder et al. (2012, 2003)).

Pecan weevil management has evolved over the time that pecans have been cultivated by humans. In the earliest years of intensive cultivation, pesticides for pecan weevil had yet to be developed. After identifying the pecan weevils as a late season pest, the earliest method for control was physically impacting the tree canopy, causing the weevils to drop to the ground. This was typically achieved using large groups of workers with long poles who were tasked with impacting the canopy in the entire orchard to remove the weevils. Technological advances, including pesticides and the airblast chemical applicator, allow for more complete control of the pecan weevil, but have only been available together since the early 1960s (Cottrell and Wood, 2003).

Today, most of the control of pecan weevils is achieved with airblast sprayer application of pesticides during the 5-7 day window between adult emergence and oviposition in nuts. Complete control of weevils can be achieved with airblast application of carbaryl or certain pyrethroids every 7-10 days from gel set to shuck split, without regard for adult weevil emergence, market value of the crop, or projected yield. For many growers, especially those targeting retail in-shell markets, this system makes sense. However, growers targeting processed pecan markets might prefer to make chemical applications only when: adult weevils are present in the orchard, yield is expected to be acceptable, and market price is high enough to justify the

expense. In such cases, growers must monitor the emergence of adults from the time the nuts reach gel stage through shuck split (Mulder et al., 2012).

While control methods have improved since the physical knock down methods, monitoring methods have likewise changed since the development of Best Management Practices (BMPs) for pecan production. Early monitoring for pecan weevils was similar to the early control method. Sheets were used to cover the orchard floor and the canopy was jarred with poles to knock weevils from the tree. The number of weevils that fell to the sheet could then be used to make a management decision on the need for pesticide application. Other systems involve using traps that attract pecan weevils to capture and count the insects during emergence. Either containment, deception, chemicals, or the weevils own instincts are used to lure them to the trap. Containment style traps include the wire-cone emergence trap, which captures all the weevils that emerge from the circular base of a wire-mesh cone which funnels all insects from the area to its (Figure 2.3 (a)). Deception is used in the case of the Tedders trap. Noting that most pecan weevils crawl to the tree trunk to enter the canopy, and hypothesizing that the weevils are drawn to dark column structures by instinct, the tedders trap is a dark painted tower with a modified boll weevil trap at the apex. It is placed between trees in the orchard, while the tree trunks are painted or wrapped white (Figure 2.3 (b)). Weevils enter these traps under the assumption that they are climbing a tree. Circle traps use the pecan weevils instinct to climb the trunk without needing deception in order to attract the weevil. Circle traps are screen or wire mesh funnels with a modified boll weevil trap at the apex, and are mounted directly on the tree trunk (Figure 2.3 (c)). In this case, the weevil is intercepted on his or her route to the canopy. Chemical or pheromone attractants have been used with minimal success in both Tedders and circle traps (Mulder et al., 2003).

(a) Pyramid traps.

(b) Tedders traps (dark tower in forground).

(c) Circle trap on tree trunk.

Figure 2.3: Pecan Weevil Traps

Different weevil pressure thresholds exist, depending on the region and if improved or native pecans are grown, to determine if chemical applications should be made. Most of these thresholds take into account the aforementioned market price, yield potential, and number of weevils captured per unit area and time. Many local extension services in pecan growing regions offer advice on thresholds for chemical application. It is important to consider that all previous inputs prior to the pecan weevil emergence season are expended capital, and must be considered when deciding whether to treat an orchard. Additionally, making the decision not to treat for pecan weevil in year N may have negative consequences in years $N+1$ and $N+2$ due to the potential for a large number of adult pecan weevils emerging in those latter years (Mulder et al., 2012).

Other methods for management aside from pesticides have been investigated, but have not seen much commercial use due to cost, technological deficiency, or lack of efficacy. Larval, rather than adult, control has proven difficult due to the hard earthen cell that the larvae inhabit, the difficulty injecting chemicals deep enough in the soil to reach the larvae, and the fact that during diapause, respiration and metabolism rates are reduced to such a level that intoxication is extremely unlikely.

In terms of biological control, both fungi and nematodes have been explored as methods to kill pecan weevils. So far, each of these methods has provided poor or unpredictable results, leading to their use only in research settings(Mulder et al., 2012).

2.2 EMBEDDED ELECTRONICS IN ORCHARD ENVIRONMENTS

Work in the area of electronics, sensors, and control systems has been ongoing in orchard crops since the late 1970's. More recently, electronic irrigation control has been a topic of research interest (Dukes and Scholberg, 2005; Fernández et al., 2001, 2008). Fruit and nut growing operations have been areas of much precision agriculture research due to the high value of such crops and the intensive management that is often required. Tumbo et al. (2002) used a combination of laser and ultrasonic sensing to measure the volume of the canopy in a citrus orchard. Aggelopoulou et al. (2011) used machine vision techniques to develop apple yield predictions based on early season flowering, while Oerke et al. (2011) used thermal imaging techniques to detect scab disease on apple tree leaves. Increasingly, Unmanned Systems are being evaluated as potential scouting and monitoring tools for pests, diseases, and stress in many orchard environments, including both utility vehicles (Bergerman et al., 2012; Hamner et al., 2010) and aerial vehicles (Baluja et al., 2012; Coopmans, 2009).

2.2.1 *Embedded Imaging Systems*

Little work has been done in the area of imaging systems for orchard management or monitoring. Most imaging applications in orchard environments have involved

vehicles, including vehicle guidance (Subramanian et al., 2006; Lang, 1999), remote sensing (Berni et al., 2009), or animal monitoring (Moruzzi et al., 2002). Guarnieri et al. (2011) integrated a cellular phone into a coddling moth trap to track emergence of the moth in high temporal precision but used no classification or identification methods other than expert interpretation, whileWen and Guyer (2012) developed an automatic identification system for similar moths that are pests to apples using laboratory-acquired images of the insects, achieving a maximum classification accuracy of 86.6%.

2.2.2 *Wireless Sensor Networks in Orchard Environments*

Wireless sensor networks have seen much more adoption in orchards than have imaging systems. Pierce and Elliott (2008) developed a large scale frost monitoring and weather monitoring network in Washington state. Antonio-Javier Garcia-Sanchez (2011) developed a system based on the 802.14.4 IEEE standard to integrate scalar and video data into a Wireless Sensor Network (WSN). Majone et al. (2013) describes a $5000\mathrm{m}^2$ deployment of 135 soil moisture and 27 temperature sensors in an apple orchard in northern Italy. WSN continues to be a highly active research area as many specialty crops continue to push forward in automation.

2.3 IMAGE PROCESSING AND CLASSIFICATION

2.3.1 *Insect Detection*

Most research on insect detection has been concerned with detecting insect infestation in stored products, mainly using acoustic or dielectric sensing techniques.

Additionally, the majority of efforts have looked to detect "many" insects as opposed to a single insect, or have had no need for a low-power method for insect detection.

Ridgway and Chambers (1996) explored detecting insects in wheat using NIR reflectance spectroscopy. Nordström et al. (2006) explored the detection of flying insects in scenes during machine vision acquisition, but had no concern for power consumption. Vick et al. (1991) patented a method for detecting insects falling into a pitfall sensor through detecting vibration. Franzen et al. (2011) determined that capacitance-based measurements are not sufficient for detecting single insects, both via experiment and finite element simulation. They also found that, in controlled environments, a simple optical beam-break sensor might be feasible as a single insect detector.

2.3.2 *Image Segmentation*

Many methods for segmenting images have been created over the years. Segmentation is an operation in image processing in which different regions of an image are separated from the rest as areas of interest. Segmentation is one of the first steps of feature extraction, and can be based on many different features of interest.

In the case of this study, the feature of interest is ultimately the insects shape or boundary. Boundary recognition is relatively easy with controlled lighting, and only slightly more difficult in the case that the background is uniform and contrast between background and object is sufficient.

In other research in which image processing of insect images is of interest, often more minute details are the features of interest. Watson et al. (2004) presented the system known as DAISY, or Digital Automated Identification System. This

classifier is used to identify species of moths. However, it requires a human to segment regions of the wings and thorax manually.

2.3.3 *Shape Recognition Algorithms*

Ashaghathra (2008) used a combination of shape recognition algorithms based on the boundary or region of a region in an image. Region based recognition algorithms include Zernike Moments, Region Properties, and Geometric Moments. Perimeter based methods include Fourier Descriptors and String Matching. Zernike Moments proved to be the fastest method with the fewest of both type I and type II errors. Using a combination of the methods independently, they were able to correctly identify pecan weevil 100% of the time for a small set of images that were used for both training and testing.

Other shape recognition methods include Pseudo-Zernike Moments, Fourier-Mellin Moments, and statistical pattern matching. Fourier-Mellin and Pseudo-Zernike moments are region based recognition methods similar to Zernike moments. Hosny presents a few variations on these methods, as well as other orthogonal polygon based moments (Hosny, 2007, 2008, 2010a,b,c, 2011a; Hosny et al., 2011; Hosny, 2011b, 2012a,b). Most variations are based either on improving the speed of calculation, or improving the accuracy. This class of moment is commonly used in handwriting or optical character recognition, or as an alternative coding set for image compression and regeneration.

Statistical pattern matching techniques, which may include related areas of fuzzy matching and neural network matching, seek to fit a known set of features from an image into the closest distribution for a category of shape. Most require expert

segmentation or the exact same number of features for every image. As such, these methods are not a good fit for a system that aspires toward automation.

Zernike moments, as used in Ashaghathra (2008) for identifying pecan weevils, belong to a group of polynomial functions that are orthogonal on the unit circle. Orthogononal implies that the integral of the product of two polynomials is zero for any combination of Zernike Moments (ZM). The ZM was originally developed as a method to quantify the abberation of optical systems by Zernike (1934). Since their introduction, the ZM has been used in image processing, more specifically, image compression and reconstruction. They have also been used as descriptors in pattern and shape recognition applications in image processing.

For many years, the ZM was one of the defacto radial moments used in image processing. Zhenjiang (2000) used ZM to identify and grade rose flowers and leaves. Optical Character Recognition (OCR) is one of the most common applications used for comparing performance of shape-recognition algorithms. Khotanzad and Hong (1990); Hosny (2012b); Papakostas et al. (2010) all used OCR to demonstrate the effectiveness of their chosen descriptors.

More recently, researchers have been looking at the performance of other types of orthogonal moments for shape recognition and reconstruction. Pseudo-Zernike Moments (pZM) were created as a modification of the original ZM and shown to provide improved performance in image coding and reconstruction by Bhatia and Wolf (1954). Fourier-Mellin Moments (FMM) were used by Chen et al. (1994). There are many other classes of orthogonal polynomials, including Chebyshev moments (Celebi and Aslandogan, 2005), Hahn moments (Zhu et al., 2007b), Legendre moments (Mukundan and Ramakrishnan, 1995), Hermite moments (Vick et al., 1991), Racah moments (Zhu et al., 2007a), as well as others.

2.3.4 *Reduced Component Methods for Classifiers*

Pattern Classification, as a field, is typically done through the process of preprocessing raw inputs, extracting features from said inputs, and using a classifier to decide which class the raw inputs belong in. As more feature descriptors are added to the representative feature vector of the raw inputs, presumably more information is available to a classifier to make better, or more informed, decisions. In real-world classification problems, additional feature descriptors are useful to an upper limit: when the same amount of information is represented in the feature space that was present in the raw input form, additional feature descriptors tend to only "muddy the waters" with the same or even worse results than simply using the raw inputs. In essence, once a certain number of feature descriptors are included, any additional descriptor that could be added to the set is dependent on descriptors already included (Duda et al., 2012).

In pattern classification problems, it can often be the case that the set of selected feature descriptors are not independent from one another, exhibiting large covariances across descriptors. According to Duda et al. (2012), "component analysis is an unsupervised approach to finding the 'right' features from the data." Principle Component Analysis (PCA), otherwise known as the Harhunen-Loeve Transform, ranks data descriptors by their eigenvalues. Beginning with descriptors with the smallest eigenvalues, descriptors are removed from the model until only those making the largest contributions to the variance remain. This yields a k-dimensional linear subspace that best represents the data based on minimum-square-error calculation. It should not be assumed that reduction in feature space through PCA is always beneficial for classification. If noise is large when compared to the distance between classes, PCA will locate the direction of the noise instead of the signal.

Factor Analysis (FA) is a similar method to PCA used to reduce the dimensionality of data by forming a linear combination of features. Whereas PCA discovers the features that account for the *variance* in the data, FA looks for a lower dimensional set of features that accounts for the *correlations* among features. FA combines features into cluster centers used to represent the data (Duda et al., 2012).

CHAPTER III

MATERIALS AND METHODS

INTRODUCTION

Image processing is a broad term describing the manipulation or quantization of data within an image to extract information that might otherwise not be directly observable. Image processing techniques might seek to improve the appearance of details by increasing contrast between regions, to locate regions of an image that contain objects of interest, to recognize written characters for text extraction from images, and to classify regions that belong to one of many defined groups. In this study, image processing techniques are utilized to determine if an image includes an insect, and if so, to identify the insect species based on its shape. Determining the presence of an insect in each image prior to further processing for insect identification can reduce total processing time in the case that the presence detection step is much less processing intensive than the steps involved in insect identification.

Pattern Recognition is the term used to describe mathematical and statistical methods that are used to decide an object's class membership. Classification methods are usually based on a set of numerical or ordinal values that describe an object.

Images might contain regions of interest that can be described by a list of member pixel coordinates, a list of the boundary pixel coordinates, a mathematical description of the positions of the pixels, or a combination of the previous. A combination of descriptive values that describe a region is known as a feature vector. Feature Vectors can include any number of individual properties that describe the region, such as color, intensity, centroid, major and minor axis, size, and many others. In most cases, some features are more powerful than others in classification problems, and it is desirable to use the features that most effectively serve to discriminate between classes, thus improving classification performance.

The remainder of Chapter 3 describes acquisition of images for the study, the insect presence detection methods, the proposed insect identification kernel, the comparison methods for feature vector performance, and classification methods efficacy in identifying pecan weevil insects.

3.1 INSECT COLLECTION AND IMAGE ACQUISITION

The insect collection and image acquisition system for the study was designed to collect images of pecan weevil and other insects that might be present in pecan orchards in their natural poses. The design intent for the system was that it be integrated with Circle Traps that are currently used by commercial pecan growers for pecan weevil monitoring. The acquisition system was built as a prototype, and was used for only one weevil emergence season. Planned improvements, based on first year performance observations, were not implemented due to factors including drought, infestation of other insects, and the demolition of the original testing site at the Oklahoma State University Botanical Garden and Arboretum.

3.1.1 *Insect Positioning and Imaging System Setup*

The field data acquisition system for this study is comprised of custom built pecan weevil traps that integrate an imaging chamber with back-lighting, camera, and microcontroller in a weather-protected enclosure. The block diagram for the strap system is shown in Figure 3.1. The imaging chamber is a custom-designed path for insects to climb that attaches securely to the cone of a standard bowl weevil trap, built in an hourglass shape that funnels insects to the back-lit imaging area that is placed in front of the camera. The camera is a commercially available consumer-grade camera (Canon Powershot A470, Canon Inc, Ota, Tokyo, Japan) and was selected from available options due to its available "Super-Macro Focus" mode that allows imaging of objects as close as $0.5\,\mathrm{cm}$. The camera was interfaced to an 8-bit microcontroller board (Arduino Pro 328, SparkFun Electronics, Niwot, Colorado, USA) by directly soldering wires for control signals to the user-interface buttons on the back of the camera to control power, auto-focus, and image acquisition. Images were acquired on fixed intervals of 30 or 60 seconds, and were saved directly to the camera's internal SD memory card.

The back-lighting in the imaging chamber is provided by white electroluminescent sheeting placed in the imaging area on the side opposite the camera. The electroluminescent sheeting requires a separate DC-to-AC power supply that provides high-voltage, high-frequency current for its excitation, while the camera requires a DC power supply at $3\,\mathrm{V}$ DC. The DC-to-AC and $3\,\mathrm{V}$ DC power supplies were also interfaced to the microcontroller via direct soldering control signals to the contacts on the user-interface buttons. The microcontroller was powered continuously, while the back-lighting and camera power supplies were disabled when not in use. The traps were operated on $6\,\mathrm{V}$ sealed lead-acid batteries which were replaced and

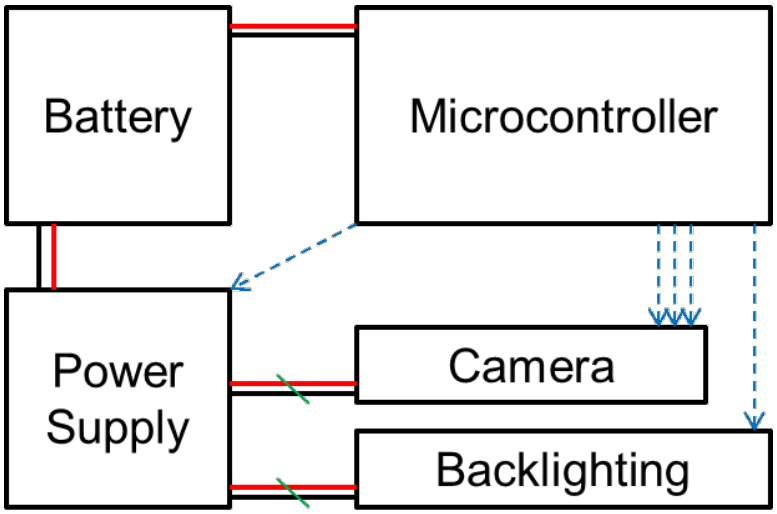


Figure 3.1: Block Diagram of the instrumented pecan weevil traps used for image acquisition. In the image, red/black (black/gray) pair lines represent power with diagonal green (gray) cross line indicating that the power is switchable. Dashed blue (gray) arrows represent control logic lines.

recharged at one to four day intervals. The installed image acquisition trap is shown in Figure 3.2. In all, 73,579 images were collected between the two instrumented traps over a 32-day period, with collection ending when weevils were no longer emerging for the season, as observed via non-instrumented companion traps.

Image processing, pattern recognition testing and training, and statistical analysis for the study were performed offline using a 64-bit commercial software package (MATLAB Release 2014b, The MathWorks Inc., 2014) with the Image Processing and Statistics toolboxes, while graphs were generated using an updated version of the same software (MATLAB Release 2015a, The MathWorks Inc., 2015). The software package was installed on a Dell mobile workstation with an Intel Core I7-2760QM 2.4GHz quad-core processor, 32-GB DDR3 1600MHz Random Access Memory (RAM), and a PCI-Express Solid-State harddrive. The operating system was Windows 8.1 Pro 64-bit.

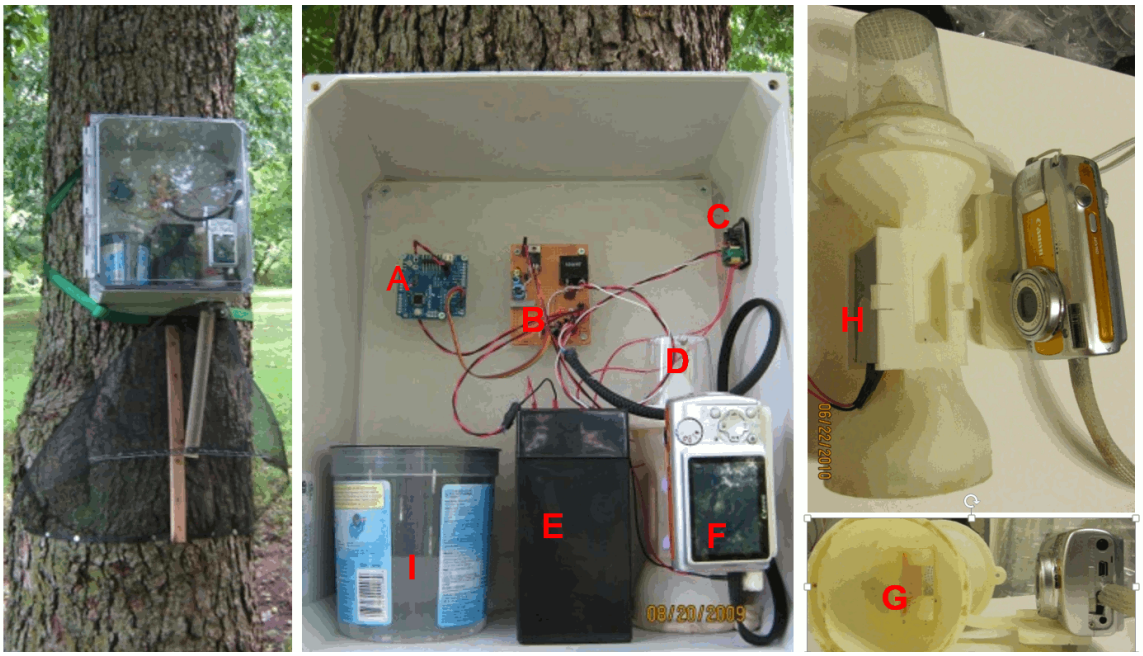
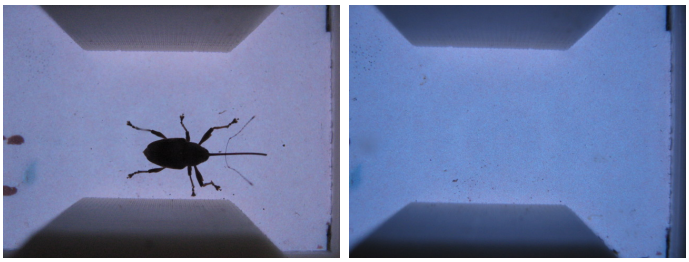


Figure 3.2: The image acquisition trap installed in a pecan orchard. Labeled components include A) microcontroller, B) power regulation and switching, C) Electroluminescent DC-to-AC inverter, D) weevil collection cup, E) battery, F camera, G) imaging chamber, H) Electroluminescent back-light, and I) Desiccant.

3.2 INSECT PRESENCE DETECTION

After the complete image set was collected, it was observed that the images could be categorized into various classes, with further processing only completed for a certain set of classes. Two base classes, Good Image Capture (GIC) and Bad Image Capture (BIC) were established. The two base classes are further broken down into sub-classes. The GIC class was broken into the sub-classes GIC-Insect Present (GIC-IP) and GIC-No Insect (GIC-NI). The BIC class was subsequently broken into four sub-classes: BIC-Backlight Malfunction Type 1 (BIC-BM1), BIC-Backlight Malfunction Type 2 (BIC-BM2), BIC-Over Saturated (BIC-OS), and BIC-Shutter Failure (BIC-SF). Example images for each of the five classes are shown in 3.3 on the following page.

(a) BIC-BM (b) BIC-OS (c) BIC-SF



(d) GIC-IP (e) GIC-NI

Figure 3.3: Example raw images of the five different image classes. The top row, a-c, shows the Bad Image Capture classes, while d-e show the Good Image Capture classes. All images are field-acquired.

3.2.1 *Insect Detection Features*

The images in figure 3.3 have obvious differences in intensity, or brightness, despite being collected from the same scene. The distribution of intensities across an entire image, otherwise known as the intensity histogram, is proposed as a feature vector for classifying images in the the classes GIC-IP, GIC-NI, BIC-BM1, BIC-BM2, BIC-OS, and BIC-SF. Both traditional and cumulative histograms were used in this study.

3.2.1.1 *Traditional Histograms*

The traditional histogram is commonly used as a method to visualize the fit of sampled data to a density function in statistics, or to detect bi-modal distributions

that can be used to segment data. The histogram, m, is defined by dividing the intensity range, $0-255$ in the case of 8-bit digital images, into j bins such that

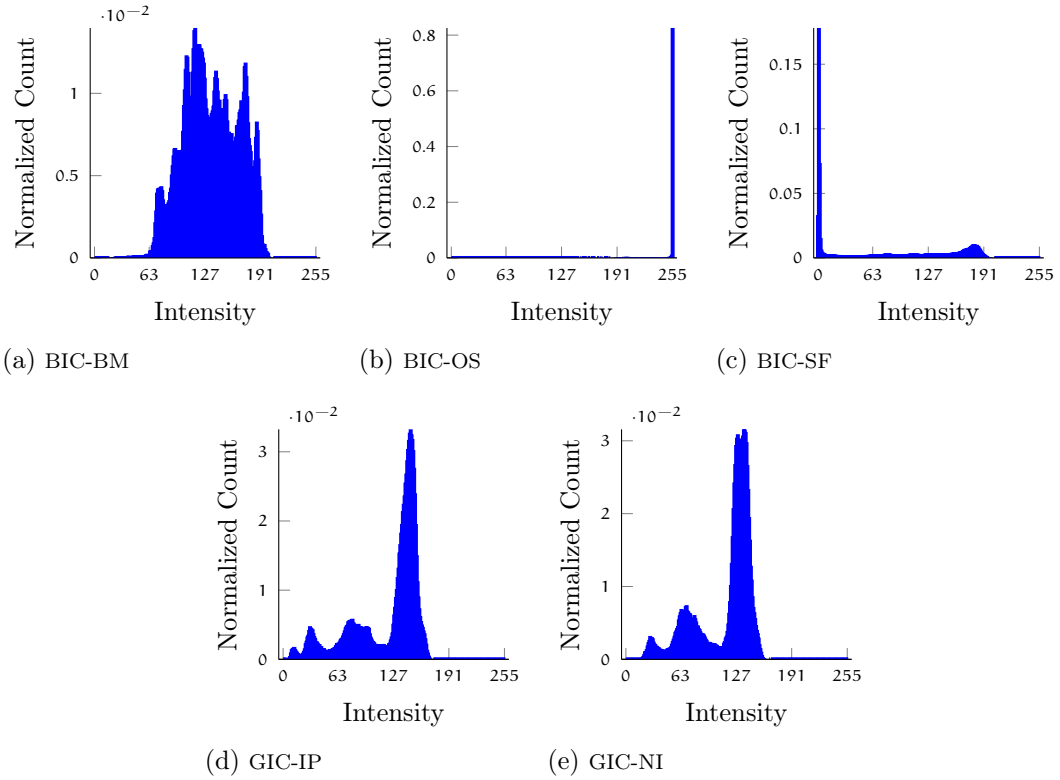
$$n = \sum_{i=1}^{j} m_i, \tag{3.1}$$

where the total number of pixels in the image is n and i is the bin label. The proposed histogram feature vector is normalized such that

$$\frac{1}{n} \sum_{i=1}^{k} m_i \equiv 1. \tag{3.2}$$

This normalization is also known as a relative histogram, and it allows images of different sizes to be used in the same classifier since the effect of image size is removed from the resulting vector. The relative histograms for the images in Figure 3.3 are shown in Figure 3.4. Note that the histogram shapes are quite different in the three BIC class images, but are not of notable difference in the GIC class images. This indicates that it should be easier to classify images between the three BIC classes than those in the GIC classes.

The vast majority of the images captured for the study fall in the GIC class, with a large percentage of the GIC images falling in the GIC-NI sub-class. This is due to the fact that the event of an insect entering the imaging chamber roughly follows a Poisson distribution for rare events. Images in the BIC class, while relatively rare, could provide crucial information to growers or researchers about the status of instrumented camera traps. In field conditions with a well-functioning system, the problem of insect detection would converge to the case of classifying between only the GIC-IP and GIC-NI sub-classes. In practice, the BIC classes could provide useful diagnostic information for a given trap, indicating failure of a camera or



(a) BIC-BM (b) BIC-OS (c) BIC-SF

(d) GIC-IP (e) GIC-NI

Figure 3.4: Image intensity histograms of the images in Figure 3.3. As in Figure 3.3, the top row, a-c, shows the Bad Image Capture classes, while d-e show the Good Image Capture classes.

backlight. Additional failure modes might be found in different implementations of an instrumented trap and could be added to the BIC class.

GIC image histograms as classification features do not differ dramatically in shape, as the major difference in the scene is the presence of an insect, or lack thereof. This results in histogram shapes that are much more similar between the two sub-classes. Figure 3.5 show histograms for two sample images in Figure 3.3 for the GIC-IP and GIC-NI sub-classes. Due to the similarity in histogram shape, it would be expected that there are larger classification errors between these to sub-classes. With the traditional histograms for the GIC-IP and GIC-NI sub-classes, it appears that the

GIC-IP distribution has four modes, or peaks, while the GIC-NI distribution is tri-modal.

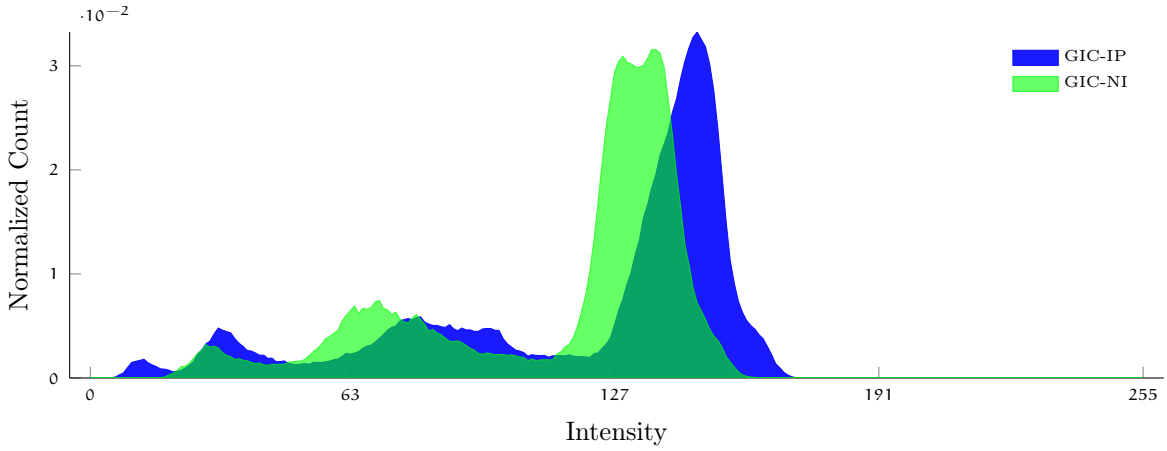


Figure 3.5: GIC class histograms for the sample images in Figure 3.3. GIC-IP is in blue (dark), while GIC-NI is green (lighter). The histograms for these two subclasses are much more similar than those of the BIC class.

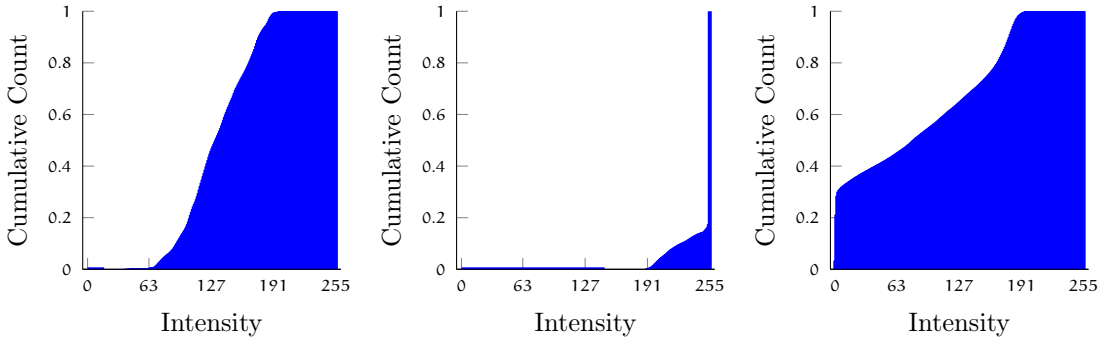
3.2.1.2 *Cumulative Histograms*

It is also possible to visualize an intensity histogram in a cumulative manner, where subsequent bin values are summed as the intensity value increases. This is also known as a cumulative frequency distribution chart, and is similar in concept to the cumulative density function in statistics. The cumulative histogram, M_i, for the histogram m_q is defined mathematically as

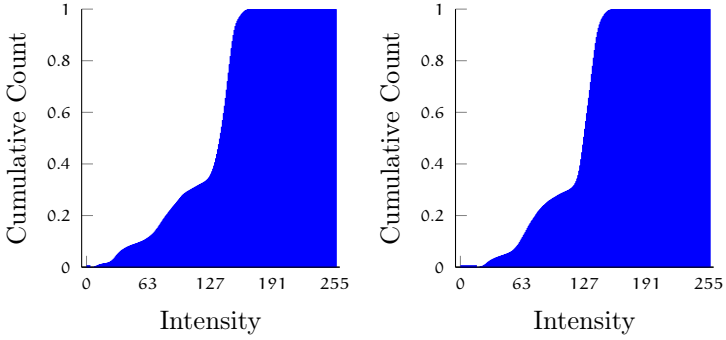
$$M_i = \sum_{q=1}^{i} m_q. \tag{3.3}$$

Figure 3.6 shows the cumulative histograms for the five image sub-classes from Figure 3.3. It should be noted that cumulative histograms do not contain more or less of a description of an image than their non-cumulative counterparts. Rather, the cumulative histogram is a mapping of the standard histogram. As with any

mapping to be used in pattern recognition or classification, it is possible that the cumulative histogram might provide different classification performance than standard histograms do. Both standard and cumulative histograms are evaluated for suitability as a feature vector for insect presence detection in this study.



(a) BIC-BM (b) BIC-OS (c) BIC-SF



(d) GIC-IP (e) GIC-NI

Figure 3.6: Cumulative histograms of the images in Figure 3.3. As in Figure 3.3, the top row, a-c, shows the Bad Image Capture classes, while d-e show the Good Image Capture classes.

Figure 3.7 shows the superimposed cumulative histograms for the GIC-IP and GIC-NI sub-classes. With the cumulative histogram, it is easy to see the initial left-shift of the GIC-IP intensity counts due to the insect present in the scene, as well as the left shift in the GIC-NI at higher intensity counts due to the larger portion of the backlight being unobstructed.

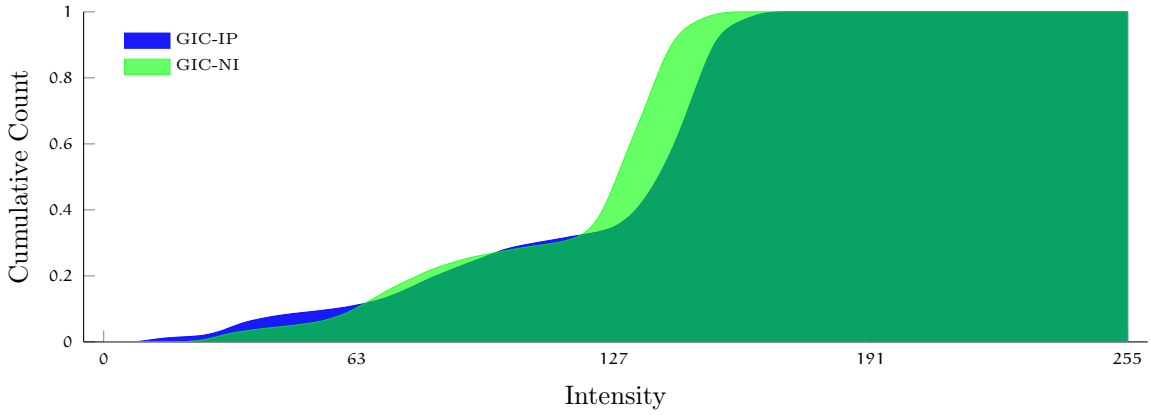


Figure 3.7: GIC class cumulative histograms for the images in Figure 3.3. GIC-IP is in blue (dark), while GIC-NI is green (lighter). The histograms for these two subclasses are much more similar than those of the BIC class.

3.2.2 *Insect Presence Classification*

Classification of images into classes was done with both traditional and cumulative histogram feature vectors using two classification methods: a simple Euclidean Distance measure from class prototype centers, and K-Nearest Neighbor clustering. For each classification method and feature type, the effects of varying the number of histogram bins and sub-sampling of the images were evaluated with respect to classification performance. Computation time was used as a proxy for energy at each level of bin size and sub-sampling interval in order to determine an optimal combination that is both accurate and efficient.

3.2.2.1 *Euclidean Distance Classifier*

Classifying based on Euclidean Distance from a mean feature vector prototype for each class is a rudimentary classification method. In essence, a group of histograms from known classes are averaged in the training procedure to produce a class prototype. This is the value that is used for testing the group classification of subsequent

images of unknown class type, where the predicted class membership is based on the minimum distance of the unknown image feature vector from each class prototype. Given that vector $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$ is a vector with n dimensions, the Euclidean Distance, or 2-norm, between the prototype and sample is

$$D(p,i) = \|\vec{x}_p - \vec{x}_i\| = \sqrt[2]{\sum_{j=1}^{n} (x_p(j) - x_i(j))^2}, \tag{3.4}$$

where $\vec{x}_p$ is the prototype feature vector for class p and $\vec{x}_i$ is the unknown, or to-be-classified feature vector for sample i. $x_p(j)$ and $x_i(j)$ are the j-th components of the prototype and unknown feature vectors, respectively.

The Euclidean Distance Classifier determines predicted class membership based on the minimum distance from the class prototype, or mean.

$$class(i) = \underset{p}{\operatorname{argmin}} [D(p,i)] \tag{3.5}$$

Using the Euclidean Distance as the classification metric is a simple approach, and is relatively efficient computationally. It is not sensitive to the number of training observations used to create the prototype means, and is thus not degraded in performance if the number of training observations for each class is radically different, assuming that all training vectors are good representations of the class. It does have the drawback that outliers in the training set contribute equally to the prototype when compared to good representations, introducing the opportunity for poor performance if an outlier is far from the prototype mean if it is not included.

The Euclidean Distance classifier is, in essence, a single parameter classifier that operates under the assumption that all members of the feature vectors are independent with respect to the rest of the vector. In the case of histograms, this is likely a poor assumption as the covariance between adjacent bins is usually large

in natural images. In the case of relative, or normalized, histograms, there is also a forced dependency introduced between non-adjacent bins: a large peak in a relative histogram in the "bright" area reduces the remaining bins' potential members since the sum of all bins must equal one. With these caveats aside, the Euclidean Distance function as a classifier is well understood and provides a benchmark for the potential classification performance of other, more advanced classifiers.

3.2.2.2 *K-Nearest Neighbor Clustering Classifier*

K-Nearest Neighbor (KNN) Clustering is an approach to pattern recognition and machine learning that seeks to develop class partitions based on a combination of the actual training samples rather than their mean prototype as in 3.2.2.1. In KNN classification, the training points are preserved as part of the classifier model, and the sum of the distances from the unknown vector to the k nearest training points in each class is used to determine the class membership. The KNN classifier uses the same definition for Euclidean Distance as in Equation (3.4). For each unknown feature vector, the total distance considered for classification of unknown vector $\mathbf{x}_i$ for class p and k neighbors is

$$D_{KNN}(\vec{\mathbf{x}}_i, p) = \sum_{j=1}^{k} D(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_{p,k}), \tag{3.6}$$

where $\vec{\mathbf{x}}_{p,k}$ are the k nearest training points to $\vec{\mathbf{x}}_p$. It should be noted that, although the distance in this research is taken to be the Euclidean distance, it is equally valid to use the 1-norm (Manhattan distance), p-norm (arbitrary power function in the exponent and root functions of the distance equation), or even inf-norm. The Euclidean distance is used here by choice.

Again, classification is based on the minimizing class for distance based on the rule

$$\mathrm{class}(i) = \underset{p}{\mathrm{argmin}} \left[D(\vec{x}_i, p) \right]. \tag{3.7}$$

KNN Clustering is also a well-known algorithm and has been used widely in many different pattern recognition and machine learning applications. When compared with using the Euclidean distance and training data means for classification as in 3.2.2.1, the KNN method is at least k-times as computationally intensive since the number of distance calculations must be increased as the number of neighbors considered grows. In practice, it is likely much more intensive if the training sets are large since the nearest neighbors from each class will be unknown for each new unknown feature vector to be classified. However, when compared to many other classifiers it is still considered a fast, low overhead method.

The KNN tools included in MATLAB set the default number of neighbors considered to $k = 1$. In a sense, KNN with $k = 1$ determines which of the training samples is the closest to the unknown feature vector to be classified. If there are outliers in the training data for one class that are close to clustered areas of training data for another class, it is possible to have misclassification in this region due to only one data point. As k increases, the influence that a single outlier has on the classification performance is reduced.

KNN Clustering is more sensitive to training data class size than the Euclidean Distance method as the number of neighbors increases since classes with smaller numbers of point in the training data may be overwhelmed in the classifier by the training data from more common classes. For instance, in the course of this study, the GIC-NI class images were more than a hundred fold more likely to occur in the field data collection than were any of the BIC classes. If a classifier were trained

with only 500 input vectors, this would imply that the number of BIC images would be in the double digits while GIC-NI would count over 400. In such a case, the rare classes are limiting in the number of neighbors, k, that can be considered.

For this study, KNN performance on classifying weevil trap images is investigated for neighborhoods ranging from $k = 1 \ldots 5$ due to the smaller number of images available from the BIC classes.

3.2.2.3 *Variation of Number of Histogram Bins*

Image histograms, both traditional and cumulative, can be computed for any number of bins from two to the maximum number of intensity values that exist in the image (256 possibilities for the 8-bit images in this study). Computational intensity and required RAM increase roughly proportionately as the number of histogram bins increases, both for histogram computation and for subsequent classification of images. This is due to the increased feature vectors produced with larger numbers of bins, both for storage and for distance calculations. Since it is desired to have autonomous, instrumented traps in the future that will likely be limited in computing resources, any reduction in computation, RAM, or data storage and transmission is desirable.

With reduced resources in mind, the effects of reducing the number of histogram bins used as the feature vector for image classification is tested to determine the effects of binning on computational requirements and classification performance. The number of bins is varied in the study from $j = 2 \ldots 256$.

3.2.2.4 *Variation of Sub-Sampling Interval*

Another potential way to reduce computational complexity and data storage/transmission requirements is to calculate histograms from smaller images. However, most cameras have a fixed image size, and the computational and storage requirements for down-scaling images are greater than that of computing histograms. Since the images are already in memory after acquisition, one alternative method to scaling the images prior to histogram computation is to sub-sample the images and compute the histograms based on a fraction of the original pixels. For example, computing a $16\,\mathtt{bin}$ histogram for a $640 \times 480,\ 8\,\mathtt{bit}$ original image requires $304,248\,\mathtt{bytes}$ of memory and takes $1.5\,\mathtt{ms}$. Resizing the image to 160×120 and computing the $16\,\mathtt{bin}$ histogram requires $323,448\,\mathtt{bytes}$ of memory and takes $4.0\,\mathtt{ms}$, while computing the $16\,\mathtt{bin}$ histogram subsampled every four pixels requires the same $304,248\,\mathtt{bytes}$, and takes $0.98\,\mathtt{ms}$. This requires the same memory footprint as the original image while providing a 59% reduction in computation time, while providing a 6.3% reduction in memory use and a 414% reduction in computation time when compared to the resized image.

In this study, the original input images are 640×480 pixel, RGB images. The original images are sub-sampled where sub-sampling is defined in MATLAB vector notation as

$$\begin{aligned} I_{input} &= I(1\colon 1\colon N, 1\colon 1\colon M) \\ I^{c}_{sampled} &= I(1\colon c\colon N, 1\colon c\colon M) \end{aligned} \tag{3.8}$$

where c is the integer sub-sampling rate. In words, subsampling is taking every c pixel in both the horizontal and vertical axes, while the size of c is constrained such that $c \leqslant \{\min M, N\}$. In practice, it makes little sense for c to even ap-

proach M or N. For this study, image subsampling is implemented at levels $c = \mathrm{round}(\sqrt{2^k})$, for $k = \langle 1,2,3,4,5,6,7,8,9,10 \rangle$, where $c = \langle 1,2,3,4,6,8,11,16,23,32 \rangle$. Subsampling effects are evaluated for histogram computation performance improvements and image classification performance.

3.2.2.5 *Histogram Adjustment for Contrast Enhancement*

As can be noted in the histograms shown in Figure 3.4 and Figure 3.5, typical images collected in this study do not fill the entire span of possible intensity values in their raw form. A common practice in image processing is to adjust images such that their intensity values do span the entire intensity gamut. The MATLAB function `imadjust()` provides this intensity adjustment to enhance contrast in images, often with the desired side effect that regions of interest are easier to segment from the rest of the image. By default, `imadjust()` stretches the images intensity gamut such that 1% of the image falls in the pure black intensity range, and 1% of the image falls in the pure white intensity range.

Since a portion of this study will need to extract insect silhouettes from the images, this contrast adjustment is likely to be beneficial at some point in an image's path from raw to indicating whether there is an insect and whether that insect is a pecan weevil. Additionally, the fact that ambient sunlight from outside of the trap will unavoidably cause fluctuations in the average intensity of collected images depending on the time of day, trap orientation, and the weather. This ambient light is unavoidable because the system relies on insects being able to enter the imaging chamber, thus necessitating at least one transparent opening to outside of the trap.

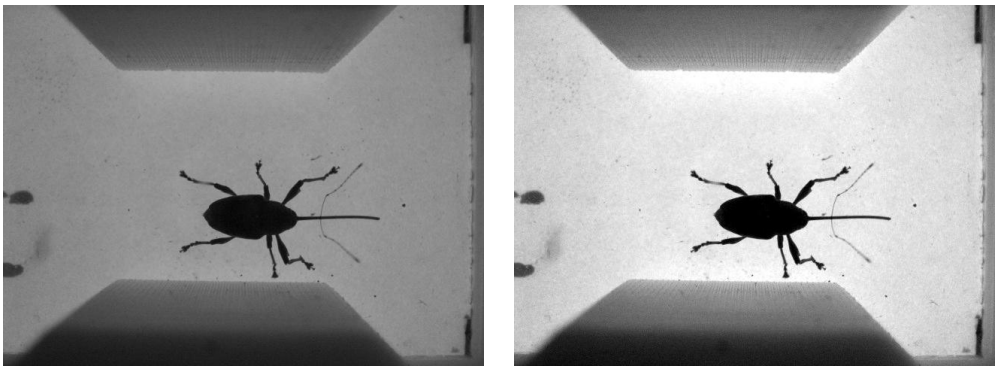
While `imadjust()` improves contrast in images, it also has some effects that could degrade image classification performance for insect detection. Figure 3.8 shows the effects of applying a histogram calculation to the contrast-enhanced image. Note that the stretching of the intensity gamut results in a sawtooth shape due to a lack of interpolation in the algorithm, and because each image has a different average intensity, the zero points in the new histogram do not always fall in the same bins from one image to the next. The effect of this pattern is also evaluated for improvement or degradation in classification performance.

It is possible that smoothing the contrast-enhanced histogram vector could provide both the benefit of improved contrast in the image for feature extraction and improve classification performance. The proposed smoothing function simply replaces any zero-valued bin in the adjusted histogram with the mean value of the adjacent bins before being normalized to sum to one. The pseudo-code for this smoothing function is

Algorithm 3.1 Histogram Smoothing Function in MATLAB

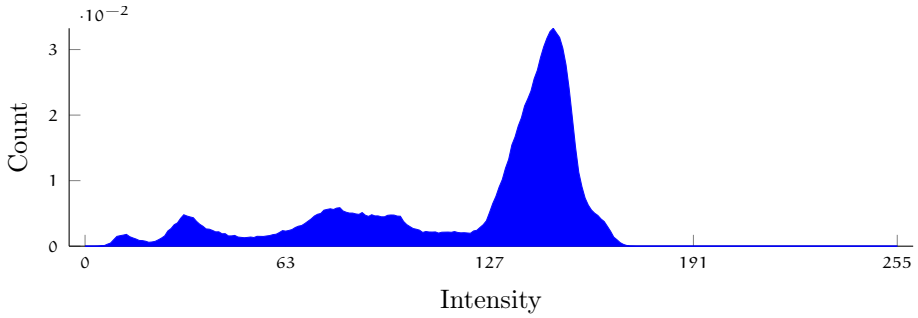
```
for index=2:(length(histogram)-1)
  if histogram(index)==1
    histogram(index) = (histogram(index-1) + histogram(index+1))/2
  end if
end for
```

The smoothed histogram, shown in Figure 3.9, has the advantage that it does not have zero-value entries interwoven through the curve. It also does not correspond to an actual image in a direct way, again due to lack of interpolation in the function `imadjust()`.

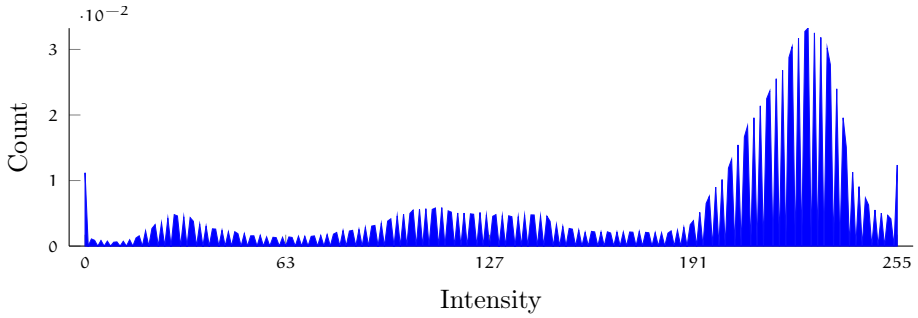


(a) Original image

(b) Adjusted image



(c) Original histogram



(d) `imadjust()` histogram

Figure 3.8: The effects of `imadjust()` contrast enhancement on images and their histograms. a) original pecan weevil image, b) histogram adjusted pecan weevil image, c) the histogram for the original image, and d) the histogram for the contrast adjusted image. The algorithm constrains the histogram such that 5% of the pixels have zero intensity (black) and 5% have maximum intensity (white), with the result a stretched histogram that may have a "comb" shape due to some intensities having zero pixel count.

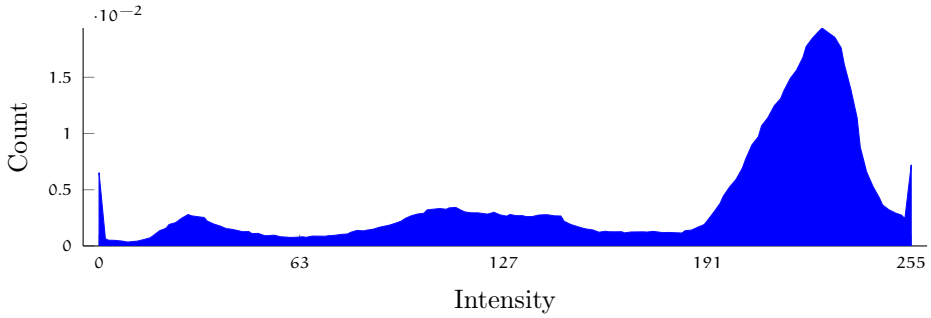


Figure 3.9: The effects of the smoothing kernel in Algorithm 3.1 on contrast enhanced histogram. The "comb" effect shown in Figure 3.8d is removed by the smoothing kernel, resulting in a smooth histogram.

The effect of this smoothing function on histograms is evaluated with respect to the classification performance when compared to raw image histograms and the sawtooth-shaped histograms that the smoothing is designed to remove.

3.2.2.6 *Image Pre-Processing for Insect Presence Detection*

Little pre-processing is required prior to using images for insect presence detection, as the presence protection detection step is intended to be a pre-processing step for insect identification classification in Section 3.3. As implemented, there is no pre-processing of images used in this step, but the effects of computing the contrast-enhanced image prior to this are evaluated. One potential processing step that might be added in future work is to crop input images slightly to remove the image edges. This would not be necessary for an instrumented trap in which the entire captured image was over backlighting rather than the present system in which the imaging chamber is clearly in view in most images.

3.3 PROPOSED INSECT IDENTIFICATION KERNEL

This section describes a proposed Insect Identification Kernel that was designed to classify insects into their prospective species based solely on the region-based shape of the specimens' silhouette. The goal of the proposed kernel is to provide accurate identification with low incidence of either false-positive or false-negative errors while attempting to reduce computational requirements to the minimal level to meet the identification needs. The remainder of this section describes the image pre-processing steps required for identification, the proposed polar-coordinate features used for training and classification of insects, the transformation method used to handle Cartesian coordinate images efficiently with polar coordinate-based features, and the proposed classifiers and their performance in the insect identification task. It concludes with a reduced feature set representation that combines the most effective of the proposed features and classifiers into the final proposed kernel.

3.3.1 *Image Pre-Processing for Insect Identification*

Pre-processing of images is required in order to attain a suitably sized binary image for moment computation. Figure 3.3 shows potential input images for this study. As can be seen, some are field acquired, some laboratory acquired. The field acquired images require more pre-processing than do the laboratory acquired, which must only be thresholded and windowed for binary conversion. In general, the pre-processing steps for field acquired images are found in Listing 3.1. In application with a network of instrumented traps, some of the pre-processing steps would be done on the node level. Figure 3.10 shows the steps of an image in pre-processing.

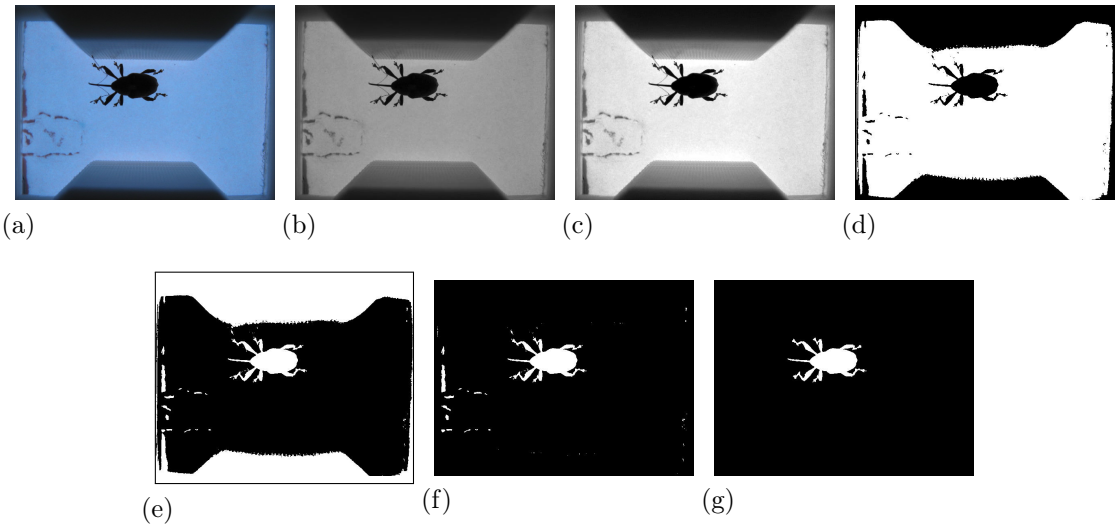


Figure 3.10: Pre-processing of field acquired images. a) Original, b) Grey-scale, c) Contrast-Enhanced, d) Thresholded, e) Compliment, f) Edge-connections removed, g) Small regions removed. It is possible that insects much smaller than a pecan weevil would be removed as small regions. A consequence of edge-connection removal is that partial insect shapes entering or exiting the imaging chamber would be ignored.

Listing 3.1: Pre-processing steps for insect shape recognition

```
1. Convert to grey-scale colorspace
2. Equalize the histogram
3. Threshold image for binary conversion
4. Take complement to make regions of interest equal one
5. Remove edge connected regions from binary image
6. Remove small regions from image
7. Remaining regions are potential insect
```

Field acquired, Run-Length Encoding (RLE) compressed images from a networked trap would be decompressed, and would enter in the pre-processing at step 5 in Listing 3.1. For this study, all images enter the pre-processing stage offline in JPEG form that can be directly loaded in MATLAB via the Image Processing Toolbox. After regions are cropped to the labeled connected regions, the images are zero-padded at the edges until the insect region centroid is in the center of a square image. If the square is smaller than 256X256, the cropped image is scaled

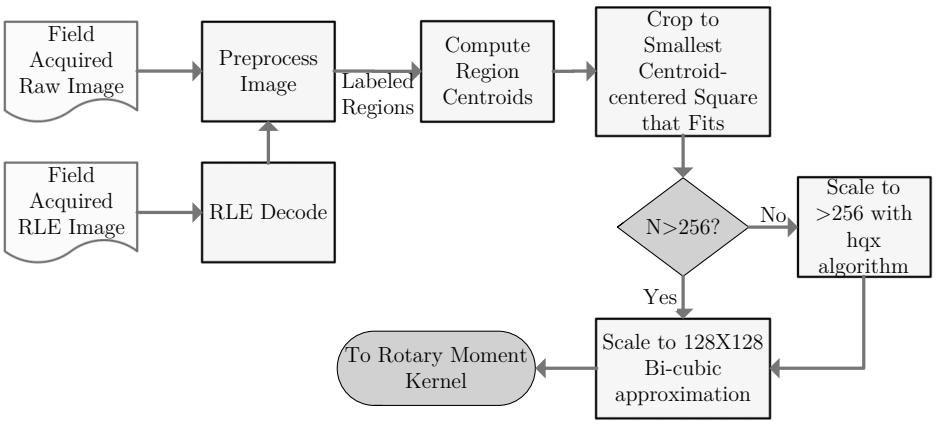


Figure 3.11: Different source images require a bit different handling. Cropping to a centroid-centered square image could require either adding or removing rows and columns at the edges of the image.

to at least 256X256 using the HQX algorithm to prevent pixel losses attributed to interpolation in bi-cubic enlargement of binary images. Then, traditional bi-cubic interpolation scaling is used to scale the image to 128X128. At this point, the image is ready for processing with the rotational moment kernel, starting with the square to circle transformation described in detail in Section 3.3.3. The final region processing flow chart is depicted in Figure 3.11. Figure 3.12 shows the results of the region resizing.

3.3.2 *Radial Moment Features for Shape Description*

The shape representation features used in this study were selected based on evidence in the literature that radial moment feature-based description of shapes provides for accurate classification of both optical characters (Chen et al., 1994; Hosny et al., 2011; Wang and Liao, 2013) and for insects (Ashaghathra, 2008). Three types of orthogonal radial moments are used in this study, including Zernike Moments (ZM), Pseudo-Zernike Moments (pZM), and Fourier-Mellin Moments (FMM). These

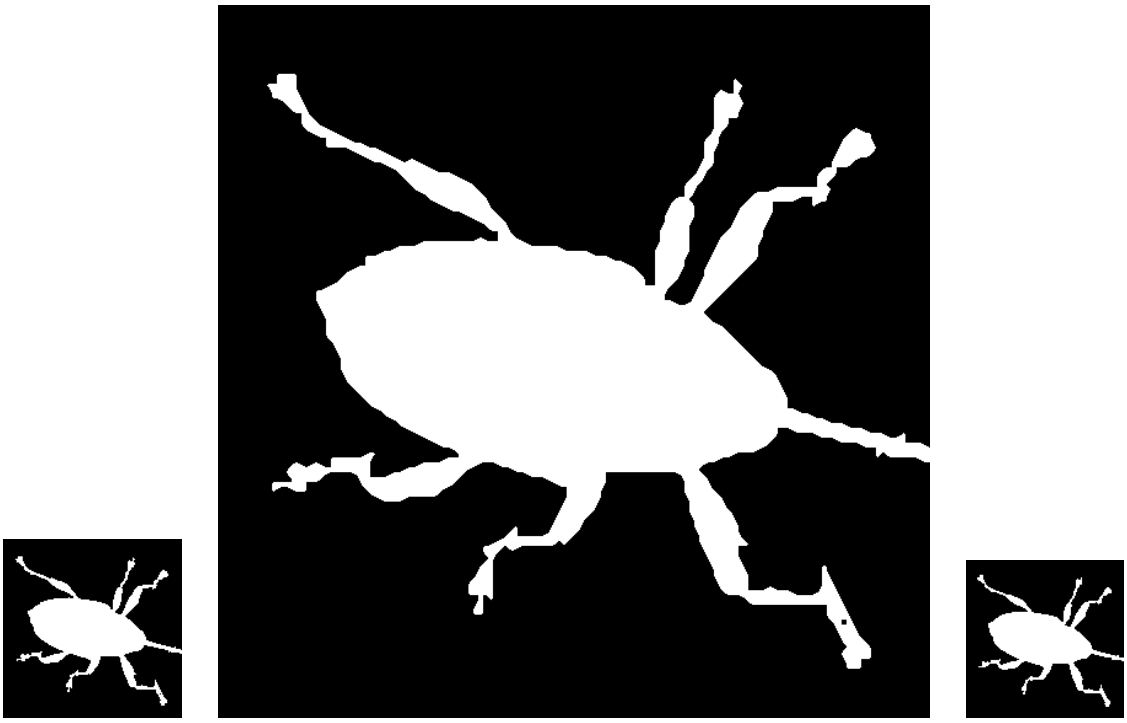


Figure 3.12: Insect region re-sizing for training and classification. a) Original 145×145, b) HQX algorithm upscale 580×580, c) Bi-cubic shrink 128×128. HQX upscaling prior to final scaling reduces the potential for lost pixels due to decimation at region edges inherent to the bi-cubic scaling algorithm when using binary images.

moments are not limited in the number of features produced for a given image, requiring a decision to made as to how many features to use for describing images. A fourth radial moment, the MPEG-7 Angular Radial Transform (ART), is defined in the MPEG-7 standard as a shape descriptor that is constrained in the number of features to provide a compact representation that may be limited in the shapes it can accurately represent. The four descriptors are compared for their efficacy in insect identification individually, and finally, combined as members of a mixed-feature vector for insect identification.

The ZM was chosen for this study due to its well established use, while pZM and FMM were chosen due to reported performance in the literature. Additionally,

as done in Hosny (2011a), these three moments can be re-written in a form that allows the kernel to be pre-computed and used for all images, resulting in improved overall speed of testing. The key attributes that make ZM, pZM, and FMM suitable for this study are orthogonality, rotation invariance, scale invariance, and definition over the unit circle (as opposed to Legendre moments, which are defined $\forall(x, y) \in \pm\infty$). Orthogonal features are those that have no dependence on other features a feature vector such that the covariance matrix is very close to zero for all non-diagonal entries. Invariance properties imply that values do not change when the given invariant property changes. Rotation invariance implies that the value of a feature would be the same for a given shape and a rotated copy of that shape, while scale invariance implies that the value would not change between an original image and a scaled version of the original.

3.3.2.1 *Zernike Moments*

The Zernike Moment was originally conceived as a way to quantify aberration in optical systems, including telescopes and microscopes, by Dutch physicist Fritz Zernike in 1934. ZMs are defined on the unit circle where radius $0 \leqslant r \leqslant 1$ and $-\pi \leqslant \theta \leqslant \pi$. For an image function $f(r, \theta)$ on polar coordinates, the ZM of the order p and repetition q are defined as:

$$Z_p^q(f(r,\theta)) = \frac{p+1}{\pi} \times \int_0^{(2\pi)} \int_0^1 \overline{V_p^q(r,\theta)} \cdot f(r,\theta) \cdot r \, dr \, d\theta \tag{3.9}$$

where, $p = 0, 1, 2, 3, ...\infty$ and q is a positive integer such that $(p - q) = even$ and $q \leqslant p$. The bar above $V_p^q(r, \theta)$ denotes the complex conjugate of the Zernike basis function, given by:

$$V_p^q(r,\theta) = R_{Z_p^q}(r) \cdot e^{jq\theta}, \tag{3.10}$$

where

$$p - q = even \tag{3.11}$$

and

$$0 \leqslant |q| \leqslant p, \quad 0 \leqslant p \leqslant \infty. \tag{3.12}$$

$R_{Z_p^q}(r)$ is the real-valued Zernike polynomial given by:

$$R_{Z_p^q}(r) = \sum_{m=0}^{\frac{p-|q|}{2}} (-1)^m \cdot \frac{(p-m)!}{m!(\frac{p+|q|}{2}-m)!(\frac{p-|q|}{2}-m)!} \cdot r^{(p-2m)}, \tag{3.13}$$

which are normalized such that $R_{Z_p^q}(1) = 1$. This normalization ensures that

$$-1 \leqslant R_{Z_p^q}(r) \leqslant 1 \tag{3.14}$$

inside the unit circle. ZM are orthogonal in that they satisfy the condition

$$\int_0^1 R_p^q(r) \cdot R_{p'}^q(r) \cdot r \, dr = \frac{1}{2 \cdot (p+1)} \cdot \delta_{pp'} \cdot R_p^q(1), \tag{3.15}$$

where

$$\delta_{pp'} = \begin{cases} 1 & \forall p = p' \\ 0 & \forall p \neq p' \end{cases}. \tag{3.16}$$

The effect of the order, p, is to increase the frequency of the radial basis polynomial, while the effect of repetition, q, is to increase the frequency of the angular basis function.

When used as a shape descriptor, the ZM can either remain in complex form as presented in equation 3.9, or it can be converted to the real magnitude of the complex moment. Using the real magnitude of the moment induces rotation invariance where the original image $f(r,\theta)$ rotated by angle α to form the new image $g(r,\theta) = f(r,\theta-\alpha)$ computes to the the same moment value:

$$Z_p^q(f(r,\theta)) = Z_p^q(g(r,\theta)) = Z_p^q(f(r,\theta-\alpha)). \tag{3.17}$$

Additionally, the repetitions for $q < 0$ are no longer orthogonal when using the real magnitude since $\left|e^{j\cdot q\cdot\theta}\right| = \left|e^{-j\cdot q\cdot\theta}\right|$.

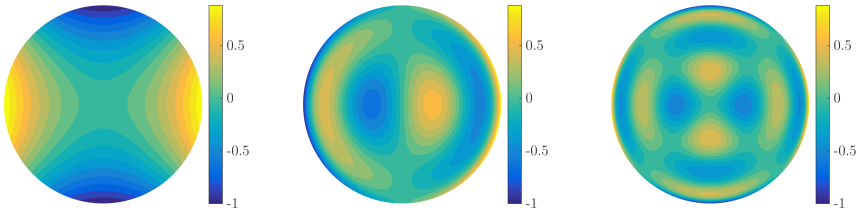
The first 12 ZM polynomials, excluding negative repetitions, are shown in Table 3.1, including all radial basis functions up to order $p = 5$. The real portion of the radial-angular basis functions can be seen in Figure 3.13[1], while a graphical comparison of the radial polynomials for all classes of radial moments can be seen in Figure 3.14.

Table 3.1: Equations for Zernike polynomials for $p = 0\ldots5$.

$R_0^0 = 1$	$R_1^1 = r$
$R_2^0 = 2r^2 - 1$	$R_2^2 = r^2$
$R_3^1 = 3r^3 - 2r$	$R_3^3 = r^3$
$R_4^0 = 6r^4 - 6r^2 + 1$	$R_4^2 = 4r^4 - 3r^2$
$R_4^4 = r^4$	$R_5^1 = 10r^5 - 12r^3 + 3r$
$R_5^3 = 5r^5 - 4r^3$	$R_5^5 = r^5$

Note: The polynomials are denoted as R_p^q, where p is the order and q is the repetition.

1 Additional ZM radial-angular basis functions can be seen in Appendix A.



(a) $V_{ZM_2^2}$ (b) $V_{ZM_5^1}$ (c) $V_{ZM_8^2}$

Figure 3.13: Real portion of selected Zernike Moments (ZM) radial-angular basis function. The imaginary portion of the same basis functions is opposite-signed copy of the real portion rotated by 90 degrees.

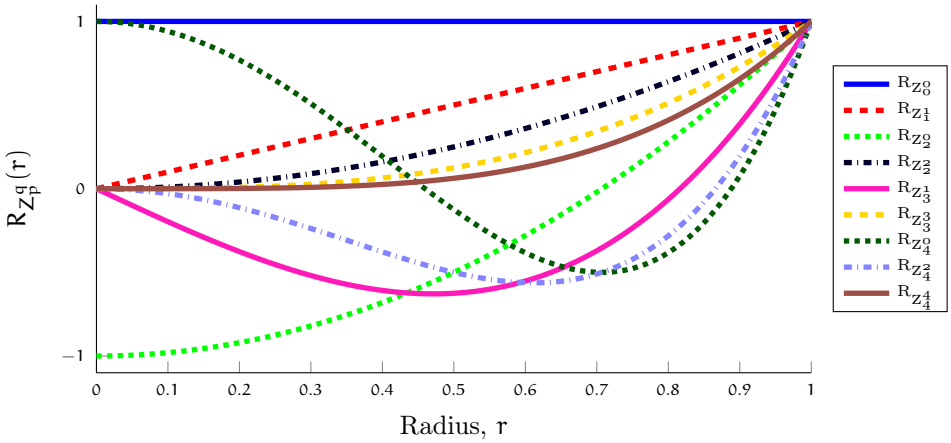


Figure 3.14: Zernike Moments (ZM) radial basis functions through $p = 4$. The radial basis polynomials for ZM are constrained to be between ± 1.

3.3.2.2 *Pseudo-Zernike Moments*

Pseudo-Zernike Moments (pZM) are another class of radial moments that were derived from the original acZM formulation, sharing many of the same properties and applications, including optics and image shape description. pZMs have been shown in practice to be more robust and less sensitive to noise than ZMs while often performing better in representing shape features.

pZMs are defined similarly to ZM, using both equations 3.9 and 3.10. With pZM, the polynomial changes from that in (3.13) to:

$$R_{pseudoZ_p^q}(r) = \sum_{k=0}^{p-|q|} (-1)^k \frac{(2p+1-k)!}{k!(p+|q|-k)!(p-|q|-k)!} r^{(p-k)} \tag{3.18}$$

such that

$$0 \leqslant |q| \leqslant p, \quad 0 \leqslant p < \infty. \tag{3.19}$$

Compared to the constraints for ZM orders and repetitions in (3.11) and (3.12), it can be seen that there are more existing pZM features for a given order $p \leqslant p_{max}$ due to the relaxation of the "even" constraint in (3.11). All pZMs are used, as all are orthogonal with respect to rotation of images as in (3.17) and there are no repetitions of $|-q|, |q|$. pZM radial basis polynomials are not constrained to the range $(0, 1)$ as ZM are in (3.14), leading to larger range in their basis functions and the potential to emphasize shape features more effectively.

The radial basis functions for the pZM are shown in Table 3.2 for orders $p = 1, 2, 3, 4, 5$. The real portion of the radial-angular basis functions can be seen in Figure 3.15[2], while a graphical comparison of the radial polynomials for all classes of radial moments can be seen in Figure 3.16.

3.3.2.3 *Fourier-Mellin Moments*

Fourier-Mellin Moments (FMM) are a third class of radial moments that were derived from the circular Fourier and radial Mellin transforms. As with pZM, FMM have also been shown provide better noise immunity and perform better in certain shape description and recognition applications. Unlike ZMs and pZMs, FMMs do not have an angular repetition dependence in their radial basis function, meaning that

2 Additional pZM radial-angular basis functions can be seen in Appendix B.

Table 3.2: Equations for Pseudo-Zernike polynomials for $p = 1 \ldots 5$.

$R_{pZ_1^0} = 3r - 2$

$R_{pZ_1^1} = r$

$R_{pZ_2^0} = 10r^2 - 12r + 3$

$R_{pZ_2^1} = 5r^2 - 4r$

$R_{pZ_2^2} = r^2$

$R_{pZ_3^0} = 35r^3 - 60r^2 + 30r - 4$

$R_{pZ_3^1} = 21r^3 - 30r^2 + 10r$

$R_{pZ_3^2} = 7r^3 - 6r^2$

$R_{pZ_3^3} = r^3$

$R_{pZ_4^0} = 126r^4 - 280r^3 + 210r^2 - 60r + 5$

$R_{pZ_4^1} = 84r^4 - 168r^3 + 105r^2 - 20r$

$R_{pZ_4^2} = 36r^4 - 56r^3 + 21r^2$

$R_{pZ_4^3} = 9r^4 - 8r^3$

$R_{pZ_4^4} = r^4$

$R_{pZ_5^0} = 462r^5 - 1260r^4 + 1260r^3 - 560r^2 + 105r - 6$

$R_{pZ_5^1} = 330r^5 - 840r^4 + 756r^3 - 280r^2 + 35r$

$R_{pZ_5^2} = 165r^5 - 360r^4 + 252r^3 - 56r^2$

$R_{pZ_5^3} = 55r^5 - 90r^4 + 36r^3$

$R_{pZ_5^4} = 11r^5 - 10r^4$

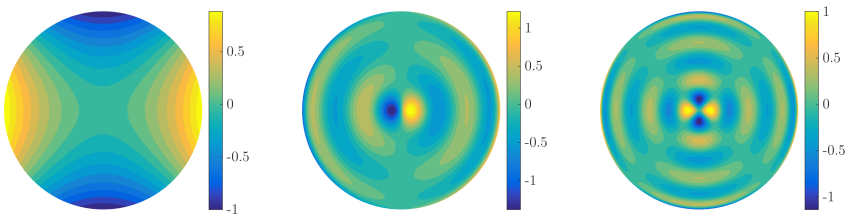
$R_{pZ_5^5} = r^5$

Note: Excluding $R_{pZ_0^0}$.
Note: The polynomials are denoted as $R_{pZ_p^q}$, where p is the order and q is the repetition.

the same radial basis function is used for all repetitions of a given order, providing potential computational savings when compared to the first two rotational moments, ZM and pZM. Basis functions for ZM and pZM depend on both order q and repetition p in the radial form, while the radial basis function for FMM depends only on the order q as see in equation 3.20.

FMM also use the definitions given in equations 3.9 and 3.10. The radial basis function for FMM is the Fourier-Mellin Polynomial:

$$R_{fm_p}(r) = \sum_{k=0}^{p} (-1)^{(p+k)} \frac{(p+k+1)!}{(p-k)!k!(k+1)!} r^k \tag{3.20}$$



(a) $V_{pZM_2^2}$ (b) $V_{pZM_5^1}$ (c) $V_{pZM_8^2}$

Figure 3.15: Real portion of selected Pseudo-Zernike Moments (pZM) radial-angular basis function. The imaginary portion of the same basis functions is opposite-signed copy of the real portion rotated by 90 degrees.

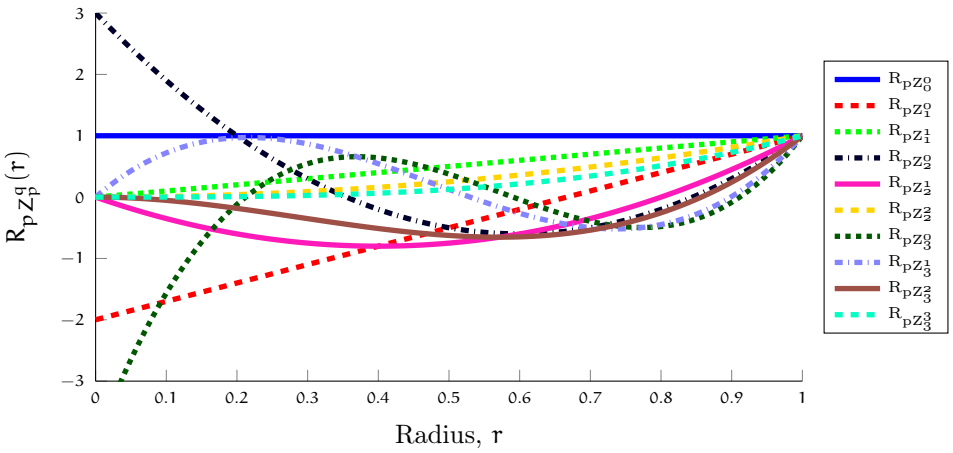


Figure 3.16: Pseudo-Zernike Moments (pZM) radial basis functions through $p = 3$. The radial basis polynomials for pZM are not constrained in magnitude as are those for ZM.

where

$$p \geqslant 0. \tag{3.21}$$

The angular repetitions, q, are not constrained for FMM to be less than the absolute value of order p. However, for this research the values of q are kept within the range $0 \leqslant q \leqslant p$, which is the same as the constraint used for pZM, and maintains the rotational invariance when the magnitude of the complex moment is used as a feature.

The radial basis functions for the FMM are shown in Table 3.3 for orders $p = 1 \ldots 10$. The real portion of the radial-angular basis functions can be seen in Figure 3.17[3], while a graphical comparison of the radial polynomials for all classes of radial moments can be seen in Figure 3.18.

Table 3.3: Equations for Fourier-Mellin polynomials for $p = 1 \ldots 5$.

$$R_{FM_1}(r) = 3r - 2$$

$$R_{FM_2}(r) = 10r^2 - 12r + 3$$

$$R_{FM_3}(r) = 35r^3 - 60r^2 + 30r - 4$$

$$R_{FM_4}(r) = 126r^4 - 280r^3 + 210r^2 - 60r + 5$$

$$R_{FM_5}(r) = 462r^5 - 1260r^4 + 1260r^3 - 560r^2 + 105r - 6$$

$$R_{FM_6}(r) = 1716r^6 - 5544r^5 + 6930r^4 - 4200r^3 + 1260r^2 - 168r + 7$$

$$R_{FM_7}(r) = 6435r^7 - 24024r^6 + 36036r^5 - 27720r^4 + 11550r^3 - 2520r^2 + 252r - 8$$

$$R_{FM_8}(r) = 24310r^8 - 102960r^7 + 180180r^6 - 168168r^5 + 90090r^4 - 27720r^3 + 4620r^2 - 360r + 9$$

$$R_{FM_9}(r) = 92378r^9 - 437580r^8 + 875160r^7 - 960960r^6 + 630630r^5 - 252252r^4 + 60060r^3 - 7920r^2 + 495r - 10$$

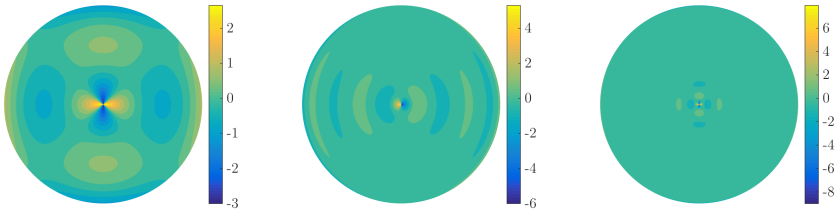
$$R_{FM_{10}}(r) = 352716r^{10} - 1847560r^9 + 4157010r^8 - 5250960r^7 + 4084080r^6 - 2018016r^5 + 630630r^4 - 120120r^3 + 12870r^2 - 660r + 11$$

Note: Excluding R_{FM_0}.
Note: The polynomials are denoted as R_{FM_p}, where p is the order.

3.3.2.4 *MPEG-7 ART Coefficients*

The final class of radial moments used in this research is the MPEG-7 Angular Radial Transform (ART) as defined in the MPEG-7 standard. While very similar to the ZM, pZM, and FMM, ART uses a radial basis function of cosine rather than a

3 Additional FMM radial-angular basis functions can be seen in Appendix C.



(a) $V_{FMM_2^2}$ (b) $V_{FMM_5^1}$ (c) $V_{FMM_8^2}$

Figure 3.17: Real portion of selected FMM radial-angular basis function. The imaginary portion of the same basis functions is opposite-signed copy of the real portion rotated by 90 degrees.

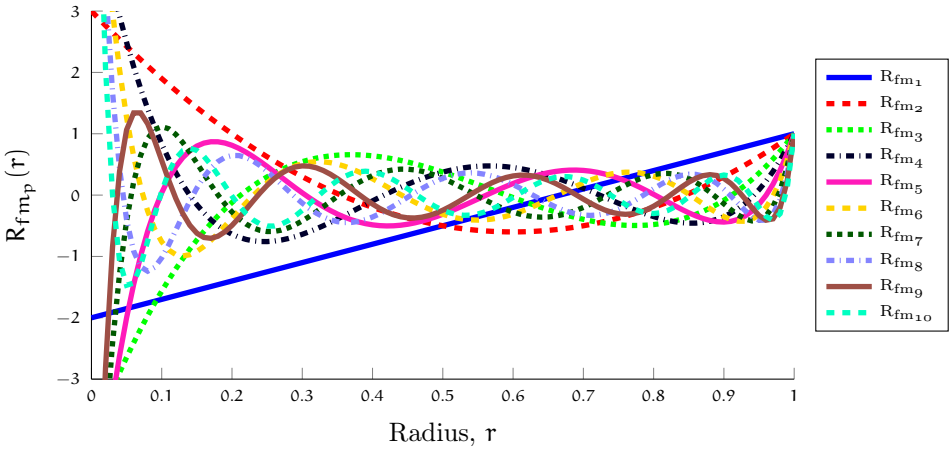


Figure 3.18: FMM radial basis functions through $p = 10$. The radial basis polynomials for FMM are not constrained in magnitude as are those for ZM.

polynomial. It is not orthagonal, but is by default limited in order and repetition such that aliasing is minimal. ART uses the same defining equation as in 3.9. The basis function for order p and repetition q is

$$\{V_{ART}\}_p^q(r,\theta) = \frac{1}{p+1} \cdot e^{j \cdot q \cdot \theta} \cdot R_p(r) \tag{3.22}$$

while the radial component is

$$R_p(r) = \begin{cases} 1, & p = 0 \\ \cos(\pi \cdot p \cdot r), & p \neq 0 \end{cases} \tag{3.23}$$

In the default implementation in the MPEG-7 standard, the orders and repetitions are limited to $p = 1 \ldots 10, \quad q = 1 \ldots 10$, and each shape descriptor is quantized to 4 bits/coefficient. In this research, the orders and repetitions are limited, but the shape descriptors remain in double precision floating point form. The radial basis functions for ART, $p = 1 \ldots 10$ are shown in Table 3.4. The real portion of the radial-angular basis functions can be seen in Figure 3.19[4], while a graphical comparison of the radial polynomials for all classes of radial moments can be seen in (3.23).

Table 3.4: Equations for ART radial basis functions for $p = 1 \ldots 10$.

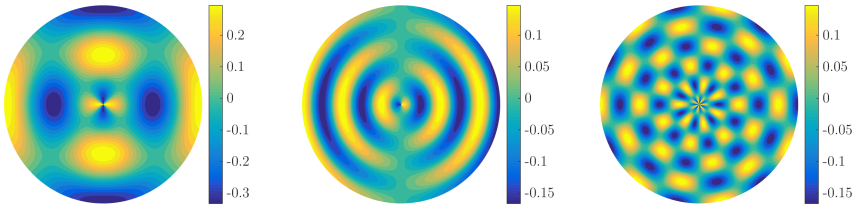
$Rart_1 = \frac{\cos(\pi r)}{\pi}$	$Rart_6 = \frac{\cos(6\pi r)}{\pi}$
$Rart_2 = \frac{\cos(2\pi r)}{\pi}$	$Rart_7 = \frac{\cos(7\pi r)}{\pi}$
$Rart_3 = \frac{\cos(3\pi r)}{\pi}$	$Rart_8 = \frac{\cos(8\pi r)}{\pi}$
$Rart_4 = \frac{\cos(4\pi r)}{\pi}$	$Rart_9 = \frac{\cos(9\pi r)}{\pi}$
$Rart_5 = \frac{\cos(5\pi r)}{\pi}$	$Rart_{10} = \frac{\cos(10\pi r)}{\pi}$

Note: Excluding R_{art_0}.
Note: The polynomials are denoted as R_{art_p}, where p is the order.

3.3.3 *Cartesian to Polar Image Transformation*

In order to compute the moments, an image is superimposed over the disks shown in Figure 3.21, multiplied, and summed over the whole unit circle. For most implementations, image files are rectangular and on a Cartesian coordinate system. This means that the rectangle must either be circumscribed inside the unit circle,

4 Additional ART radial-angular basis functions can be seen in Appendix D.



(a) $V_{ART_2^2}$ (b) $V_{ART_5^1}$ (c) $V_{ART_5^7}$

Figure 3.19: Real portion of selected ART radial-angular basis function. The imaginary portion of the same basis functions is opposite-signed copy of the real portion rotated by 90 degrees.

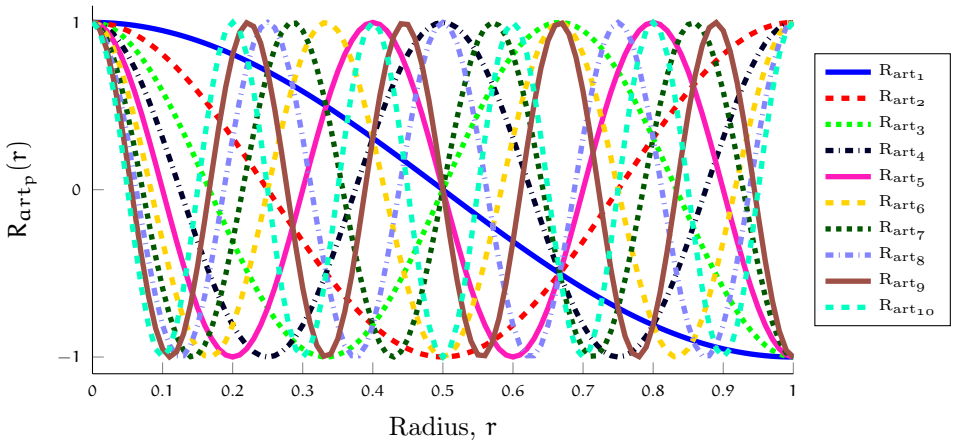


Figure 3.20: ART radial basis functions through $p = 10$. Since the radial basis functions for ART are dependent on a cosine function, they fall between ± 1.

ensuring that no information exists in much of the outer circle, or outside the circle, ensuring that data from the corners of the image is discarded. This is depicted in Figure 3.21.

For this study, a transformation used by Hosny (2012a) will be implemented to ensure all data remains in the moment calculation. This method maps the square pixel into polar coordinates without discarding pixels, as shown in Figure 3.22. When using orthogonal moments for image coding and reconstruction, this reduces some of the aliasing that occurs due to coordinate conversion. It is unknown whether

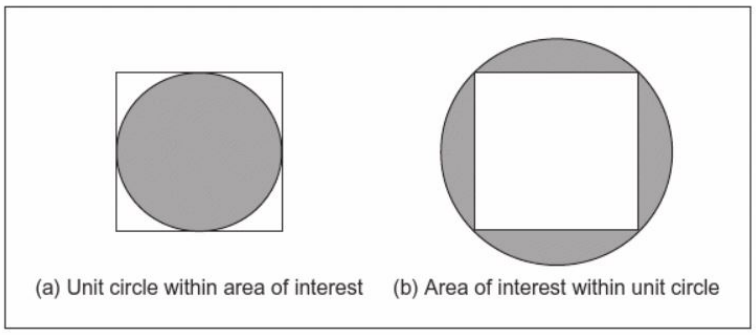


Figure 3.21: The Circle/Square incompatibility introduced due to rectangular digital images and polar-coordinate basis functions defined on the unit circle. In case (a), all pixels outside of the unit circle are dropped from the calculation, including 21.5% of the total shape near the corners that could be important in defining the shape for classification. In case (b), the unit circle is drawn to include all of the pixels in the square image. This results in 57% of the unit circle being composed of points that are not of interest in the calculation.

this transformation will have benefits in the case of feature coding for recognition purposes, but it allows for analytical evaluation of the moment functions on a per-pixel basis.

The transformation is done by dividing the square image into concentric rings, with the four pixels adjacent to the centroid set at the center of the rings. In radial coordinates, the radius calculations depend only on the pixel's ring membership, while the angular calculations depend on the ring and the individual pixel coordinates.

For an $N \times N$ image, the number of rings in the transformation is $N/2$. For each ring, the upper and lower bounds of the radius r are pre-initialized. Subsequently, the upper and lower bounds for the angular coordinate θ are pre-initialized for each pixel in the images. This initialization improves computational efficiency when many $N \times N$ images are to be used in the same problem. For the simple case of $N = 6$, the initialized variables include:

$$\mathtt{rings}(i,j) = \begin{Bmatrix} 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 0 & 1 & 2 \\ 2 & 1 & 0 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \end{Bmatrix},$$

$$r_l(\mathtt{ring}) = \begin{Bmatrix} 0 & \frac{1}{3} & \frac{2}{3} \end{Bmatrix},$$

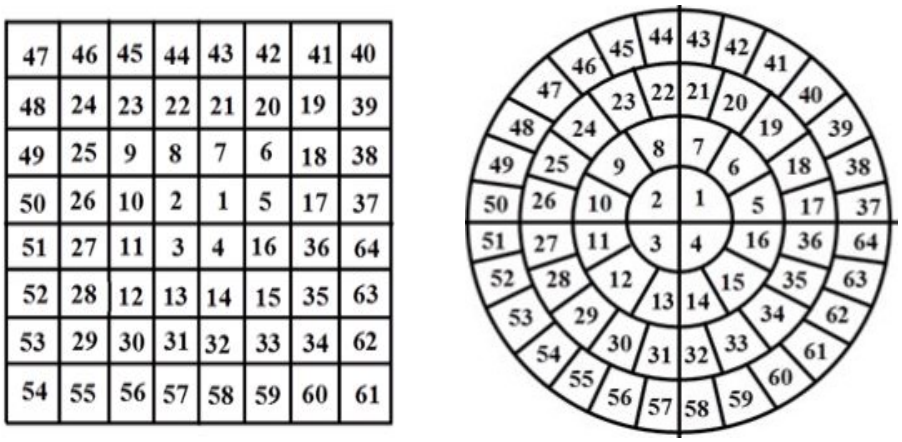
$$r_u(\mathtt{ring}) = \begin{Bmatrix} \frac{1}{3} & \frac{2}{3} & 1 \end{Bmatrix},$$

$$\theta_l(i,j) = \begin{Bmatrix} \frac{7\pi}{10} & \frac{6\pi}{10} & \frac{5\pi}{10} & \frac{4\pi}{10} & \frac{3\pi}{10} & \frac{2\pi}{10} \\ \frac{8\pi}{10} & \frac{4\pi}{6} & \frac{3\pi}{6} & \frac{2\pi}{6} & \frac{\pi}{6} & \frac{1\pi}{10} \\ \frac{9\pi}{10} & \frac{5\pi}{6} & \frac{\pi}{2} & 0 & 0 & 0 \\ \pi & \pi & \pi & \frac{3\pi}{2} & \frac{11\pi}{6} & \frac{19\pi}{10} \\ \frac{11\pi}{10} & \frac{7\pi}{6} & \frac{8\pi}{6} & \frac{9\pi}{6} & \frac{10\pi}{6} & \frac{18\pi}{10} \\ \frac{12\pi}{10} & \frac{13\pi}{10} & \frac{14\pi}{10} & \frac{15\pi}{10} & \frac{16\pi}{10} & \frac{17\pi}{10} \end{Bmatrix},$$

and

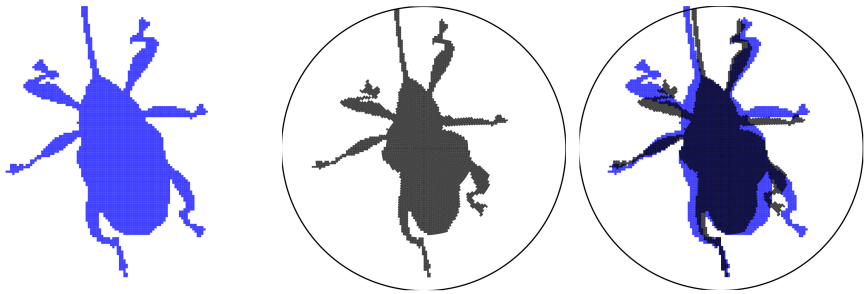
$$\theta_u(i,j) = \begin{Bmatrix} \frac{8\pi}{10} & \frac{7\pi}{10} & \frac{6\pi}{10} & \frac{5\pi}{10} & \frac{4\pi}{10} & \frac{3\pi}{10} \\ \frac{9\pi}{10} & \frac{5\pi}{6} & \frac{4\pi}{6} & \frac{3\pi}{6} & \frac{2\pi}{6} & \frac{2\pi}{10} \\ \pi & \pi & \pi & \frac{\pi}{2} & \frac{\pi}{6} & \frac{\pi}{10} \\ \frac{11\pi}{10} & \frac{7\pi}{6} & \frac{3\pi}{2} & 2\pi & 2\pi & 2\pi \\ \frac{12\pi}{10} & \frac{8\pi}{6} & \frac{9\pi}{6} & \frac{10\pi}{6} & \frac{11\pi}{6} & \frac{19\pi}{10} \\ \frac{13\pi}{10} & \frac{14\pi}{10} & \frac{15\pi}{10} & \frac{16\pi}{10} & \frac{17\pi}{10} & \frac{18\pi}{10} \end{Bmatrix}.$$

This corresponds to the pixels $1 \ldots 36$ in Figure 3.22. $\mathtt{rings}$, θ_l, and θ_u are size $N \times N$, while r_l and r_u are size $1 \times N/2$.



(a) Original Cartesian Image (b) Transformed Image

Figure 3.22: The square to circle pixel mapping used in this research. This mapping ensures that all of the pixels in the square image of interest are considered in the calculation of moment features, and that none of the unit circle is made up of area that will never contain part of the region of interest. The mapping also converts the square pixel boundaries into polar coordinates, allowing for exact calculation of integrals that are defined in the polar coordinate system.



(a) Cartesian weevil (b) Polar transformed weevil (c) Transformed weevil superimposed on original.

Figure 3.23: Example image shape and Cartesian to polar transformation results. The square to circle transformation deforms the original image slightly, with the most severe deformation occurring along the diagonals of the square. While this transformation causes deflection, images reconstructed from moment features computed using the transformation will not be deformed.

3.3.4 *Computation of Radial Moments*

In this study, all insect shapes to be recognized are in the form of logical images, with $f(r,\theta) = 1$ for pixels that are in the shape and $f(r,\theta) = 0$ for pixels outside of the shape, assuming the transformation in 3.3.3 are in effect. Note in 3.9 that the integrals are separable because the radial basis is independent of θ and the angular basis is independent of r. For each of the four radial moment descriptors, the basis functions can be written as:

$$V_p^q(r,\theta) = I_p^q(r) \cdot J_p^q(\theta) = R_p^q(r) \cdot e^{jq\theta}, \tag{3.24}$$

where $I_p^q(r)$ is the radial basis and $J_p^q(\theta)$ is the angular basis. After breaking the basis into its radial and angular components, the radial moment equation can be rewritten as

$$Z_p^q(f(r,\theta) = f(r,\theta) \bigcap \sum_n \sum_m \int_{r_l}^{r_u} R_p^q(r) \cdot dr \cdot \int_{\theta_l}^{\theta_u} e^{j \cdot q \cdot \theta} \cdot d\theta. \tag{3.25}$$

For any of the ZM, pZM, FMM, or ART moments, r_l and r_u are the lower and upper radius limits, respectively, for pixel (n,m). θ_l and θ_u are the lower and upper limits for angle for the same pixel. The $\cap$ operator indicates that only portions of the summation that are logical $\mathtt{true}$ in the image $f(r,\theta)$ are included in the moment. Variables m and n are the pixel row and column indices for the input image.

Equation (3.24) is the underlying enabler for the radial moment engine developed for this research. The summation can be performed for each class, order, and repetition in advance of testing, and can subsequently be used for use with each image under test simply by summing the pixel values that are $\mathtt{true}$ in the image.

3.3.5 *Insect Shape Classifiers*

With the insect shape descriptors set as ZM, pZM, FMM, or ART, the next task is to select a classification approach to determine the suitability of the chosen features to represent insect shapes. Generally, a group of inputs known as the training set is used to build a model that is subsequently used to classify further samples. In the case of this research, the approach taken is that of supervised learning since the true class of all insect shapes is known.

A good classifier is able to discern between different groups very accurately, with minimal computational intensity. The two classifiers to be examined for insect shape recognition are Support Vector Machine (SVM) and Naive Bayes (NB), which are further described in this section.

3.3.5.1 *Support Vector Machine Classifiers*

Support Vector Machine (SVM) classifiers seek to find the optimal hyper-plane in N-dimensional space that separates a given class from the rest of the data. This hyper-plane is optimal in that it gives the largest minimum distance to training data examples. This minimal distance is given the name *Margin* in SVM, and is the space that does not contain any training data. This makes the optimal hyper-plane the one that maximizes the margin of the training data. The training data points that are closest to the hyper-plane are given the name *Support Vectors*, which leads to the name of SVM. Often there is no hyper-plane that fully divides two classes. In this case the support vectors are used to determine class membership.

The MATLAB Statistics and Machine Learning Toolbox has two types of SVM routines: one for a single class membership classifier, and one for a discriminator be-

tween two classes. In implementing SVM for this research, the two-class formulation is used wherein the two classes are 'Pecan Weevil' and 'Other'.

SVM defines a hyper-plane that optimally separates two classes of training data. Feature vectors x_i made up of radial moments are of dimension d, while the two-class classifier constrains the class labels to be $y_i = \pm 1$, with positive values for the 'in' class and negative values for the 'out' class. The hyper-plane is defined as

$$\langle w, x \rangle + b = 0 \tag{3.26}$$

where $\langle w, x \rangle$ is the inner product of vectors w and x, and b is a real number. The classification problem is to find w and b to minimize $\|w\|$ so that for all training data (x_i, y_i) is

$$y_i \cdot (\langle w, x_i \rangle + b) \geqslant 1. \tag{3.27}$$

The support vectors are the data samples x_i that fall on the boundary of the hyperplane such that $y_i \cdot (\langle w, x_i \rangle + b = 1$. The optimal solution for separable data is the set w and b that allows for classification of the form

$$\mathrm{class}(z) = \mathrm{sign}(\langle w, z \rangle + b). \tag{3.28}$$

Given a dataset that has classes that are not separable, a soft margin is defined to separate many rather than all of the data points. This is done by adding *slack* variables, s_i and a penalty value, C. The training procedure is an optimization problem of

$$\min_{w,b,s} \left[\frac{1}{2} \langle w, w \rangle + C \sum_i s_i \right] \tag{3.29}$$

such that

$$y_i \cdot (\langle w, x_i \rangle + b) \geqslant 1 - s_i, \text{ and } \quad s_i \geqslant 0. \tag{3.30}$$

This is known as the L^1-norm problem, and is the form that is implemented in the standard MATLAB SVM functions. The classifier function remains the same as in (3.28).

3.3.5.2 *Naive Bayes Classifiers*

The Naive Bayes (NB) classifier seeks to build a parametric probability distribution for each class included in a training dataset, and then compare the predicted probabilities between classes for unknown feature vectors in order to assign a predicted class. Given the radial moments to be used as feature vectors in this research, NB will be used with multi-variate normal (Gaussian) distributions. When compared to SVM, NB has the advantage of being able to work on multi-class problems as opposed to the binary or two-class limitation in SVM.

Training in NB with a Normal Distribution requires that the training data for each class c_i be used to compute the within-class mean μ_i and standard deviation σ_i, where

$$\mu_i = \frac{1}{n} \sum_{j=1}^{n} x_{i,j}, \tag{3.31}$$

and

$$\sigma_i = \sqrt{\sum_{j=1}^{n} (x_{i,j} - \mu_i)^2}, \tag{3.32}$$

where i is the class index, j is the training vector index, and n is the number of dimensions of the sample vectors. The probability of an unseen vector x belonging to class c_i is given as

$$P(c_i|x) = \frac{1}{\sigma_i \cdot \sqrt{2 \cdot \pi}} e^{-\frac{(x-\mu_j)^2}{2\sigma^2}}. \tag{3.33}$$

The classification rule is set such that an unseen vector x is assigned to class i if and only if $P(c_i|x) > P(c_j|x) \qquad \text{for } 1 \leqslant i \leqslant m,\ j \neq i$. This is otherwise written as

$$\text{class}(x) = \underset{i}{\text{argmax}} \left[P(c_i|x)\right]. \tag{3.34}$$

NB makes the simplifying, if not always true assumption that the the features are conditionally independent of one another given the class membership. This assumption can be defined as

$$P(x, c_i) \approx \prod_{j=1}^{n} P(x_j|c_i). \tag{3.35}$$

This is the "naive" assumption that lends to the naming of the procedure, as the independence of the feature sets greatly simplifies the computation required.

3.3.6 *Dimensionality Reduction Methods*

In many different modeling or classification regimes, high-dimensional data has unintended consequences including increased computational intensity, greater storage requirements, and even loss in model accuracy. Some variables included in a data set can even be mainly noise. Dimensionality reduction methods are approaches to reduce the length of feature vectors in such a way that classification performance remains adequate, or even improves.

In this research, the chosen features could yield vectors with hundreds or thousands of dimensions. In the case that only one type of orthogonal radial moment is used, these thousands of features would continue to have mathematically independent features. However, it is also possible that much of the discernment ability of the said feature vectors lies within a small subset of the possible dimensions. This section describes two methods for reducing the dimensionality of the insect shape feature vectors, including Principle Component Analysis (PCA) and Fisher Multiple Discriminant Analysis (FMDA).

3.3.6.1 *Principle Component Analysis*

Principle Component Analysis (PCA) is a common procedure in regression analysis for determining which of the dimensions describes the majority of the variance in a given model. It is also used as a method to reduce the number of features of a model. In PCA, the feature vector is reordered from $1 \ldots N$, with the features of greatest variance listed with the lowest index. Higher index items can be pruned from the list with minimal impact on the ability of the model to describe the data. In application, PCA projects the features to a representation that best describes the data in a least-squares sense.

PCA depends on the a matrix representation of the data vectors known as the scatter matrix

$$S = \sum_{k=1}^{n} (x_k - \mu) \cdot (x_k - \mu)^\top \tag{3.36}$$

where k is the index of the sample vector, n is the number of samples, x is the feature vector, and μ is the mean of all sample data. The scatter matrix is analogous to the

covariance matrix for data that is normalized to have zero mean. We also define $\mathbf{e}$ as a unit vector such that $\|\mathbf{e}\| = 1$ and note that

$$\mathbf{x} = \mu + a\mathbf{e} \tag{3.37}$$

is a projection in the direction of $\mathbf{e}$ that passes through the mean data vector. The scatter matrix projected onto a unit vector yields the result

$$\mathbf{S}\mathbf{e} = \lambda\mathbf{e}, \tag{3.38}$$

where λ is an eigenvalue of the scatter matrix. This means that the optimal one dimensional projection of the data, in the least squares sense, is a projection through the mean along the eigenvector with the largest eigenvalue. Rearranging (3.38), the result is the standard eivenvalue problem with operator $J(\,)$ the Jacobian of the matrix

$$J(\mathbf{e}) = \mathbf{e}^\top \mathbf{S}\mathbf{e}. \tag{3.39}$$

This can be extended to d' dimensions so long as $d' < d$, where d is the original data dimensionality. The projection of the data into d' dimensions takes the form

$$\mathbf{x} = \mu + \sum_{i=1}^{d'} a_i \mathbf{e}_i, \tag{3.40}$$

where i is the eigenvector number, $\mathbf{e}_i$ is the i-th eigenvector, and a_i is an arbitrary scalar value. This projection can be visualized as being the principal axes of the hyper-ellipsoid of the original data.

The procedure for PCA dimensional reduction used in this research follows in Listing 3.2.

Listing 3.2: Principal Component Analysis Procedure

```
1. Compute Scatter Matrix
2. Compute Eigenvalues of Scatter Matrix
3. Sort Feature Vectors by Eigenvalue in Ascending Order 1...N
4. For the Eigenvector corresponding to largest Eigenvalue, select the
   feature with the largest absolute value weight
4. Add selected dimension to Best Feature Vectors list, retrain and test
   classification
5. If features remain in the unranked list, GOTO 4, ELSE
6. Record list of reduced features and performance
```

3.3.6.2 *Linear Discriminant Analysis*

Fisher Multiple Discriminant Analysis (FMDA) is a similar procedure to PCA in that it is a way to project data in such a way that makes dimensional reduction possible. Unlike PCA which seeks to select dimensions that describe the most variance in the data, discriminant analysis seeks to select dimensions based on those that describe most of the discriminating ability of the feature. The FMDA approach seeks to maximize the between-class scatter matrix given the within-class scatter matrix, while the PCA approach seeks to only maximize the overall scatter matrix. With PCA, it is possible that most of the variance in the data is due to noise, where selecting on maximum variance description may lead to selecting features that describe only noise. With FMDA, the projection or selection of features is done to maximize the distance between classes.

Given a set of n samples $\mathbf{x}_1 \ldots \mathbf{x}_n$ with n_1 in the class c_1 and n_2 in the class n_2, with class labels w_1 and w_2 for the two classes, we can form a linear combination of the components of $\mathbf{x}$ with the dot product equivalent to

$$z = \mathbf{w}^\top \mathbf{x}, \tag{3.41}$$

where there is a set of corresponding n samples divided into subsets for the classes c_1 and c_2. With $\|\mathbf{w}\| = 1$, each z is a projection of $\mathbf{x}$ in the $\mathbf{w}$ direction. For the two-class case, the scatter matrices S_i for class i are

$$S_i = \sum_{x \in D_i} (\mathbf{x} - \mu_i) \cdot (\mathbf{x} - \mu_i)^\top, \tag{3.42}$$

where D_i is the dataset belonging to class i. The within-class scatter matrix S_W is given as

$$S_W = S_1 + S_2. \tag{3.43}$$

The relation between the original space and the FMDA transformed space is

$$z_i^2 = \mathbf{w}^\top S_i \mathbf{w}, \tag{3.44}$$

while the between-class scatter matrix is given as

$$S_B = (\mu_1 - \mu_2) \cdot (\mu_1 - \mu_2)^\top. \tag{3.45}$$

The relation between the original space and the FMDA transformed space is

$$(\mu_1 - \mu_2)^2 = \mathbf{w}^\top S_B \mathbf{w}, \tag{3.46}$$

and the criterion for maximizing the ratio of S_B/S_W is

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top S_B \mathbf{w}}{\mathbf{w}^\top S_W \mathbf{w},}. \tag{3.47}$$

Equation (3.47) can be solved as a traditional eigenvalue problem as long as S_W is non-singular. This solution, as in PCA, is a projection through the mean in the direction of the eigenvector with the largest eigenvalue.

The procedure for FMDA dimensional reduction used in this research follows in Listing 3.3.

Listing 3.3: Fisher Multiple Discriminant Analysis Procedure

```
1. Compute Scatter Matrices S_B and S_W
2. Compute Eigenvalues of Scatter Ratio
3. Sort Feature Vectors by Eigenvalue in Ascending Order 1...N
4. Prune one dimension from Feature Vectors from RHS, retrain and test
   classification
5. If performance remains constant, GOTO 4, ELSE
6. Record list of reduced features and performance
```

3.4 EXPERIMENTAL DESIGN

Experiments in this research were intended to determine which features, classification methods, and image processing procedures would best enable detection and identification of insects in an instrumented insect trap. Both insect detection and insect identification problems are ultimately pattern recognition or classification problems. The ultimate test for variables examined in this research is whether they contribute to improved classification performance. Experiments and metrics for comparing performance when using different classifiers or variables are presented in this section, as well as the experiments used to demonstrate the performance of the proposed moment computation kernel.

3.4.1 *Insect Detection Experiments*

For analysis of the suitability of histograms as features for insect detection, a random sample of 2,036 from the 73,579 images were manually classified into the classes GIC-Insect Present (GIC-IP), GIC-No Insect (GIC-NI), BIC-Backlight Malfunction Type 1 (BIC-BM1), BIC-Backlight Malfunction Type 2 (BIC-BM2), BIC-Over Saturated (BIC-OS), and BIC-Shutter Failure (BIC-SF). This set of classified images was further divided into image sets for training and testing such that 50% were used to train and 50% were used to test classifier performance over 10 random permutations of the data. In the random permutations, the rule was enforced that nearly 50% of the specimens for each class were included in both the testing and training sets. The class membership for the dataset is shown in Table 3.5.

Table 3.5: Class membership for insect detection testing and training data.

Class	Members
GIC-NI	1611
GIC-IP	354
BIC-BM1	12
BIC-BM2	8
BIC-OS	37
BIC-SF	14

Note: This table represents the entire set of insect images used for both training classifiers and testing classification performance for insect detection. The classes are randomly divided between training and testing sets for a ten-fold cross-validation of classification performance.

The experiments for insect presence detection were a complete factorial treatment of histogram bins, subsampling rate, histogram type, classifier, and in case

of K-Nearest Neighbor (KNN) classification, the number of neighbors used by the classifier.

Listing 3.4: Members of the factorial treatment for insect presence detection.

```
1. Histogram bins,              i = 1...256
2. Subsampling rate,            j = 1...10
3. Histogram type,              k = 1...6
4. Classifier type,             p = 1...2
  4*. Number of neigbors,       q = 1...10
```

The metrics used to compare performance across treatments are the confusion matrix for the classifier, the correct classification rate, and the error rate. The confusion matrix for an N-class classifier is an $N \times N$ matrix in which the diagonals are the number of samples correctly classified, while the remaining members are the number of samples classified incorrectly for each between-class pairing. Table 3.6 shows an example confusion matrix for the performance of the 256^{th} bin, full size image, traditional unadjusted histogram feature vector with a single-neighbor KNN classifier, summed across the ten random training permutations.

Table 3.6: Example confusion matrix for insect detection classification.

		Predicted Membership					
		GIC-IP	GIC-NI	BIC-BM_1	BIC-BM_2	BIC-OS	BIC-SF
Actual Membership	GIC-IP	**1326**	442	0	0	2	0
	GIC-NI	175	**7880**	1	1	3	0
	BIC-BM_1	0	4	**186**	0	0	0
	BIC-BM_2	0	0	0	**70**	0	0
	BIC-OS	2	5	0	0	**53**	0
	BIC-SF	0	0	0	0	0	**40**

Note: Table entries along the bold diagonal are counts of correctly classified images. The sum of entries across rows would be the number of images belonging to each class, while the column sums would be the number of images classified in each class. All entries off of the bold diagonal are incorrect classifications.

3.4.2 *Insect Identification Experiments*

Insect images from instrumented weevil traps were combined with lab collected images from Ashaghathra's previous work to form a combined insect shape database of 1,829 samples with only 166 samples from non-pecan weevil. Table 3.7 shows a breakdown of the classes of insects[5] in the database.

Table 3.7: Insect shape images used for insect recognition.

Source	Super Class	Specimen	Count
Field	NPW	Ants	12
		Moths	5
		Beetles	70
		Spiders	5
	PW	Double PW	8
		Pecan Weevil	1,451
Lab	PW	Pecan Weevil	204
	NPW	Other	74

Note: This table represents the entire set of insect images used for both training classifiers and testing classification performance for insect identification. The classes are randomly divided between training and testing sets for a ten-fold cross-validation of classification performance.

The database images were randomly divided into 40% training images and 60% testing images, ensuring that both sets contain images from each class in Table 3.7. Feature vectors for each of the four classes were generated for each image, with the maximum order of each radial moment type listed in Table 3.8, providing a maximum feature vector for each insect shape of $n = 1,068$.

5 See Ashaghathra (2008) page 60 for a breakdown of non-weevil species.

Table 3.8: Polar Moment Features used as feature vectors.

Radial Moment	Maximum Order	Number of Features
Zernike Moments	30	256
Pseudo Zernike Moments	25	351
Fourier-Mellin Moments	25	351
Angular Rotary Transform	10	110

Both the Support Vector Machine (SVM) and Naive Bayes (NB) classifiers were trained with feature vectors from individual radial moment classes for maximum order $p_{max} = 2 \ldots Max$ where Max is the number from column two in Table 3.8, and tested for classification accuracy with the testing set. This allows for testing the efficacy of the radial moment classes themselves for coding discernible insect shapes. Each individual radial moment class is then subjected to both Principle Component Analysis (PCA) and Fisher Multiple Discriminant Analysis (FMDA) dimensionality reduction techniques to determine the potential performance of a reduced-space classifier using individual radial moment classes. The result is a factorial treatment across radial moment classes, radial moment feature vector length, classifier type, and dimensionality reduction technique.

In the combined feature vector formed from multiple polar moment classes, care is taken to ensure that singularities are not formed in the feature space due to repeated values. This is due to the fact that some of the radial and angular basis functions are shared between the different radial moment classes. The total set of equations for radial basis functions is combined, with repeat sets removed from the combined vector. The combined sets are then reduced in dimension using both PCA and FMDA, in the manner described in 3.3.6.

3.4.3 *Classification Performance Metrics*

The effects of these treatments are compared using the same type of confusion matrix as shown in Table 3.6 with either 2×2 size for the a binary classifier or $C \times C$ for a multi-class classifier, where C is the number of classes in the dataset. In the case of the binary classification, we can refer to the the terms Accuracy Rate (Acc), Error Rate (ER), Type I Error ($Error_I$), and Type II Error ($Error_{II}$) where

$$Acc = \frac{\sum_{i=j} Diag(confusion)}{\sum_i \sum_j confusion}, \tag{3.48}$$

$$ER = \frac{1}{Acc}, \tag{3.49}$$

$$Error_I = \frac{confusion(1,2)}{\sum_i \sum_j confusion}, \tag{3.50}$$

and

$$Error_{II} = \frac{confusion(2,1)}{\sum_i \sum_j confusion}. \tag{3.51}$$

That is, the Accuracy Rate is the ratio of the correctly classified images to the total number predicted. The Type I Error is the ratio of the total images that are not pecan weevil but are predicted to be by the classifier, and the Type II Error is the ratio of the total images that are pecan weevil that are predicted not to be. In the multi-class classification, the Accuracy Rate equation remains the same, but the sense of false positive or false negative of Type I and Type II errors are not as straightforward. In this case, one of the classes is considered to be the most important and the $C \times C$ confusion matrix is condensed to a 2×2 case, allowing the $Error_I$ and $Error_{II}$ to be used. However, in the case of multi-class classification, the general relationship $Acc = 100\% - Error_1 - Error_{II}$ no longer holds as the equations for accuracy and error remain the same, but the mis-

classifications between the classes deemed less important are not included in the error calculations.

The binary classifier has additional defined terms given its 2×2 confusion matrix, as shown in Table 3.9. In the table, the "positive" entries are those that are members of the class deemed more important, or are wrongly predicted to be by the classifier. "Negative" entries are those that either belong to the other class, or are incorrectly predicted to belong by the classifier. The "true" entries are those that are correctly classified by the classifier, while the "false" entries are incorrectly classified. True Positive (TP), is the number of testing samples correctly classified in the true group, while False Positive (FP), is the number of testing samples incorrectly predicted to belong to the true group. Conversely, True Negative (TN), is the count of testing samples belonging to the negative group that are correctly classified, while False Negative (FN), is the number of samples belonging to the true class incorrectly predicted to belong to the false class by the classifier. The binary confusion matrix terms give rise to the rate terms TPR, FNR, FPR, TNR, PPV, and NPV which correspond to the rates at which the classifier correctly predicts positives, incorrectly predicts positives, incorrectly predicts negatives, and correctly predicts negatives, the reliability of the classifier given a positive class prediction, and the reliability of the classifier given a negative class prediction, respectively. Three of the rate terms, TPR,TNR, and PPV, are also commonly referred to by the terms Sensitivity, Recall, and Precision, respectively. The equations for each of these rates in also included in Table 3.9.

Measures of classifier performance other than accuracy and error rates used in this study include the F_1 score, G score, and the Matthews Correlation Coefficient (MCC). The F_1 score is the harmonic mean of precision and sensitivity, given as

$$F_1 = 2 \cdot \frac{TP}{TP + FN + FP}. \tag{3.52}$$

The G score is the geometric mean of precision and sensitivity, given as

$$G = \sqrt{precision \cdot sensitivity} = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}}. \tag{3.53}$$

F_1 and G both range from zero to one, with zero indicating poor classifier performance and one indicating perfect performance. Both F_1 and G are undefined in the case that either there are no predicted positive samples, or there are no samples that truly belong in the positive class. The MCC is a balanced measure of classifier performance, depending on all four entries in the binary confusion matrix. It is given as

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TF + FP) \cdot (TN + FN)}}. \tag{3.54}$$

MCC is bound between -1 and 1, with a score $MCC = 1$ indicating perfect classification and $MCC = -1$ indicating that the classifier predicts perfectly wrong. $MCC = 0$ is the special case where the total number of correct prediction is exactly equal to the total number of incorrect predictions. Generally, MCC values greater than $MCC = 0.7$ are considered good classification results, with values approaching unity as performance improves toward the perfect classifier.

Table 3.9: Binary Classifier confusion matrix and definitions

		Predicted: Positive	Predicted: Negative		
True	Positive	TP	FN	$\mathsf{TPR} = \frac{\mathsf{TP}}{\mathsf{TP+FN}}$	$\mathsf{FNR} = \frac{\mathsf{FN}}{\mathsf{TP+FN}}$
	Negative	FP	TN	$\mathsf{FPR} = \frac{\mathsf{FP}}{\mathsf{FP+TN}}$	$\mathsf{TNR} = \frac{\mathsf{TN}}{\mathsf{FP+TN}}$
		$\mathsf{PPV} = \frac{\mathsf{TP}}{\mathsf{TP+FP}}$	$\mathsf{NPV} = \frac{\mathsf{TN}}{\mathsf{TN+FN}}$		

Note: The definitions for a binary classifier are shown. Abbreviations are TP for True Positives, FP for False Positives, FN for False Negatives, TN for True Negatives, TPR for True Positive Rate, FNR for False Negative Rate, FPR for False Positive Rate, and TNR for True Negative Rate. Note that the terminology denoting positive and negative is that of the medical field where that of a positive means that the subject would have a condition based on a gold standard, while negative subjects do not have the condition.

3.4.4 *Moment Kernel Performance Metrics*

The proposed insect identification kernel computes moment features based on an assumption that some of the variables used to calculate the moment features can be created globally in memory, resulting in a savings in computation time. Depending on the number of feature types selected for the identification task and the available system resources, this global list of variables could include the pre-computed values for each pixel in an image of given size, moment order, and repetition, or rather just the values of the polar coordinate limits for each pixel in the transformed cartesian to polar mapping.

The proposed rotary moment computation engine pre-allocates variables for each pixel's ring membership for the Cartesian to Polar transformation, the upper and lower limit of the radius for each ring, and the upper and lower limits for the rotation

angle for each pixel. This pre-allocation is optional, and was intended to improve moment computation efficiency for processing many insect shapes simultaneously. Figure 4.43 shows the computation time required to compute one moment feature for one image using the moment computation engine in four different scenarios:

1. Pre-allocated variables with one image passed to the engine at a time in a for-loop
2. Pre-allocated variables with a stack of images in a single 3-D matrix passed to the engine
3. No pre-allocation of variables with a stack of images in a single 3-D matrix passed to the engine
4. Pre-allocated variables, stacked 3-D matrix of images, running in a parfor parallel loop

The reference for comparison is the Lans Zernike Moment code that was used in the work of Ashaghathra (2008) to calculate Zernike Moments (ZM) features. This method is used to calculate the moment feature in a for-loop as in the first scenario of the moment engine listed above.

The next chapter in this work presents the results of experiments in both insect detection methods and insect identification methods.

CHAPTER IV

RESULTS AND DISCUSSION

4.1 INTRODUCTION

This chapter examines the results of the methods introduced in Chapter 3 with an eye toward automated, camera-enabled, networked traps being deployed in pecan orchards. For growers to use the feature vectors, algorithms, and classifiers examined in this research to make management decisions, it is important that the performance of each is well classified and documented. In this research, good performance consists of minimal impact on battery life of instrumented traps and well defined classification performance for each recognition task.

For a system deployed in pecan orchards, the limiting factor for desired performance is assumed to be power consumption at the by-trap level. Pecan orchards are typically characterized by shade under the canopy and reduced wind velocities due to the canopy, thereby limiting the potential for either solar or wind–based energy harvesting. This lack of energy availability for collection necesitates solutions that are minimally power intensive. Results presented in this chapter use computation time on a modern PC as a predictor of computation time on an embedded microcontroller system, and as such, as a proxy for power consumption. Put differently,

small computation time in MATLAB is taken to imply that the power consumption for an embedded system would be proportionately small.

Classification performance in this research is considered to be the rate at which the classifier correctly predicts the group membership of an unseen specimen, and is dependent upon both the feature vector and the classifier used for prediction. The nature of the classification, and the cost associated with misclassification, must be considered when considering classification performance. For instance, a raw image without an insect that is misclassfied as a with-insect image is only costly in that unnecessary further processing would be applied to the image, draining the battery. In contrast, a non-pecan weevil shape incorrectly classified as a pecan weevil would change the overall count of pecan weevil in an orchard and could lead to premature application of insecticide before the adult weevils have reached the canopy. Although misclassified image captures could have negative impact on the system power budget, it would not lead to the potential poorly timed application of chemicals that would result from an insect shape classification system that is highly biased toward predicting the pecan weevil class.

4.2 INSECT DETECTION CLASSIFICATION PERFORMANCE

Performance of the insect detection classification methods in this research is dependent on the histogram features being used, as well as the variations of histogram bins and image subsampling, and finally, on the classifier used to make predictions. The effects of each of these factors are presented here beginning with the type of histogram features.

4.2.1 *Baseline insect detection classification metrics*

The baseline classification performance is taken to be the case of standard and cumulative histograms for raw images with unity subsampling and 256 bins using the Euclidean Distance classifier such that only two classes are considered: GIC-Insect Present (GIC-IP) and OTHER, with OTHER consisting of GIC-No Insect (GIC-NI) and the four Bad Image Capture (BIC) classes. This results in a 2×2 confusion matrix with correct classifications in the diagonal elements and misclassifications in the anti-diagonal elements. The classifier is trained on 1,017 images selected by ten separate random permutations of the data such that 177 images were GIC-IP and 841 were OTHER, which matches the distribution of classes across the whole data set. The classifier performance was tested on 1,019 unseen images with the same distribution as the training set. Table 4.1 shows confusion matrices for this baseline performance with standard histogram features, including the best, worst, and average performance across the ten train-test pairs. Table 4.2 shows the confusion matrices for baseline performance of cumulative histogram features.

The baseline classifier metrics are shown in Table 4.3 for standard histograms and in Table 4.4 for cumulative histograms, including Accuracy rate, F_1 score, G score, Matthews Correlation Coefficient (MCC), and $\mathtt{Error}_{\mathrm{I}}$ and $\mathtt{Error}_{\mathrm{II}}$ rates, where a classification is considered a false-positive when an image of class OTHER is predicted to be in class GIC-IP. It should also be noted that both the training and testing sets for each permutation of the dataset contains 82.6% OTHER class images to only 17.4% GIC-IP class images.

Statistics for two "dumb classifiers" are also included in Tables 4.3 and 4.4 for comparison. The first dumb classifier predicts membership based only on the class membership percentages in the training data, and can be referred to as a Prior Prob-

Table 4.1: Confusion matrices for standard histogram baseline. The confusion matrices are shown for the (a) worst classification performance, (b) best classification performance, and (c) average classification performance across the ten-fold cross-validation of training and testing images.

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**153**	24
	OTHER	404	**437**

(a) Worst Performance, Acc = 58%

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**143**	34
	OTHER	264	**577**

(b) Best Performance, Acc = 71%

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**144.7**	32.3
	OTHER	316.1	**524.9**

(c) Average Performance, Acc = 66%

Note: Bold entries along the diagonal are counts of correctly classified images, while off-diagonal entries are incorrect classifications. The sum of all entries in each confusion matrix is the total number of images in the testing set. Diagonal entries represent the True Positives (TP) and True Negatives (TN) in the first and second rows, respectively. Off-diagonal entries represent the False Negative (FN) and False Positive (FP) counts in the first and second rows, respectively. FP are considered Type-I error while FN are Type-II error.

Table 4.2: Confusion matrices for cumulative histogram baseline. The confusion matrices are shown for the (a) worst classification performance, (b) best classification performance, and (c) average classification performance across the ten-fold cross-validation of training and testing images.

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**126**	51
	OTHER	402	**439**

(a) Worst Performance, Acc = 55.5%

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**137**	40
	OTHER	387	**454**

(b) Best Performance, Acc = 58.1%

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**132.2**	44.8
	OTHER	401.8	**439.2**

(c) Average Performance, Acc = 56.1%

Note: Bold entries along the diagonal are counts of correctly classified images, while off-diagonal entries are incorrect classifications. The sum of all entries in each confusion matrix is the total number of images in the testing set. Diagonal entries represent the True Positives (TP) and True Negatives (TN) in the first and second rows, respectively. Off-diagonal entries represent the False Negative (FN) and False Positive (FP) counts in the first and second rows, respectively. FP are considered Type-I error while FN are Type-II error.

ability Classifier. The Prior Probability Classifier predicts OTHER no matter the histogram input based on the higher prevalence of OTHER-class training images, and would have an accuracy rate $\mathsf{Acc} = 82.6\%$, with error rates of $\mathsf{Error}_{\mathrm{I}} = 17.4\%$ and $\mathsf{Error}_{\mathrm{II}} = 0\%$ given that the training and testing sets maintain the same prior probabilities. The second dumb classifier simply predicts all images to be in the GIC-IP class without regard to the histogram features presented, and can be referred to as a Select All Classifier. The Select All Classifier would have an accuracy rate $\mathsf{Acc} = 17.4\%$ with error rates of $\mathsf{Error}_{\mathrm{I}} = 0\%$ and $\mathsf{Error}_{\mathrm{II}} = 82.6\%$, all based on the prevalence of true class memberships in the testing image set. These two dumb classifiers provide a sane bound for the expected performance of a classifier, as well as a basis for comparing the performance of other classification methods. When comparing modifications of the classification toolchain, changes in classification performance must be compared to the baseline as well as the dumb classifier.

The baseline Euclidean classifiers have poorer overall accuracy than the Prior Probability Classifier but have much lower rates of $\mathsf{Error}_{\mathrm{I}}$. Similarly, the baseline Euclidean classifiers have improved accuracy when compared to the Select All Classifier with much lower rates of $\mathsf{Error}_{\mathrm{II}}$, with both results owing to the unbalanced nature of the test image set. Maximum accuracy in the baseline is 70.7% using standard histogram features and 58.1% using cumulative histograms. Note that, for both baseline cases, the loss in accuracy is highly attributable to $\mathsf{Error}_{\mathrm{II}}$ due to false-positive classification. Both standard and cumulative histograms demonstrate performance that is stable across training and testing data permutations based on the small ratio of the standard deviation of the accuracy to the mean, indicating that the histogram features show good potential for use as classifiers. In all calculated measures, the standard histograms perform better than the corresponding cumulative histograms. Classification performance for both baseline feature sets

Table 4.3: Baseline performance metrics for standard histogram features.

	Standard Histogram Classifier				Prior Prob Classifier	Select All Classifier
	Min	Mean	Max	St. Dev		
Acc (%)	58.0	65.8	70.7	3.50	82.6	17.4
F_1	0.417	0.455	0.496	0.022	0.0	0.296
G	0.487	0.507	0.544	0.017	$\sqrt{0.0/0.0}$	0.417
MCC	0.292	0.337	0.395	0.029	0.0/0.0	0.0/0.0
Error_I (%)	2.36	3.17	4.03	4.81	17.4	0.0
Error_II (%)	25.9	31.1	39.7	3.75	0.0	82.6

Note: Euclidean Distance Classifier with 256 bin standard histogram features and unity subsampling across ten random permutations of training and testing images, including performance metrics for dumb "select all" and "prior probability" classifiers. The prior probability classifier always selects the class that is the most prevalent in the sample population, while the select all classifier rejects the null hypothesis for any unseen sample. Acc is classification accuracy, MCC is the Mathews Correlation Coefficient, Error_I and Error_II are the Type-I and Type-II errors, respectively.

Table 4.4: Baseline performance metrics for cumulative histogram features.

	Cumulative Histogram Classifier				Prior Prob Classifier	Select All Classifier
	Min	Mean	Max	St. Dev		
Acc (%)	54.5	56.1	58.1	1.1	82.6	17.4
F_1	0.357	0.372	0.391	0.010	0.0	0.296
G	0.412	0.430	0.450	0.013	$\sqrt{0.0/0.0}$	0.417
MCC	0.177	0.204	0.238	0.018	0.0/0.0	0.0/0.0
Error_I (%)	3.73	4.40	5.01	0.005	17.4	0.0
Error_II (%)	36.9	39.5	41.6	0.013	0.0	82.6

Note: Euclidean Distance Classifier with 256 bin cumulative histogram feature and unity subsampling across ten random permutations of training and testing images, including performance metrics for dumb dumb "select all" and "prior probability" classifiers. The prior probability classifier always selects the class that is the most prevalent in the sample population, while the select all classifier rejects the null hypothesis for any unseen sample. Acc is classification accuracy, MCC is the Mathews Correlation Coefficient, Error_I and Error_II are the Type-I and Type-II errors, respectively.

is likely marginal for applied use: while more than 94% of collected images that should be further processed would be submitted for further processing, nearly 40% of those would not actually need further processing due to lack of insect presence.

Time for computing standard and cumulative histograms were equal, averaging $3.5\,\text{ms}$ for each 640×480 image in the set. Time for making the classification was also nearly equal for both histogram types at $t_{standard} = 6.08\,\mu\text{s}$ and $t_{cumulative} = 6.01\,\mu\text{s}$, as the number of calculations required for both classifications is the same. Computation time may be either increased or decreased with modifications to the baseline. In the case that classification performance is improved, increases in computation time may be warranted. However, marginal improvements in classification performance that result in large increases in computation time should be avoided.

4.2.2 *K-Nearest Neighbor Classification Baseline Performance*

The classification performance should see its greatest improvement when a more robust classifier is used with the same features as those of the baseline. In this research, the baseline is the Euclidean Distance Classifier using standard and cumulative histograms as feature vectors, while the incremental classifier is the K-Nearest Neighbor (KNN) binary classifier, using classes GIC-IP and OTHER. As expected, the more robust KNN classifier provides significant performance benefits over the baseline at the cost of increased calculation time.

The baseline KNN classifier was trained on $1,017$ of the same ten random permutations of the $2,036$ images from the image set, and tested against the remaining $1,019$ images for each permutation. Confusion matrices representing the maximum, minimum, and mean accuracy for the 1-Nearest Neighbor classifier are shown in

Table 4.5: Confusion matrices for KNN standard histogram baseline. The confusion matrices are shown for the (a) worst classification performance, (b) best classification performance, and (c) average classification performance across the ten-fold cross-validation of training and testing images.

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**108**	69
	OTHER	21	**821**

(a) Worst Performance, Acc = 91.1%

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**121**	56
	OTHER	19	**823**

(b) Best Performance, Acc = 92.6%

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**116.7**	60.3
	OTHER	25.1	**816.9**

(c) Average Performance, Acc = 91.6%

Note: Bold entries along the diagonal are counts of correctly classified images, while off-diagonal entries are incorrect classifications. The sum of all entries in each confusion matrix is the total number of images in the testing set. Diagonal entries represent the True Positives (TP) and True Negatives (TN) in the first and second rows, respectively. Off-diagonal entries represent the False Negative (FN) and False Positive (FP) counts in the first and second rows, respectively. FP are considered Type-I error while FN are Type-II error.

The main difference between the KNN classifier and the Euclidean classifier is that the model for KNN utilizes all of the training data for each classification while the Euclidean classifier uses only the mean feature vector of the training set. This results in improved performance, but also requires more memory and processing time. Another important difference is that the KNN classifier can predict class membership based on the distance between k neighbors in the training set, such that $k \geqslant 1$. Larger k values require still greater processing time, as the number of calculations required for any class prediction must be done k-times.

Table 4.6: Confusion matrices for KNN cumulative histogram baseline. The confusion matrices are shown for the (a) worst classification performance, (b) best classification performance, and (c) average classification performance across the ten-fold cross-validation of training and testing images.

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**127**	50
	OTHER	23	**819**

(a) Worst Performance, Acc = 92.8%

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**141**	36
	OTHER	17	**825**

(b) Best Performance, Acc = 94.8%

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**132.6**	44.4
	OTHER	17.7	**824.3**

(c) Average Performance, Acc = 93.9%

Note: Bold entries along the diagonal are counts of correctly classified images, while off-diagonal entries are incorrect classifications. The sum of all entries in each confusion matrix is the total number of images in the testing set. Diagonal entries represent the True Positives (TP) and True Negatives (TN) in the first and second rows, respectively. Off-diagonal entries represent the False Negative (FN) and False Positive (FP) counts in the first and second rows, respectively. FP are considered Type-I error while FN are Type-II error.

Table 4.7: Baseline KNN performance for standard histogram features.

	Standard Histogram Classifier				Prior Prob Classifier	Select All Classifier
	Min	Mean	Max	St. Dev		
Acc (%)	90.2	91.6	92.6	0.70	82.6	17.4
F_1	0.680	0.732	0.763	0.026	0.0	0.296
G_1	0.686	0.737	0.769	0.025	$\sqrt{0.0/0.0}$	0.417
MCC	0.631	0.689	0.727	0.028	0.0/0.0	0.0/0.0
Error_I (%)	5.20	5.92	6.97	0.65	17.4	0.0
Error_II (%)	1.77	2.46	3.34	0.51	0.0	82.6

Note:KNN Classifier with 256 bin standard histogram feature and unity subsampling across ten random permutations of training and testing images with $k = 1$, including performance metrics for dumb "select all" and "prior probability" classifiers. The prior probability classifier always selects the class that is the most prevalent in the sample population, while the select all classifier rejects the null hypothesis for any unseen sample. Acc is classification accuracy, MCC is the Mathews Correlation Coefficient, Error_I and Error_II are the Type-I and Type-II errors, respectively.

Table 4.8: Baseline KNN performance for cumulative histogram features.

	Cumulative Histogram Classifier				Prior Prob Classifier	Select All Classifier
	Min	Mean	Max	St. Dev		
Acc (%)	92.8	93.9	94.8	0.72	82.6	17.4
F_1	0.777	0.810	0.842	0.023	0.0	0.296
G_1	0.779	0.813	0.843	0.023	$\sqrt{0.0/0.0}$	0.417
MCC	0.738	0.778	0.813	0.027	0.0/0.0	0.0/0.0
$Error_I$ (%)	3.53	4.36	5.00	0.51	17.4	0.0
$Error_{II}$ (%)	0.98	1.74	2.26	0.40	0.0	82.6

Note:KNN Classifier with 256 bin cumulative histogram feature and unity subsampling across ten random permutations of training and testing images with $k = 1$, including performance metrics for dumb "select all" and "prior probability" classifiers. The prior probability classifier always selects the class that is the most prevalent in the sample population, while the select all classifier rejects the null hypothesis for any unseen sample. Acc is classification accuracy, MCC is the Mathews Correlation Coefficient, $Error_I$ and $Error_{II}$ are the Type-I and Type-II errors, respectively.

The baseline 256 bin standard histogram classification performance is shown in Table 4.7 while the baseline performance for using cumulative histograms is in Table 4.8. Perhaps surprisingly, the average accuracy for the KNN classifier is $\mathsf{ACC} = 91.6\%$ using standard histogram features and $\mathsf{ACC} = 93.9\%$ with cumulative histogram features, while the Euclidean classifier was more accurate with standard histogram features. KNN achieves these accuracy improvements due to a great reduction in $\mathsf{Error}_{\mathrm{II}}$ rates with a smaller and varying change $\mathsf{Error}_{\mathrm{I}}$ rates. Mean $\mathsf{Error}_{\mathrm{II}}$ rates were reduced from 31.1% to 2.46% for standard histogram features and from 39.5% to 1.74% for cumulative histogram features. Mean $\mathsf{Error}_{\mathrm{I}}$ rates were increased from 3.17% to 5.92% for standard histogram features and decreased from 4.40% to 4.36% for cumulative histogram features. Computation time for classification was $\mathsf{t}_{\mathsf{standard}} = 56.8\,\mu\mathsf{s}$ and $\mathsf{t}_{\mathsf{cumulative}} = 57.8\,\mu\mathsf{s}$, a full order of magnitude greater than that of the Euclidean classifier. However, this increase in computation time is relatively small when compared to the time required to calculate the histogram features themselves at $1.1\,\mathsf{ms}$.

4.2.3 *KNN Performance with varied* k

Prior to performing experiments, it was assumed that classifier performance using KNN classifiers would be improved as the number of neighbors, k, being considered was increased at a cost of increased computational time. While larger neighborhoods require greater computation time as expected, the classification performance does not improve as was expected.

Figure 4.1 shows the effect of varying k from one to ten on the classification $\mathsf{Error}_{\mathrm{I}}$ and $\mathsf{Error}_{\mathrm{II}}$ rates, as well as the effect of varying k on the average computation time required for a single classification with 256 bin standard histograms

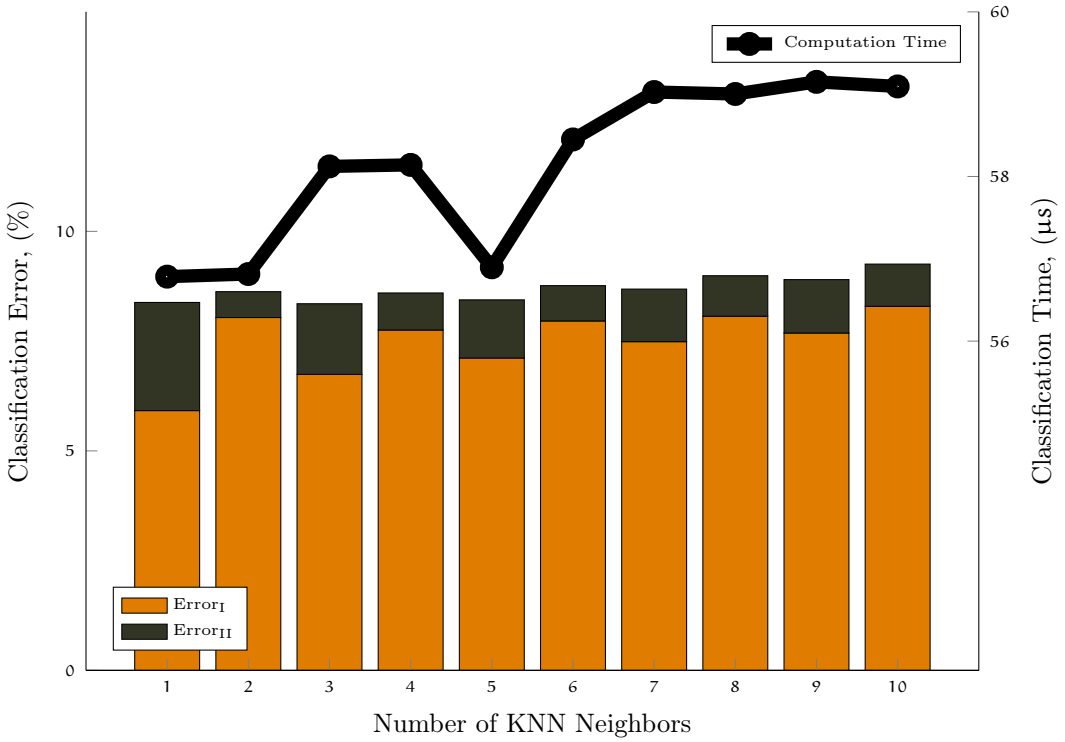


Figure 4.1: The effects of varying the number of neighbors, k, on KNN classifier performance with standard histogram features. The height of the stacked bar for each value of k is the total error rate, where the accuracy is equal to $\mathrm{Acc} = 1 - \mathrm{Error}_{\mathrm{I}} - \mathrm{Error}_{\mathrm{II}}$.

averaged over the ten training and testing permutations. The overall error rate increases as the number of neighbors increases, although the $\mathrm{Error}_{\mathrm{II}}$ rate is smaller for $k > 1$. In the case of insect detection in an image from a trap, reductions in $\mathrm{Error}_{\mathrm{II}}$ rate is desirable as this would be the rate of images that contain insects that are classified as not containing insects. Computation time does increase as k increases, with a $k = 1$ neighbor classification requiring $56.7\,\mu\mathrm{s}$ while the computation cost of a $k = 10$ neighbor classification is only $59.1\,\mu\mathrm{s}$, representing an increase of only 4.1% over the base while providing worse performance.

Figure 4.2 shows the effect of varying k from one to ten on the classification $\mathrm{Error}_{\mathrm{I}}$ and $\mathrm{Error}_{\mathrm{II}}$ rates, as well as the effect of varying k on the average compu-

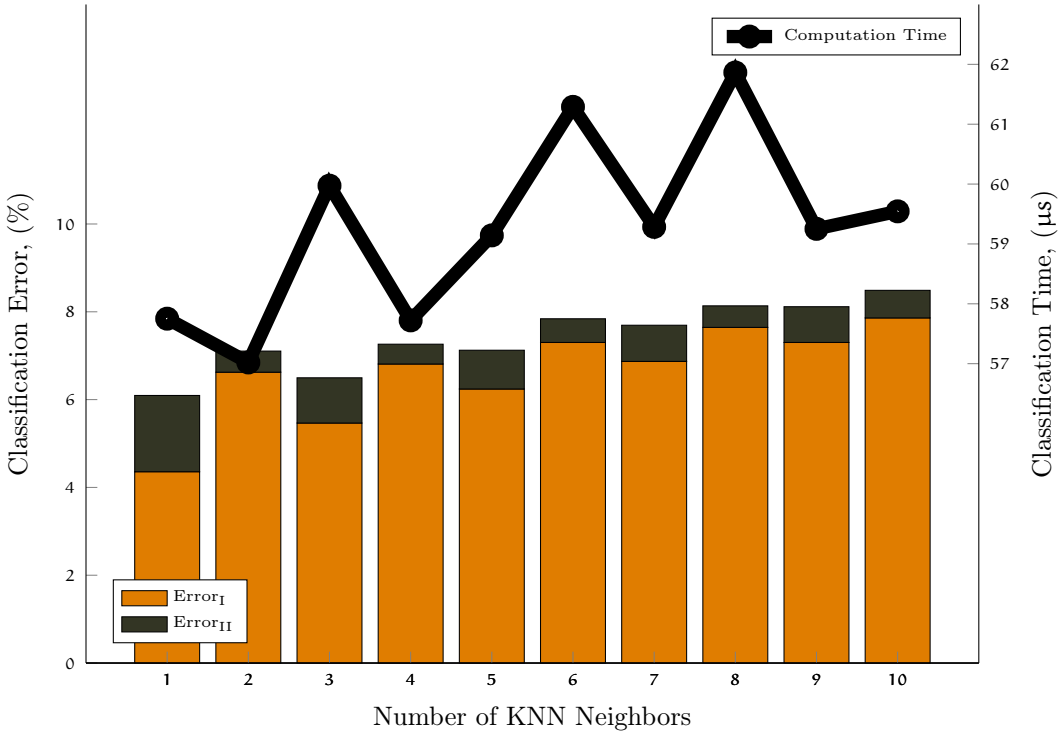


Figure 4.2: The effects of varying the number of neighbors, k, on KNN classifier performance with cumulative histogram features. The height of the stacked bar for each value of k is the total error rate, where the accuracy is equal to $\mathrm{Acc} = 1 - \mathrm{Error}_{\mathrm{I}} - \mathrm{Error}_{\mathrm{II}}$.

tation time required for a single classification with 256 bin cumulative histograms averaged over the ten training and testing permutations. The overall error rate increases as the number of neighbors increases, although the $\mathrm{Error}_{\mathrm{II}}$ rate is smaller for $k > 1$. In the case of insect detection in an image from a trap, reductions in $\mathrm{Error}_{\mathrm{II}}$ rate is desirable as this would be the rate of images that contain insects that are classified as not containing insects. Computation time does increase as k increases, with a $k = 1$ neighbor classification requiring $57.7\,\mu s$ while the computation cost of a $k = 10$ neighbor classification is only $59.5\,\mu s$, representing an increase of only 3.1% over the base while providing worse performance.

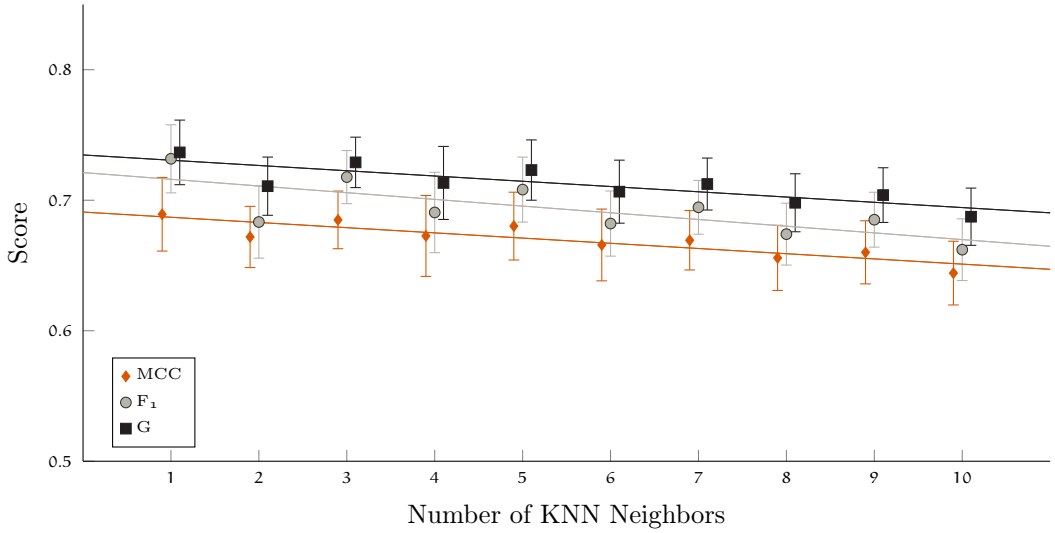


Figure 4.3: Performance measures for KNN classification for varying k with standard histogram features. The three classifier performance measures, MCC, F_1, and G are shown averaged across ten training and testing data permutations. Error bar lengths are ± the standard deviation value for the ten training permutations. Lines are the linear least-squares fit. Two of the three data series are offset in the x-axis to make markers more easily visible. Note that all three are trending toward worse performance as the neighborhood size increases.

The MCC, F_1, and G scores for varying neighborhood size k are shown in Figure 4.3 for standard histogram features and in Figure 4.4 for cumulative histogram features. The same trends hold for both types of histogram features, with overall performance measures diminishing as the classifier neighborhood size k increases. With KNN classifiers, the cumulative histogram features provide better classification performance than standard histogram features, which represents a role reversal from that of the Euclidean classsifier. The differences in computation time between the two feature sets are negligible.

4.2.4 *Variation of histogram bins*

In this section, the effects of reducing the size of the histogram feature vectors are explored for both standard and cumulative histogram features, and for both

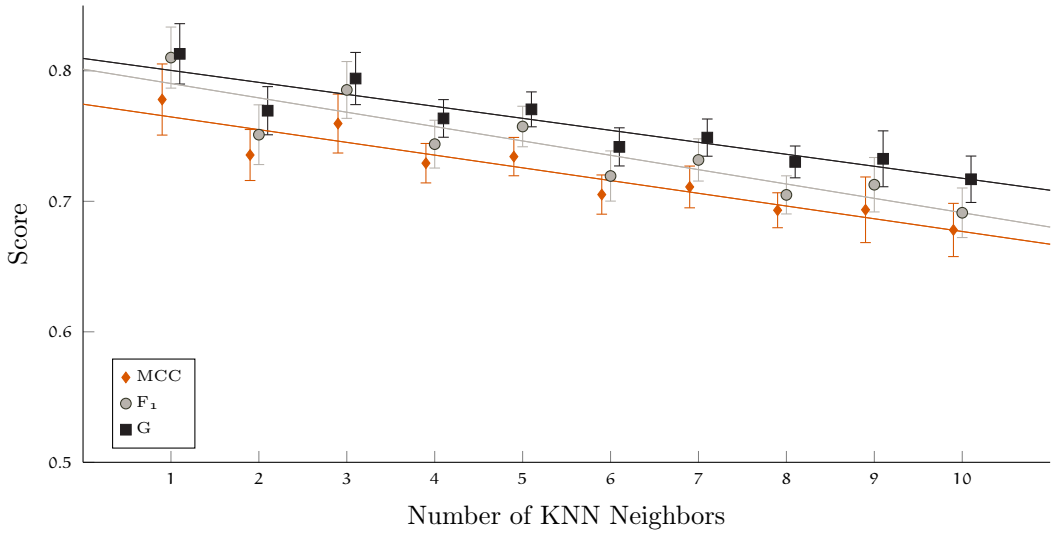


Figure 4.4: Performance measures for KNN classification for varying k with cumulative histogram features. The three classifier performance measures, MCC, F_1, and G are shown averaged across ten training and testing data permutations. Error bar lengths are $\pm$ the standard deviation value for the ten training permutations. Lines are the linear least-squares fit. Two of the three data series are offset in the x-axis to make markers more easily visible. Note that all three are trending toward worse performance as the neighborhood size increases.

the Euclidean Distance classifier and the KNN classifier. Since it was shown in 4.2.3 that increasing the neighborhood size k for KNN reduced the classification performance, KNN is only to be considered with $k = 1$. Reduction in the number of histogram bins was expected to reduce computation time for both histogram calculation and classification. Additionally, fewer histogram bins would create a smaller memory footprint since the number of features in the vector is smaller. Prior to the experiments, there was no expectation for the effects on classification performance. Rather, it was only hoped that classification performance might be enhanced with smaller feature vectors.

The best classification performance for the Euclidean classifier occurs when the number of bins $n = 14$ for standard histogram features and at $n = 9$ for cumulative histogram features, while the best performance for the KNN classifier occurs at $n = 10$ for both standard and cumulative histogram features. The performance in terms of MCC using both standard and cumulative histogram features is shown in Figure 4.5 and Figure 4.6 for the Euclidean classifier and KNN classifier, respectively. For both classifiers using standard histogram features, performance generally improves as the number of histogram bins is reduced before falling off at $n < 9$ bins. With cumulative histograms features, both classifiers maintain roughly the same classification performance as the number of bins is reduced before slightly peaking near $n = 9$ bins, then falling precipitously. For both classifiers and feature types, the MCC score becomes unstable near the peak performance level, with alternating improvement and decline before falling off to the minimum. Performance in terms of MCC is generally considered to be good for any values $MCC > 0.7$. As such, the KNN classifier could be considered a strong classifier for insect detection using either histogram type, while the baseline Euclidean classifier would be no more than adequate. Interestingly, the classification performance was better using standard

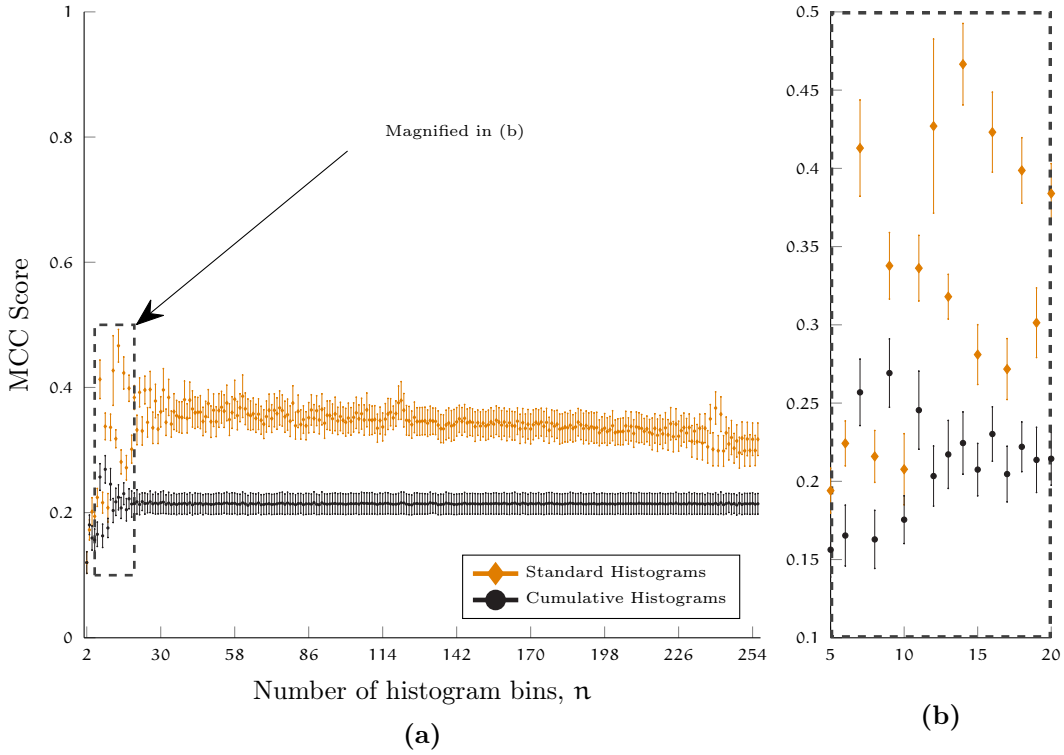


Figure 4.5: MCC scores for the Euclidean Classifier with varying histogram bin numbers. (a) shows the full X-axis for $n = 2..256$, while (b) shows the zoomed view for the area in (a) with the best performance. Error bars extend $\pm\sigma_{MCC}$ from the mean, where σ_{MCC} is the standard deviation of the MCC score across the ten testing sets.

histogram features with the Euclidean classifier while, the KNN classifier performed better with cumulative histogram features, although the overall best performance was with a KNN classifier using standard histogram features.

Classification accuracy improves in much the same way as the MCC score, as shown in Figure 4.7 and Figure 4.8 for the Euclidean and KNN classifiers, respectively. The measure of accuracy itself could be misleading due to the imbalance in the class membership between class GIC-IP and class OTHER. This imbalance in class membership means that the Prior Probability classifier would have an accuracy rate of $ACC = 82.6\%$ rather than 50% for balanced classes. The KNN classifier

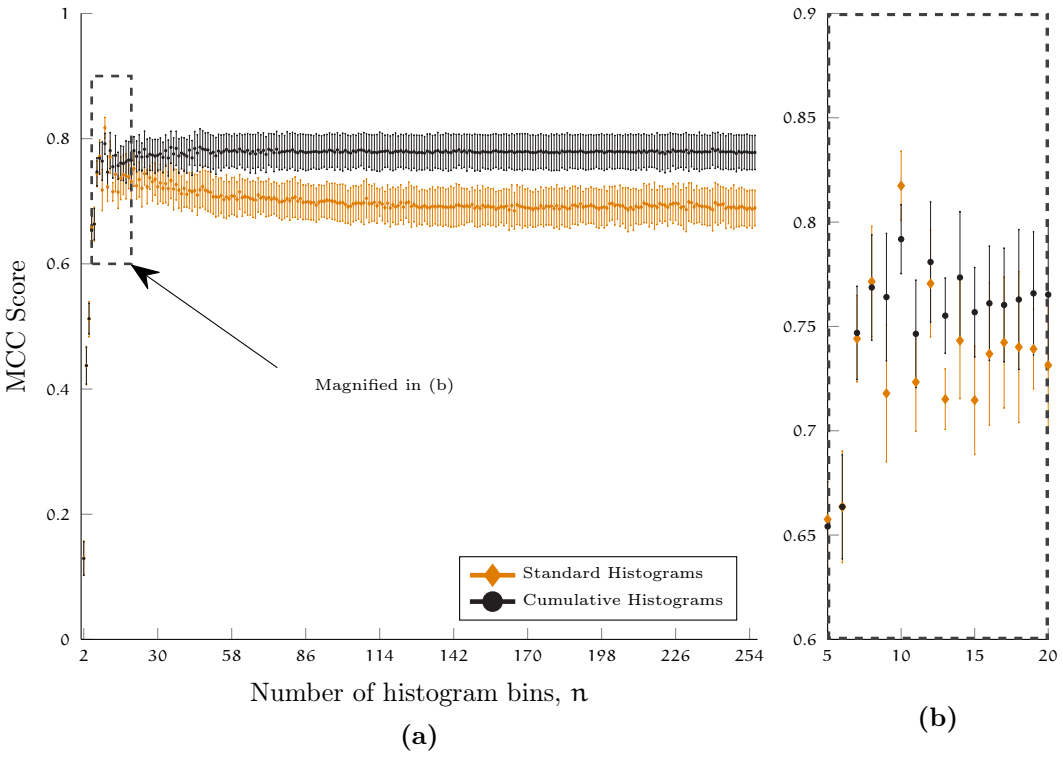


Figure 4.6: MCC scores for the KNN Classifier with varying histogram bin numbers. (a) shows the full X-axis for $n = 2..256$, while (b) shows the zoomed view for the area in (a) with the best performance. Error bars extend $\pm\sigma_{MCC}$ from the mean, where σ_{MCC} is the standard deviation of the MCC score across the ten testing sets.

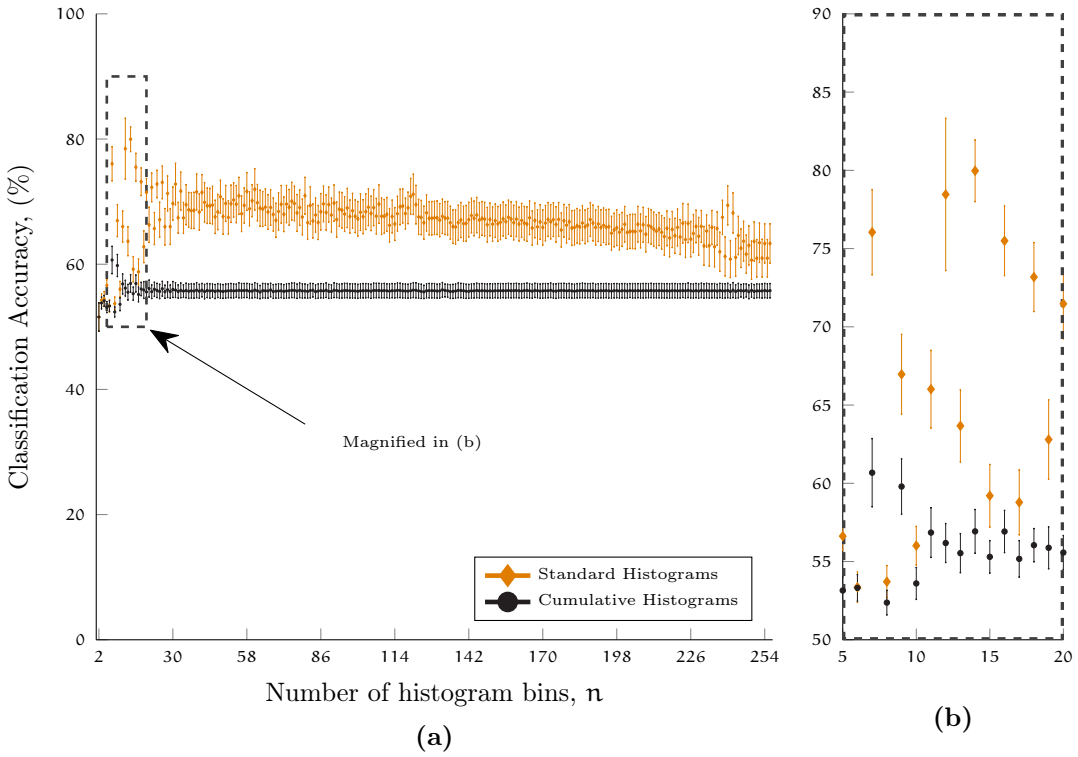


Figure 4.7: Classification accuracy of the Euclidean Classifier with varying histogram bins. (a) shows the full X-axis for $n = 2..256$, while (b) shows the zoomed view for the area in (a) with the best performance. Error bars extend $\pm\sigma_{ACC}$ from the mean, where σ_{ACC} is the standard deviation of the Accuracy across the ten testing sets.

performs better than the Prior Probability classifier for both standard and cumulative histogram features for all values of n except for the minimum accuracy level at $n = 2$ ($ACC = 75.3\%$). The best performance, in terms of accuracy, is at $n = 10$ for both histogram types, with accuracy at $ACC = 94.9\%$ for standard histograms and $ACC = 94.3\%$ for cumulative histograms. In contrast, the Euclidean classifier only approaches the accuracy of Prior Probability classifier using standard histogram features at its best case performance, where $n = 14$ and $ACC = 80.0\%$, while the best accuracy using cumulative histogram features comes at $n = 7$ where $ACC = 60.7\%$.

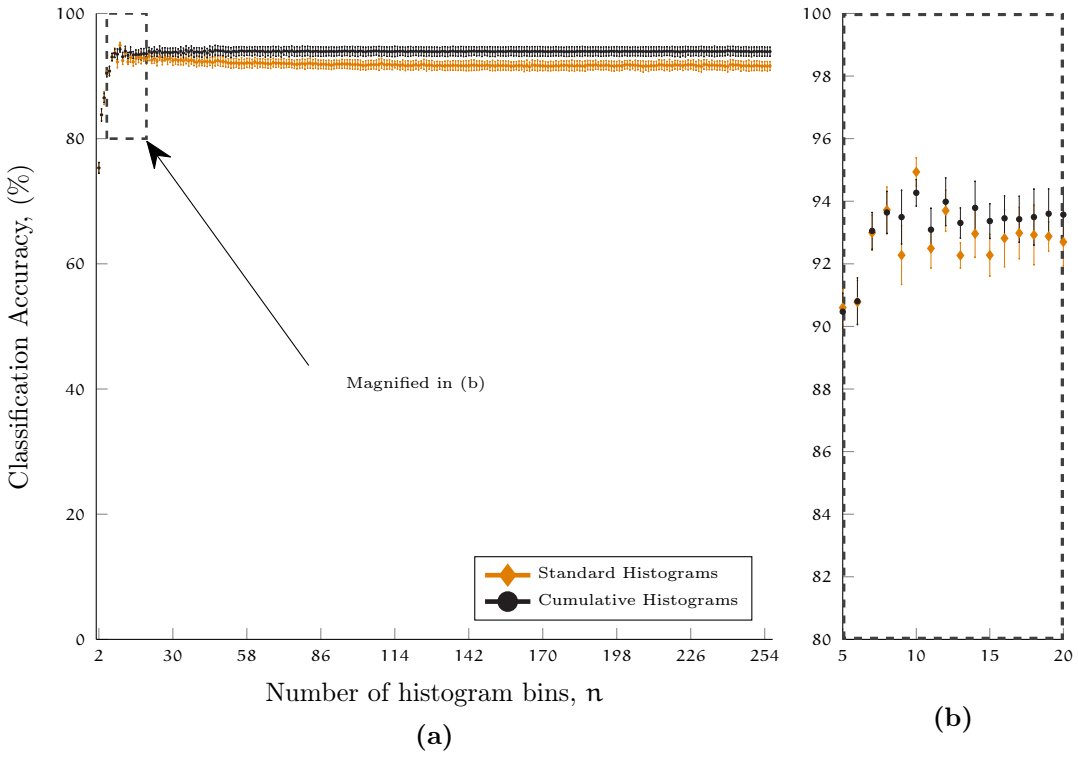


Figure 4.8: Classification accuracy of the KNN Classifier with varying histogram bins. (a) shows the full X-axis for $n = 2..256$, while (b) shows the zoomed view for the area in (a) with the best performance. Error bars extend $\pm\sigma_{ACC}$ from the mean, where σ_{ACC} is the standard deviation of the Accuracy across the ten testing sets.

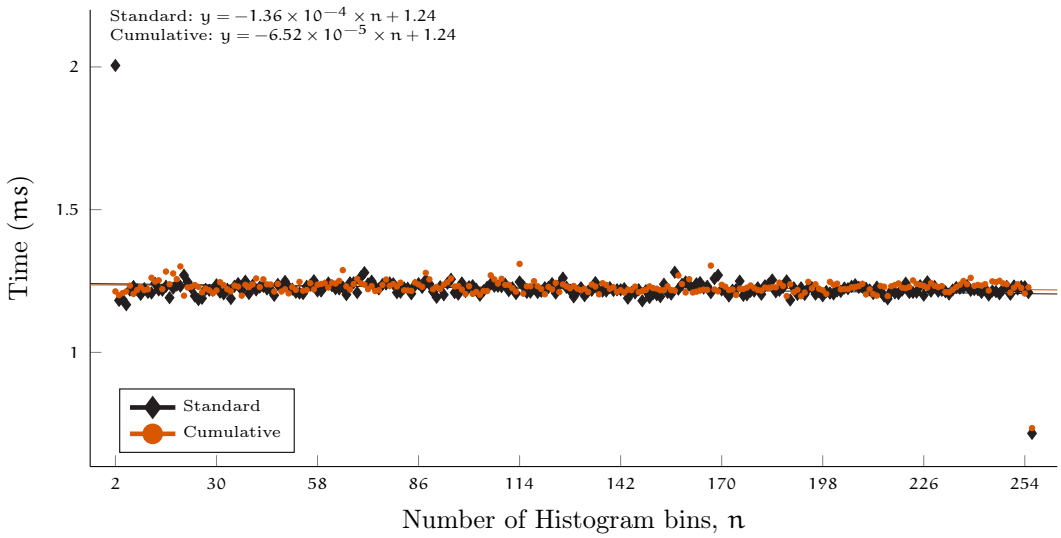


Figure 4.9: Time to compute standard and cumulative histogram features for 8 bit, 640×480 grayscale images for varied number of histogram bins. Histogram computation is based on number of pixels, and remains relatively constant since the number of operations is the same no matter the number of bins.

Results in this section suggest that improved insect presence detection can be achieved using fewer histogram bins in the feature vectors, and that performance of the classification is either minimally impacted in the case of cumulative histogram features, and even improved when using standard histograms. In terms of computation time, a reduction in the number of bins used for classification is beneficial as it provides a corresponding reduction in computation for classification, but not for histogram calculation. Figure 4.9 shows the nearly constant computation time for standard and cumulative histograms ranging from $n = 2$ to $n = 256$.

The computation time savings from reducing the number of histogram bins occur when images are classified, although the computation savings are much greater for the KNN classifier. Figure 4.10 shows the average time required to classify an image histogram using KNN and Euclidean classifiers for both standard and cumulative histogram feature vectors. As expected, the classification time for a given

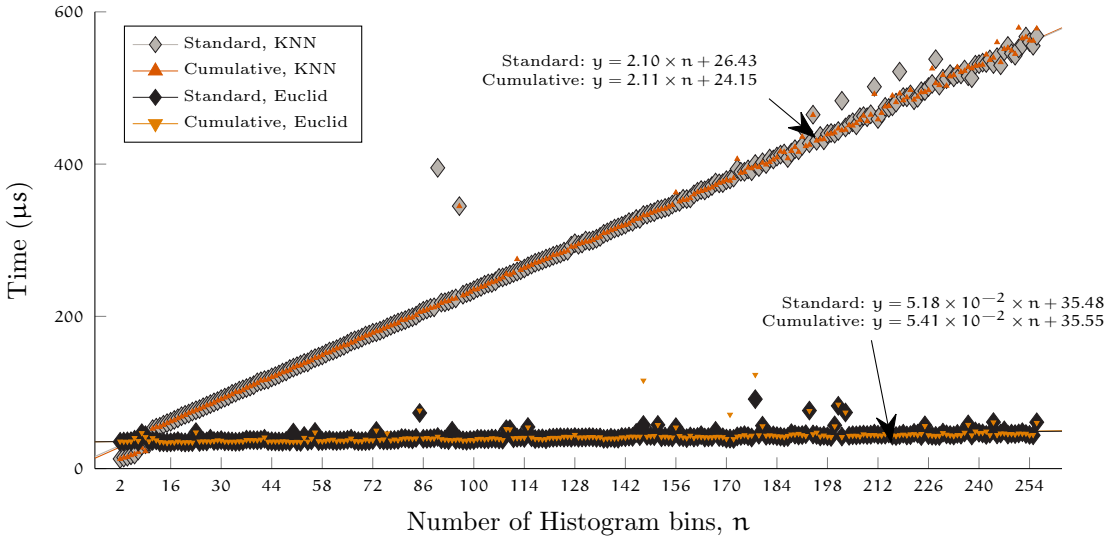


Figure 4.10: Time to train and test Euclidean and KNN classifiers with a single image for varied number of histogram bins. Reduction in number of histogram bins provides more reduction in classification time for the KNN classifier than for the Euclidean distance classifier.

classifier and histogram type mirrors that of the same classifier with the other type of histogram features. Both KNN and Euclidean classifiers show reduction roughly linear reduction in computation time as the number of histogram bins is reduced. In the case of Euclidean classifiers, each $1 \times n$ reduction in histograms bins corresponds to a $52\,ns$ reduction in computation time, while the same $1 \times n$ reduction in histogram bins results in a $2.1\,\mu s$ reduction in computation time. Therefore, the computation savings for reduced-n histograms are forty times greater for the KNN classifier than they are for the Euclidean distance classifier.

4.2.5 *Variation of Sub-Sampling Interval*

Sub-sampling of input images is a potential way to reduce processing and memory usage without creating new variables in the system. Unlike variation of histogram bins in 4.2.4, the reduced computation requirements for generating the histogram

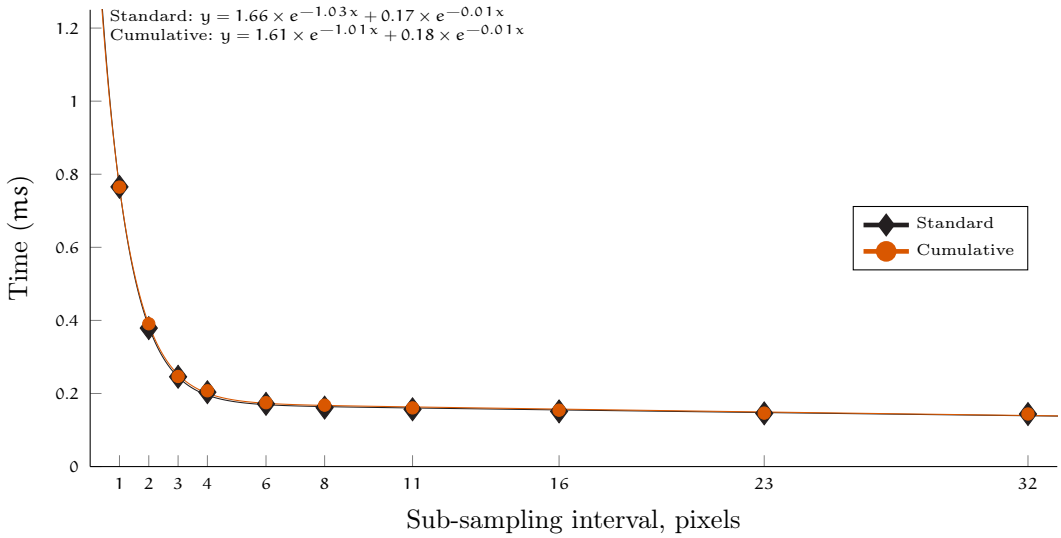


Figure 4.11: Time to compute standard and cumulative histogram features for 8 bit, 640×480 grayscale images for varied image subsampling interval. Histogram computation is based on number of pixels, resulting in a reduction in computation time as the image is more coarsely sampled.

features are impacted rather than that of the classification. Figure 4.11 shows the time required to compute histogram features with varied levels of input image sub-sampling. The time required for histogram computation follows a roughly exponential curve where the time improvements for increased sub-sampling interval flatten as the sub-sampling interval is increased. Figure 4.12 shows the time required for training and classification of images based on their histograms for varied levels of input image sub-sampling. As expected, the computation time for training and classification does not depend on the sub-sampling interval, with time remaining roughly flat across the sub-sampling interval axis.

Sub-sampling the images has little impact on the performance of classification for either type of classifier or histogram. Figure 4.13 shows the classification performance of the Euclidean Distance classifier in terms of MCC score for various subs-sampling intervals, for histograms with 256 bins and 16 bins. Figure 4.14

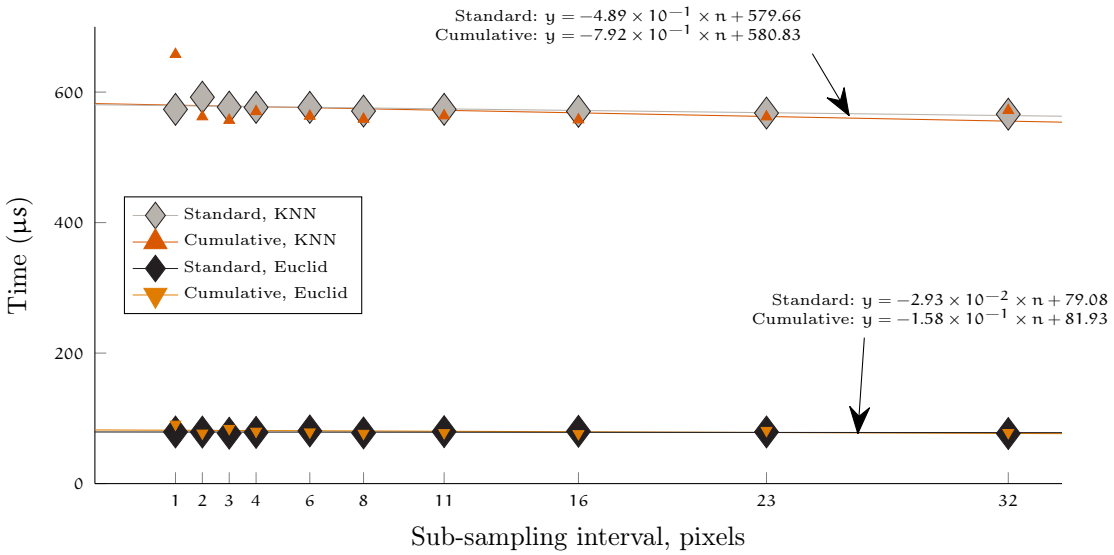
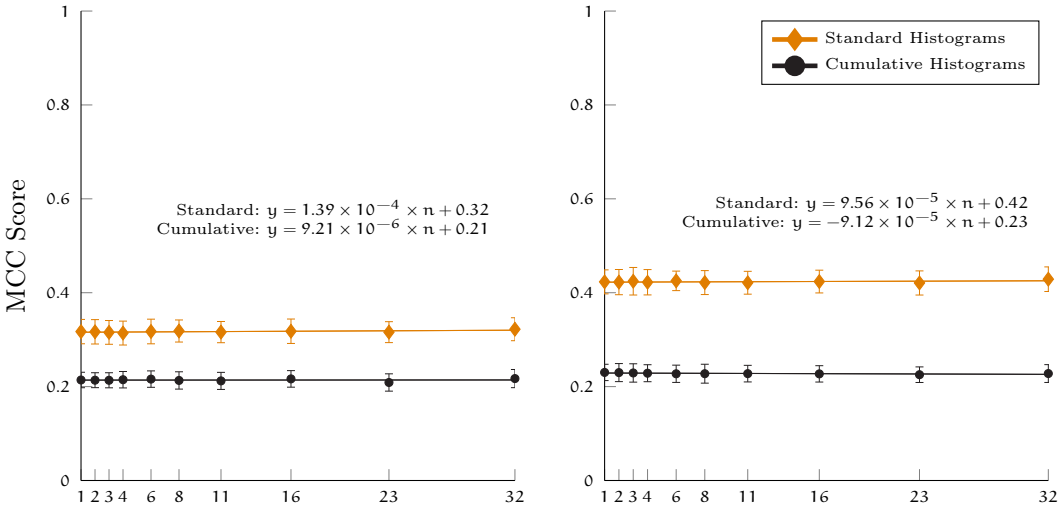


Figure 4.12: Time to train and test Euclidean and KNN classifiers with a single image for varied image subsampling interval. Training and testing time is not significantly impacted by image subsampling.

shows the classification performance of the KNN classifier in terms of MCC score for various subs-sampling intervals, for histograms with the same 256 bins and 16 bins. Classification performance remains nearly constant across the sub-sampling interval axis for nearly all combinations of classifier and histogram feature type, with the exception of standard histogram features using the KNN classifier where there is a noticeable negative impact on classification due to increased sub-sampling interval.

Figure 4.15 shows the classification performance surfaces across both the number of histogram bins and the sub-sampling interval. Based on this figure the best choice for classifier type, number of histogram bins, and sub-sampling interval for binary classification was the KNN classifier with 10 histogram bins, sub-sampled at the maximum sub-sampling interval of every 32 pixels. Operating with these parameters provides nearly the maximum savings in classification computation time due to the reduced number of histogram bins, operating in the range where the cost



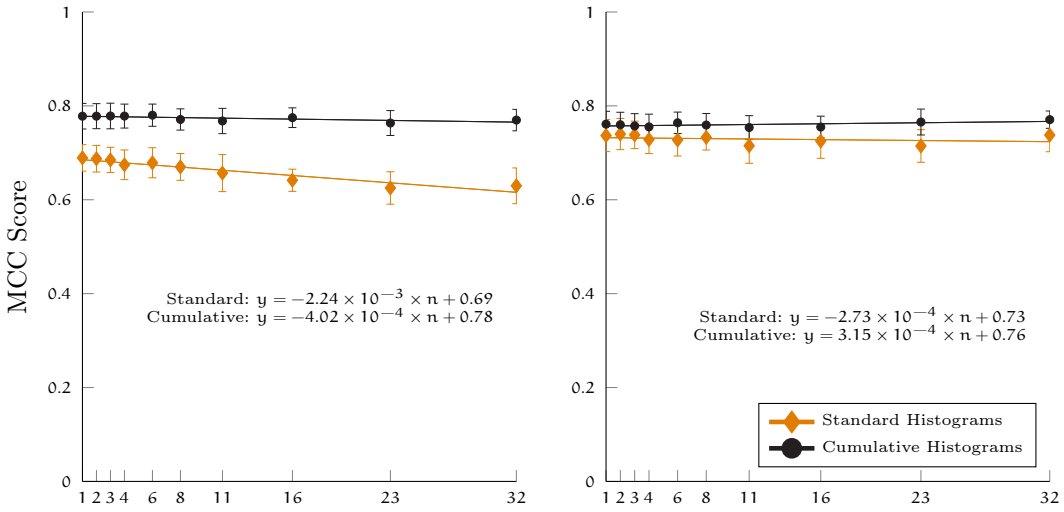
Subsampling: Sample every m pixels, $n = 256$ bins

(a)

Subsampling: Sample every m pixels, $n = 16$ bins

(b)

Figure 4.13: MCC of Euclidean Classifier with varied image subsampling. (a) shows the effects of image subsampling interval for histograms with 256 bins, while (b) shows the same effects for histograms with only 16 bins. The effect of subsampling interval on classification MCC is much less than the effect of number of histogram bins.



Subsampling: Sample every m pixels, $n = 256$ bins **(a)**

Subsampling: Sample every m pixels, $n = 16$ bins **(b)**

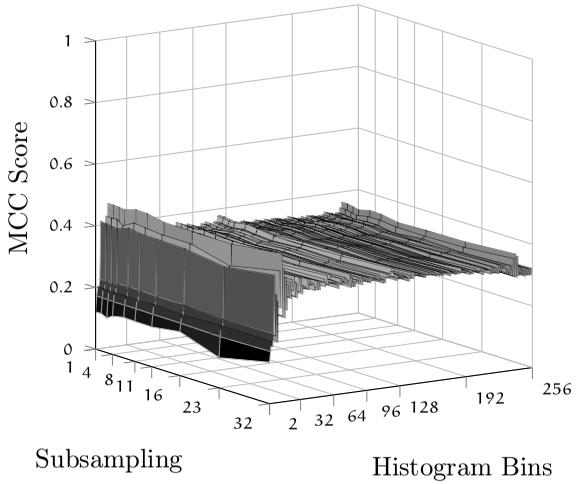
Figure 4.14: MCC of KNN Classifier with varied image subsampling. (a) shows the effects of image subsampling interval for histograms with 256 bins, while (b) shows the same effects for histograms with only 16 bins. The effect of subsampling interval on classification MCC is much less than the effect of number of histogram bins, with the only case of performance reduction due to increased sampling interval coming for 256 bin standard histograms.

of using the KNN classifier versus the Euclidean Distance classifier is negligible. It also provides the maximum savings in histogram computation time due to the sub-sampling of the image. Additionally, the classifier performance is better with these parameters than using the maximum number of histogram bins and the full input image.

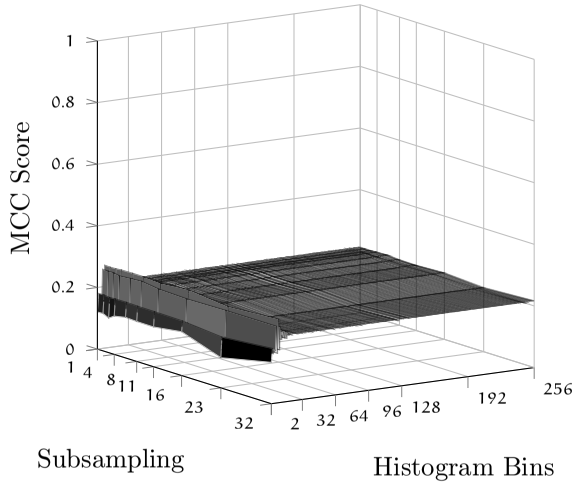
4.2.6 *Histogram Adjustment for Contrast Enhancement*

Images from the instrumented trap will need to be adjusted for contrast in order to make it easier for insect shapes to be extracted from the overall image. This step, while essential to the overall insect identification process, could be done either before or after insect detection is attempted. If the images are adjusted prior to insect detection, the resulting adjusted image and it's corresponding histograms could potentially either improve or worsen insect detection performance while adding computation time to the overall insect detection algorithm. Contrast adjustment does not change the number of histogram bins or the sub-sampling interval, and has an impact on the classification performance potential while adding computation time for the adjustment itself. This section describes the impacts of adjusting images for contrast prior to performing insect detection.

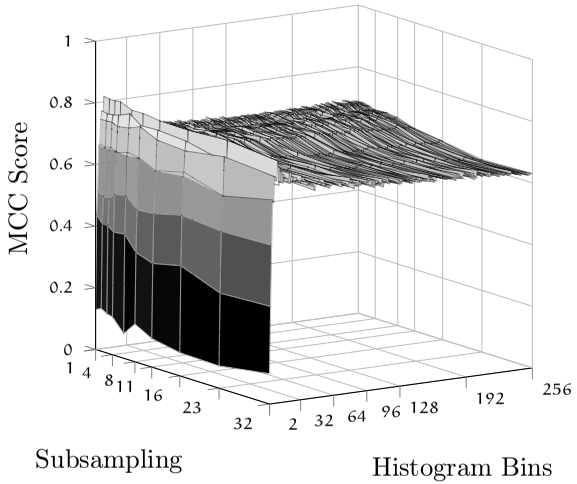
Contrast adjustment prior to insect detection improves performance of the classifiers in nearly all cases. For the Euclidean Distance classifier, the average MCC score performance is improved by 8% and 28% for standard and cumulative histogram features, respectively. For the KNN classifier, the same MCC score improvement is 11% and 7.5%. Adjusting the histogram for contrast prior to insect detection improves the performance of the KNN classifier with cumulative histogram features such that MCC > 0.8 for all histogram bin levels where $n > 6$. Recall that MCC



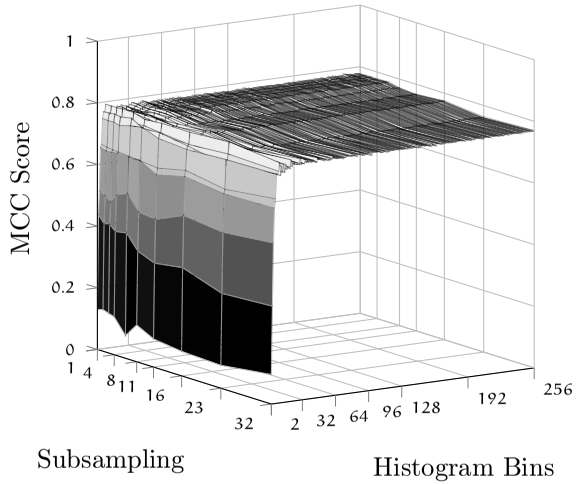
(a) Euclidean Distance Classifier, Standard Histograms



(b) Euclidean Distance Classifier, Cumulative Histograms



(c) KNN Classifier, Standard Histograms



(d) KNN Classifier, Cumulative Histograms

Figure 4.15: MCC Score surface across histogram bins and sub-sampling. The performance relationships remain roughly the same across the two variables. Classification performance is better with standard histogram features using the Euclidean Distance Classifier, and better with cumulative histogram features using the KNN Classifier.

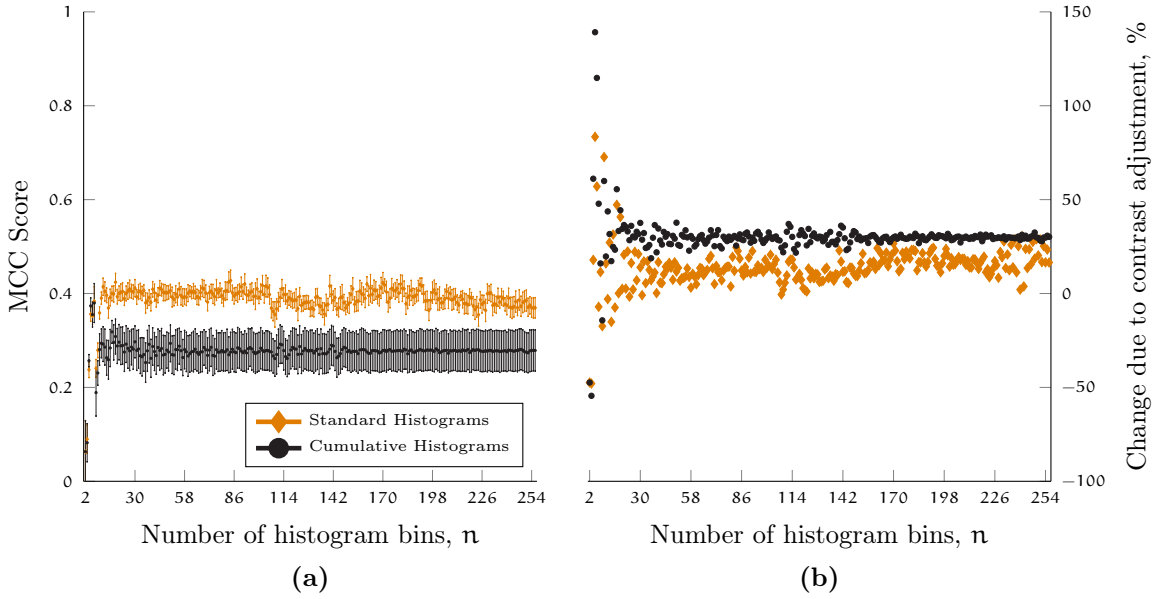


Figure 4.16: Effect of contrast adjustment on Euclidean Distance performance. a) MCC score versus number of histogram bins for insect detection classification, and b) Percent improvement in MCC score when compared to the performance of the Euclidean Distance classifier without performing contrast adjustment.

scores greater than 0.7 are considered to be strong classifiers. Such improvements in classification performance do come at a cost in computation time, as contrast adjustment for a 680×480, 8 bit image takes 1.5 ms on average, while computation of the histogram features takes 1.1 ms.

To address the "comb effects" due to the contrast adjustment described in Figure 3.8, histogram smoothing is employed after histogram adjustment. Smoothing the adjusted histograms has little impact on computation time, requiring just 93 µs per image. Additionally, any potential improvements in classification performance are, at best, unpredictable, and at worst, actually damaging to classification performance. Figure 4.18 and Figure 4.19 show the MCC scores and the relative improvement in MCC due to smoothing the adjusted histograms for the Euclidean Distance and KNN classifiers, respectively. For the Euclidean Distance classifier with stan-

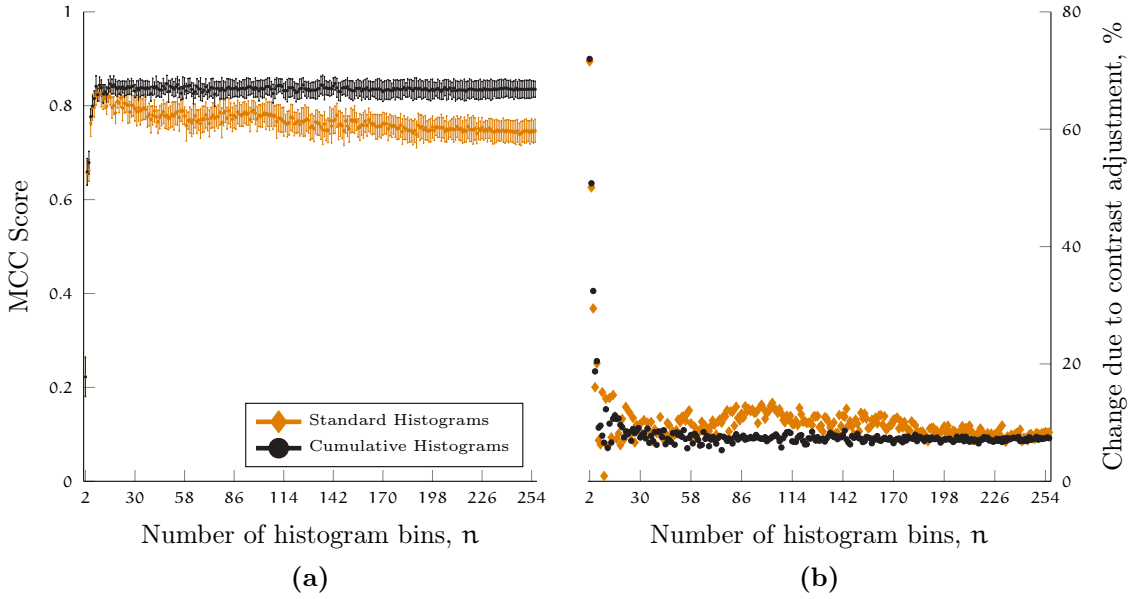


Figure 4.17: Effect of contrast adjustment on KNN performance. a) MCC score versus number of histogram bins for insect detection classification, and b) Percent improvement in MCC score when compared to the performance of the KNN classifier without performing contrast adjustment.

dard histogram features, the effect of smoothing the adjusted histogram is almost wholly negative. With cumulative features, there is improvement in performance for $n > 200$ histogram bins, but worse improvement for $n \leqslant 200$. In the case of the KNN classifier with cumulative histograms, the effects of smoothing are only positive for $n > 230$, while smoothing has a negative effect for all cases $n \leqslant 230$. Interestingly, the KNN classifier with standard histograms sees improved performance for nearly all numbers of histogram bins $n > 25$, and even rivals the performance of adjusted cumulative histogram features where MCC > 0.8 for $25 < n \leqslant 128$, while the cumulative features had shown better KNN performance in all previous tests.

Contrast adjustment by itself provides predictable, positive improvement in classification performance at the cost of greater computational intensity. As such, con-

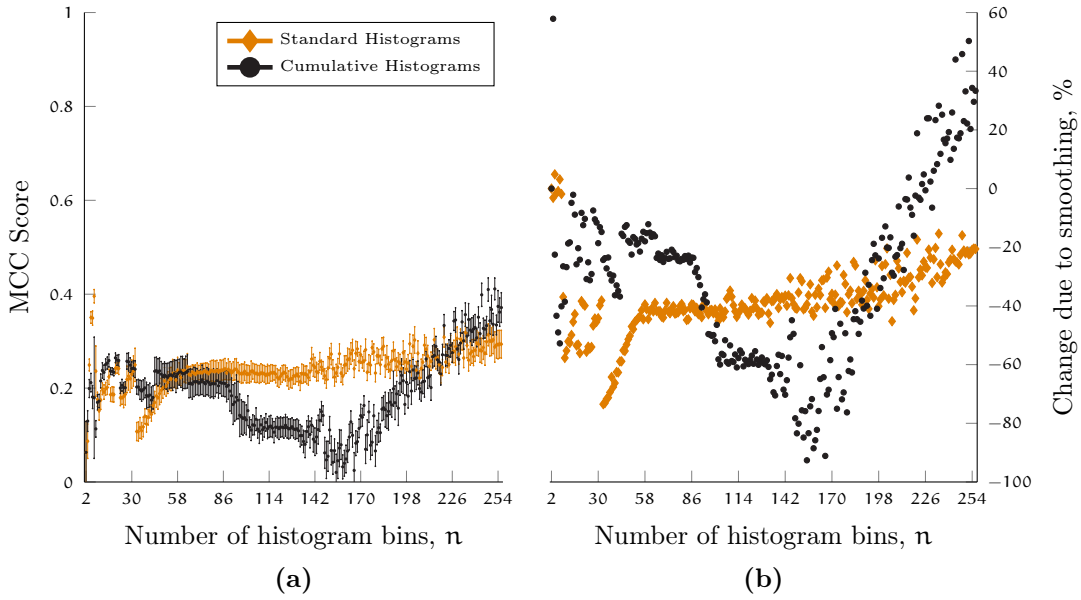


Figure 4.18: Effect of histogram smoothing on Euclidean Distance performance. a) MCC score versus number of histogram bins for insect detection classification, and b) Percent improvement in MCC score when compared to the performance of the Euclidean Distance classifier on contrast-adjusted histograms without smoothing.

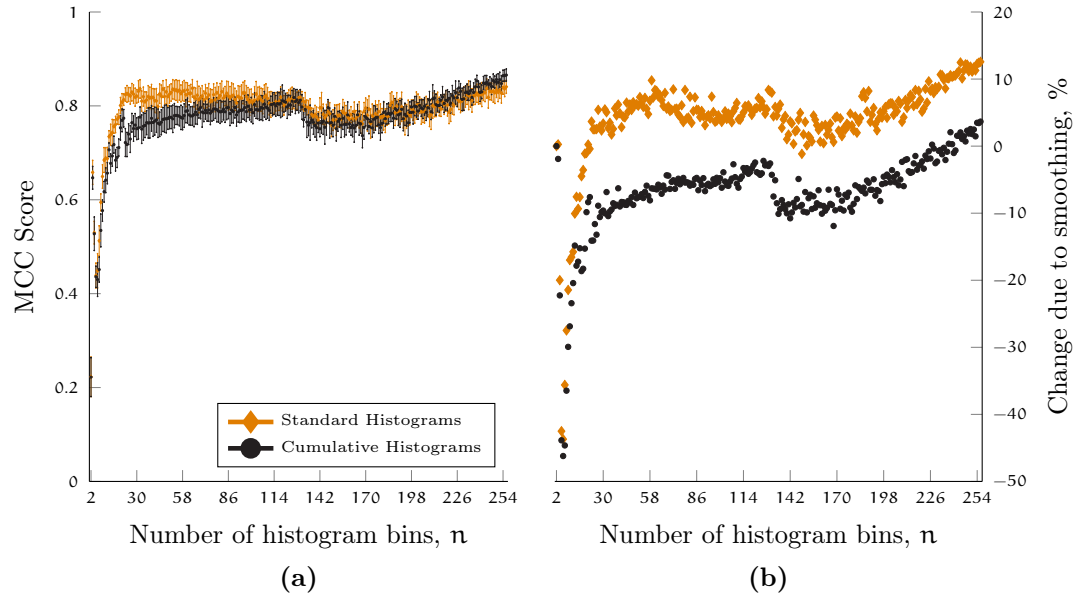


Figure 4.19: Effect of histogram smoothing on KNN performance. a) MCC score versus number of histogram bins for insect detection classification, and b) Percent improvement in MCC score when compared to the performance of the KNN classifier on contrast-adjusted histograms without smoothing.

trast adjustment should be considered as a step prior to insect detection classification if the microcontroller used for image capture and insect detection are not memory-constrained. Smoothing the contrast adjusted histograms is not predictable, and often results in poorer classification performance. Smoothing should not be performed without some future compelling reason.

4.2.7 *Multi-Class Classification Effects*

All results presented to this point have been based on binary classification for insect detection in which images were considered to be in class GIC-IP or OTHER. Additional improvements in classification results could reasonably be expected if the main image classes were broken into sub-classes, allowing for improved discernment ability of the classifiers due to differentiation of similar sub-classes prior to training. This section presents the results of dividing the images into the two classes Good Image Capture (GIC) and Bad Image Capture (BIC), and further into the sub-classes GIC-Insect Present (GIC-IP), GIC-No Insect (GIC-NI), BIC-Backlight Malfunction Type 1 (BIC-BM1), BIC-Backlight Malfunction Type 2 (BIC-BM2), BIC-Over Saturated (BIC-OS), and BIC-Shutter Failure (BIC-SF). The sub-class GIC-NI and all sub-classes of BIC were considered part of the OTHER group for binary classification.

In multi-class classification, there are no suitable analogs for the performance scores F_1, G, and MCC that are used to evaluate and compare classification performance. Even simple accuracy may be a poor way to evaluate classification performance in the case that one type of classification is more "costly" than another. In this research, the ability to discern the GIC-IP class from the rest is of much greater importance than classification between the other classes. As such, the classifica-

tion results are pooled for all non-GIC-IP instances. This means that, even when considering multiple classes to potentially improve classification performance, the performance scores treat a BIC-SF image classified incorrectly as GIC-NI as a true negative rather than a mis-classification since both the true class and the predicted class membership are not GIC-IP.

Multi-class classification improves the performance of the Euclidean Distance classifier for any number of histogram bins $n > 6$ using both standard and cumulative histogram features by 10.4% and 11.6%, respectively. The improvement in classification performance remains roughly constant for cumulative histogram features while increasing in a linear fashion for standard histogram features as the number of histogram bins increases. Figure 4.20 shows the MCC score for multi-class Euclidean Distance classification with varied numbers of histogram bins, as well as the improvement in performance when compared to the binary classification task.

Perhaps surprisingly, the use of multi-class classification results in exactly the same performance as the binary case when using the KNN classifier, with no improvement or degradation in classification performance no matter the number of histogram bins. Figure 4.21 shows the performance of the KNN classifier for multi-class classification, as well as the difference in performance when compared to the binary classification task. Table 4.9 is a confusion for the multi-class KNN classifier, while Table 4.10 is the confusion matrix for the same parameters using binary KNN classification. Since all images classified as other than GIC-IP are considered to be true negatives for the performance measures, the resulting MCC score is the same using either binary or multi-class KNN classification. Every combination of histogram bins, subsampling, contrast adjustment, or histogram smoothing results in the same equivelant performance between binary KNN and multi-class KNN classification.

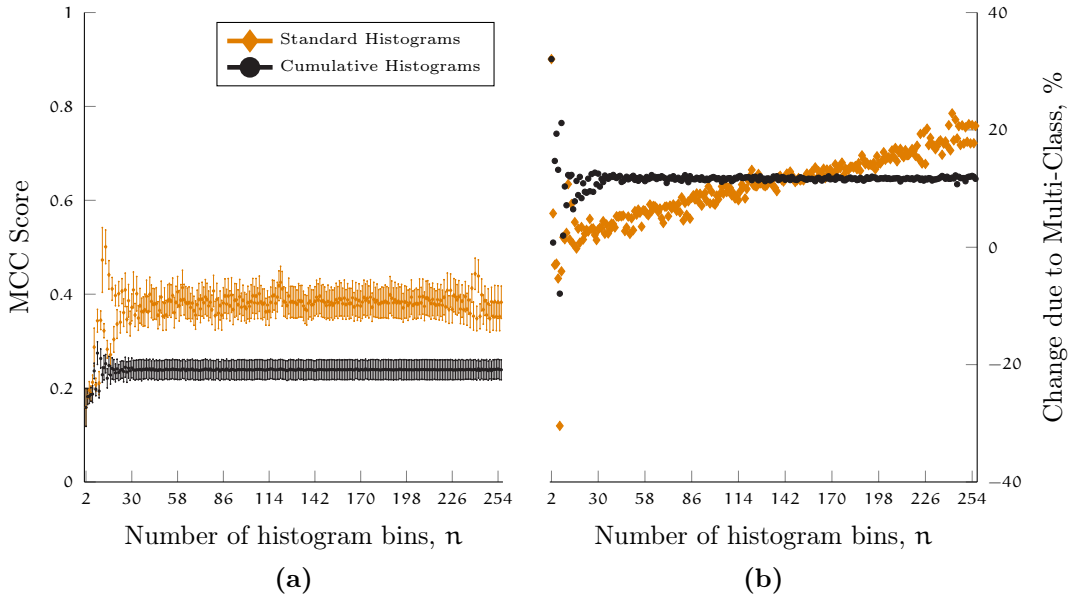


Figure 4.20: Effect of Multi-class classification on Euclidean Distance performance. a) MCC score versus number of histogram bins for insect detection classification, and b) Percent improvement in MCC score when compared to the performance of the Euclidean Distance classifier binary classification performance.

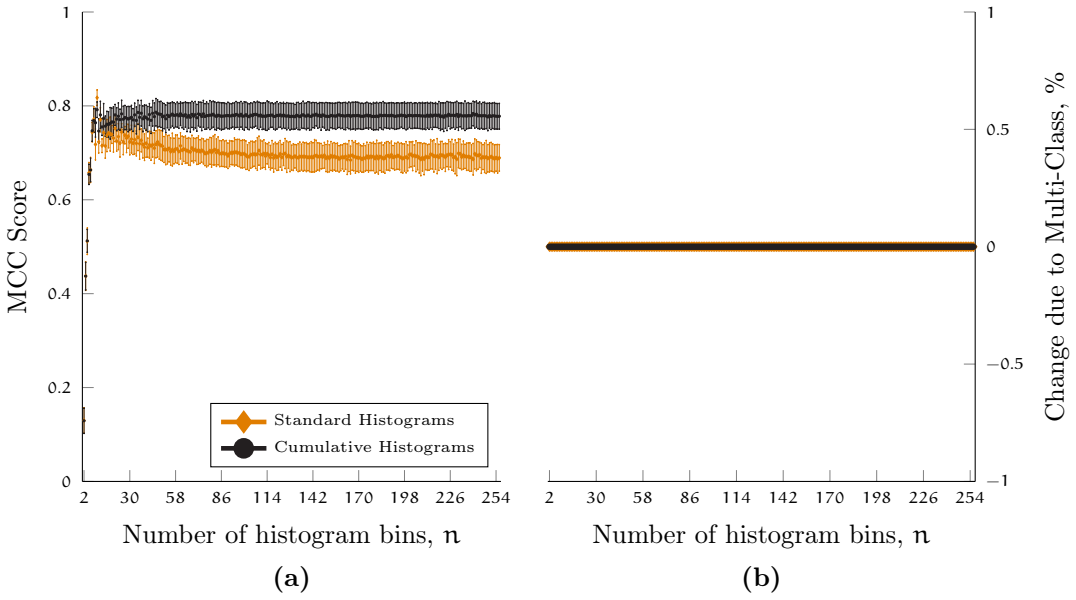


Figure 4.21: Effect of Multi-class classification on KNN performance. a) MCC score versus number of histogram bins for insect detection classification, and b) Percent improvement in MCC score when compared to the performance of the KNN classifier binary classification performance.

Table 4.9: Confusion matrix for multi-class KNN classification using standard histogram features with 256 histogram bins, unity sub-sampling, and $k = 1$ neighbor.

		Predicted Membership					
		GIC-IP	GIC-NI	BIC-BM_1	BIC-BM_2	BIC-OS	BIC-SF
Actual Membership	GIC-IP	**1167**	600	0	0	3	0
	GIC-NI	251	**7801**	1	0	7	0
	BIC-BM_1	0	4	**171**	6	9	0
	BIC-BM_2	0	0	1	**69**	0	0
	BIC-OS	0	0	0	0	**60**	0
	BIC-SF	0	0	0	0	0	**40**

Note: Table entries along the bold diagonal are counts of correctly classified images. The sum of entries across rows would be the number of images belonging to each class, while the column sums would be the number of images classified in each class. All entries off of the bold diagonal are incorrect classifications.

Multi-class classification does provide for improvement when using the Euclidean Distance classifier for contrast adjusted images as well as smoothed histograms from contrast adjusted images. Figure 4.22 shows the MCC scores for contrast adjusted image histograms, while Figure 4.23 shows the MCC scores for smoothed histograms from contrast adjusted images, both using the Euclidean Distance classifier. Generally, both types of features have improved classification performance with the multi-class classifier. However, neither is improved to the extent that performance is on par with the KNN classifier.

Using a multi-class classification does not impact the computational requirements of generating the histogram features, as it is independent of their generation. Additionally, the multi-class case has little influence on the time required to perform classification in the case of KNN classification, while generally taking longer in the case of Euclidean Distance classification. Figure 4.24 shows the relative increase in

Table 4.10: Confusion matrix for binary KNN classification using standard histogram features with 256 histogram bins, unity sub-sampling, and $k = 1$ neighbor.

		Predicted	
		GIC-IP	OTHER
True	GIC-IP	**1167**	603
	OTHER	251	**8169**

Note: Bold entries along the diagonal are counts of correctly classified images, while off-diagonal entries are incorrect classifications. The sum of all entries in each confusion matrix is the total number of images in the testing set. Diagonal entries represent the True Positives (TP) and True Negatives (TN) in the first and second rows, respectively. Off-diagonal entries represent the False Negative (FN) and False Positive (FP) counts in the first and second rows, respectively. FP are considered Type-I error while FN are Type-II error.

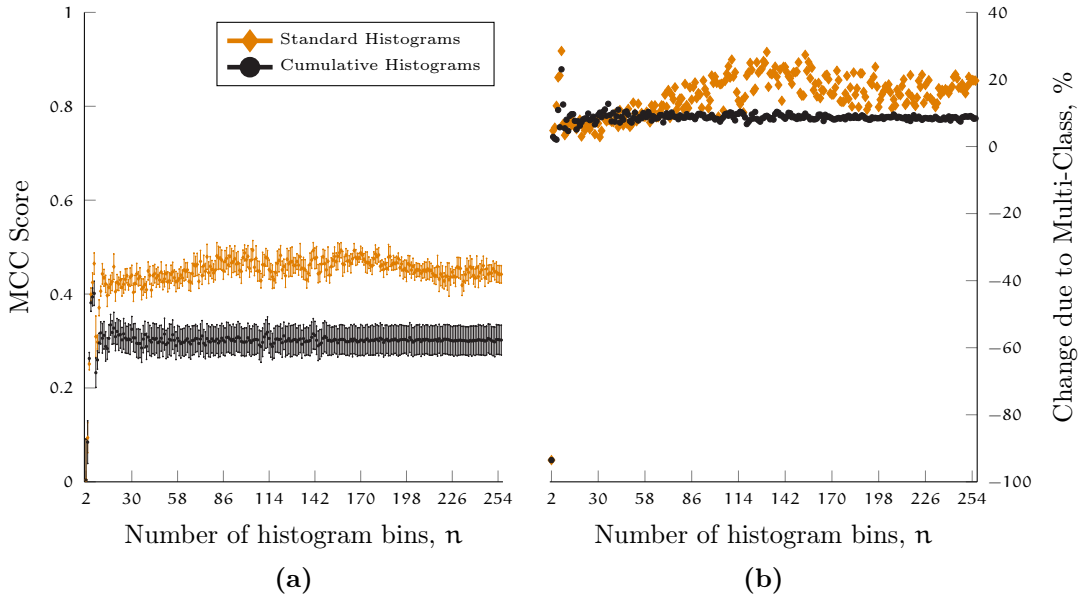


Figure 4.22: Effect of multi-class on contrast adjusted histograms for Euclidean Distance classifier. a) MCC score versus number of histogram bins for insect detection classification, and b) Percent improvement in MCC score for multi-class classification compared to binary classification.

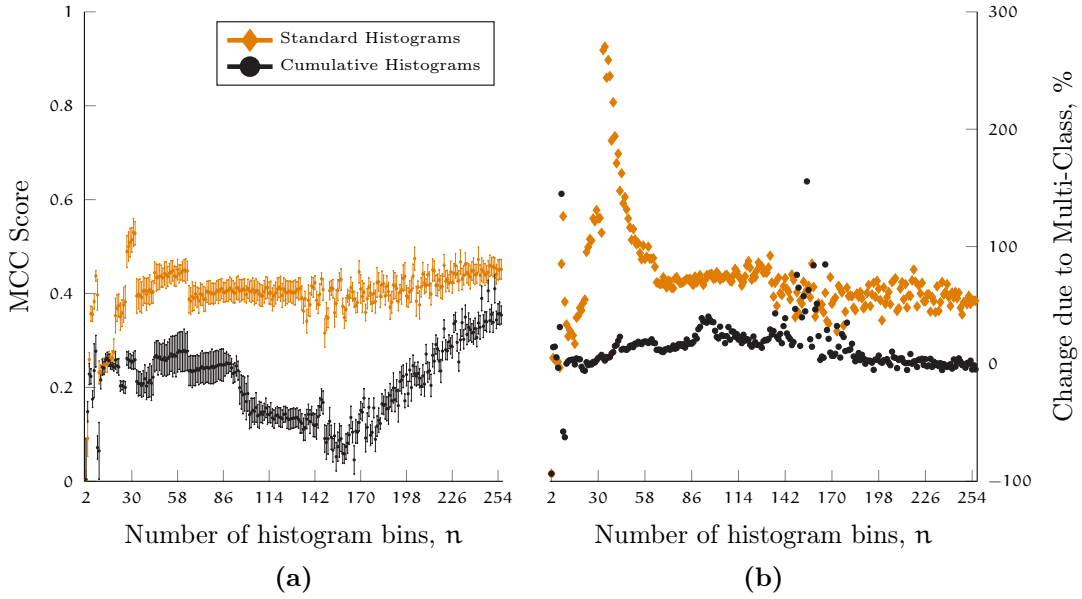


Figure 4.23: Effect of multi-class on smoothed histograms for Euclidean Distance classifier. a) MCC score versus number of histogram bins for insect detection classification, and b) Percent improvement in MCC score for multi-class classification compared to binary classification.

classification time for multi-class classification when compared to binary classification.

4.2.8 *Summary*

The best insect detection performance in this research achieves an accuracy rate of

4.3 PROPOSED INSECT IDENTIFICATION KERNEL PERFORMANCE

The proposed insect identification kernel in this research uses angular-radial basis functions computed on the shape of insects. An optimal system would have excellent recognition rates while using as little computational requirements as possible. As such, the parameters presented in this section for comparison of different features include computation time, memory requirements, and classification perfor-

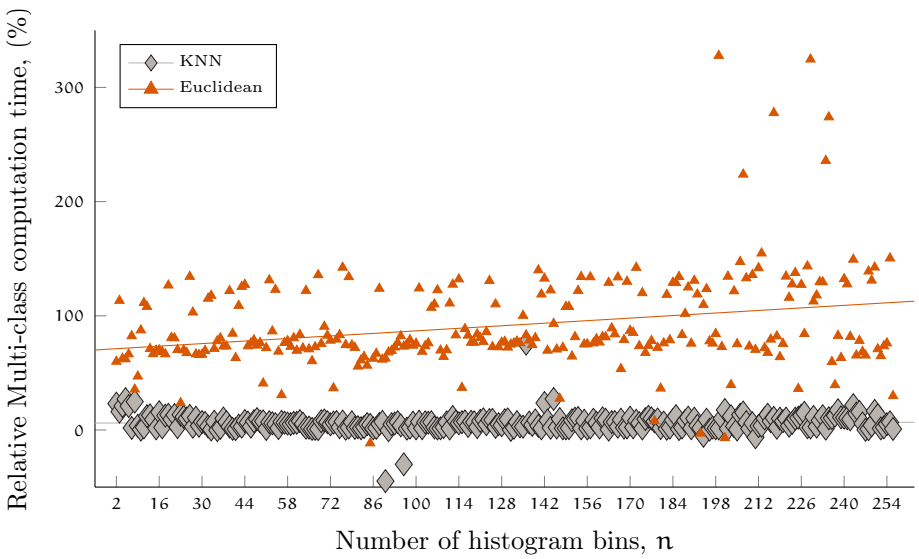


Figure 4.24: Relative time to train and test Euclidean and KNN multi-class classifiers versus binary classifiers. The number of histogram bins has little effect on the relative time required. KNN classification is very close to the same speed in the binary and multi-class cases, while multi-class classification takes longer than binary classification for the Euclidean Distance Classifier.

mance. Four angular-radial moment types Zernike Moments (ZM), Pseudo-Zernike Moments (pZM), Fourier-Mellin Moments (FMM), and MPEG-7 Angular Radial Transform (ART) are presented as separate feature vectors and combined feature vectors with combinations of the four. Finally, the number of features of each type is reduced based on methods of Principle Component Analysis (PCA) and Fisher Multiple Discriminant Analysis (FMDA) to determine if there are features that best represent prototypical insects for discernment from other classes.

Zernike Moments (ZM) features were the point of departure for this research based on the findings of Ashaghathra (2008). In his work, he found the best performance of ZM for pecan weevil identification to be with order $p = 3$ using manually positioned insects post-mortem and in the laboratory setting. In this research, the majority of the testing and training images were field collected of live insects in their natural poses. Additionally, Ashaghathra used an algorithm that approximated the moment

integration and discarded portions of each insect shape to fit the circular nature of acZM and the resulting circle-to-square incompatibility (See Figure 3.21). This research used the Cartesian to Polar Image Transformation in Figure 3.20, and thus uses the entire insect shape. In the next four sections, results are presented based on binary classification using the Support Vector Machine (SVM) classifier. A comparison of the performance using binary and multi-class Naive Bayes (NB) classifiers follows in 4.3.3.

4.3.1 *Individual Moment Feature Performance*

4.3.1.1 *Zernike Moment Performance*

Zernike Moments (ZM) classification performance generally improves as the maximum ZM order increase, from order $p_{max} = 1$ to $p_{max} = 26$, then remain relatively unchanged for $p_{max} < 55$. The maximum accuracy for ZM features at order $p = 26$ is $Acc = 98.0\%$, with error rates $Error_I = 0.07\%$ and $Error_{II} = 1.97\%$. Figure 4.25 shows the $Error_I$ and $Error_{II}$ rates and the time required to perform one classification for varied levels of p_{max} from 1 to 60. Classification time increases in an exponential manner as p_{max} increases, which is as expected since the number of features in the feature vector for ZM also increases exponentially with p_{max}.

For $p_{max} > 55$, the orthogonality of the ZM begins to break down due to decimation and the scale of images, resulting in a degenerate training matrix. This leads to a sudden increase in $Error_I$ rate, which minimal for $p_{max} \leqslant 55$, and a drop in computation time for completing a classification due to the reduced dimensionality of the resulting classifier model.

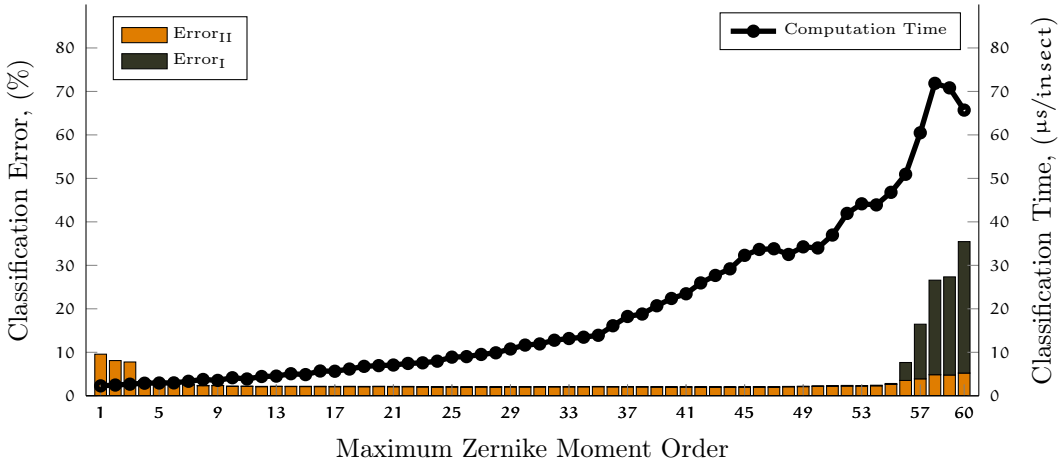


Figure 4.25: ZM classification performance for increasing moment order. Error_{I} and Error_{II} denote Type-I and Type-II error, respectively. Error rates are stacked, with the top of each stacked bar representing the total error rate. ZM have fewer repetitions for a given order resulting in fewer overall features for any maximum order p. ZM are shown to order $p = 60$, whereas pZM and FMM are only used to order $p = 30$.

4.3.1.2 *Psuedo-Zernike Moment Performance*

Pseudo-Zernike Moments (pZM) classification performance generally improves as the maximum pZM order increases, from order $p_{max} = 1$ to $p_{max} = 21$, then remains relatively unchanged for $p_{max} < 27$. The maximum accuracy for pZM features at order $p = 21$ is 97.3%, with error rates of $\text{Error}_{\text{I}} = 0.06\%$ and $\text{Error}_{\text{II}} = 2.60\%$. Figure 4.26 shows the Error_{I} and Error_{II} rates and the time required to perform one classification for varied levels of p_{max} from 1 to 35. Classification time increases in an exponential manner as p_{max} increases, which is as expected since the number of features in the feature vector for pZM also increases exponentially with p_{max}.

For $p_{max} > 27$, the orthogonality of the pZM again begins to break down due to decimation and the scale of images, creating a degenerate training matrix. This leads to a sudden increase in Error_{I} rate, which were minimal for $p_{max} \leqslant 27$,

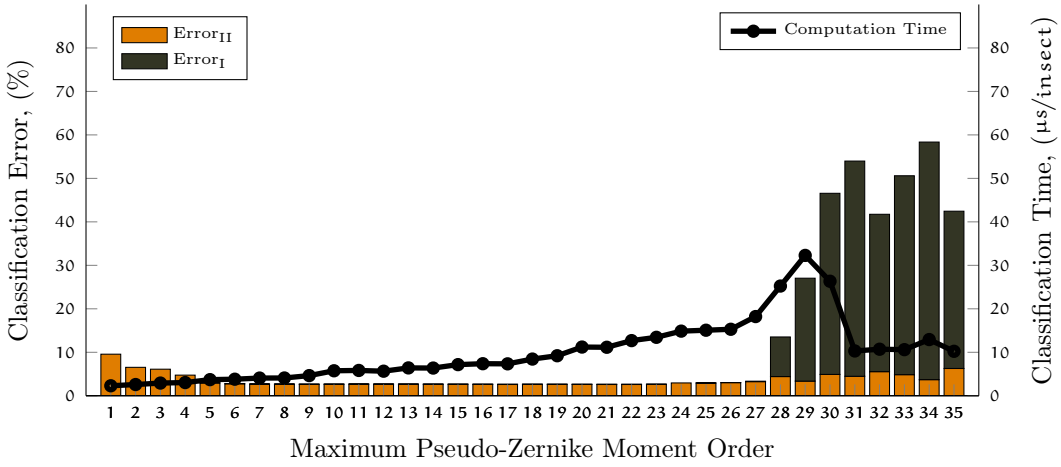


Figure 4.26: pZM classification performance for increasing moment order. $\mathsf{Error}_{\mathrm{I}}$ and $\mathsf{Error}_{\mathrm{II}}$ denote Type-I and Type-II error, respectively. Error rates are stacked, the top of each stacked bar representing the total error rate.

and a drop in computation time for completing a classification due to the reduced dimensionality of the resulting classifier model.

4.3.1.3 *Fourier-Mellin Moment Performance*

Fourier-Mellin Moments (FMM) classification performance generally improves as the maximum FMM order increases from order $p_{max} = 1$ to $p_{max} = 26$. The maximum accuracy for FMM features of 95.9% occurs at order $p = 26$, with error rates of $\mathsf{Error}_{\mathrm{I}} = 0.65\%$ and $\mathsf{Error}_{\mathrm{II}} = 3.42\%$. Figure 4.27 shows the $\mathsf{Error}_{\mathrm{I}}$ and $\mathsf{Error}_{\mathrm{II}}$ rates and the time required to perform one classification for varied levels of p_{max} from 1 to 35. Classification time increases in an exponential manner as p_{max} increases, which is as expected since the number of features in the feature vector for FMM also increases exponentially with p_{max}.

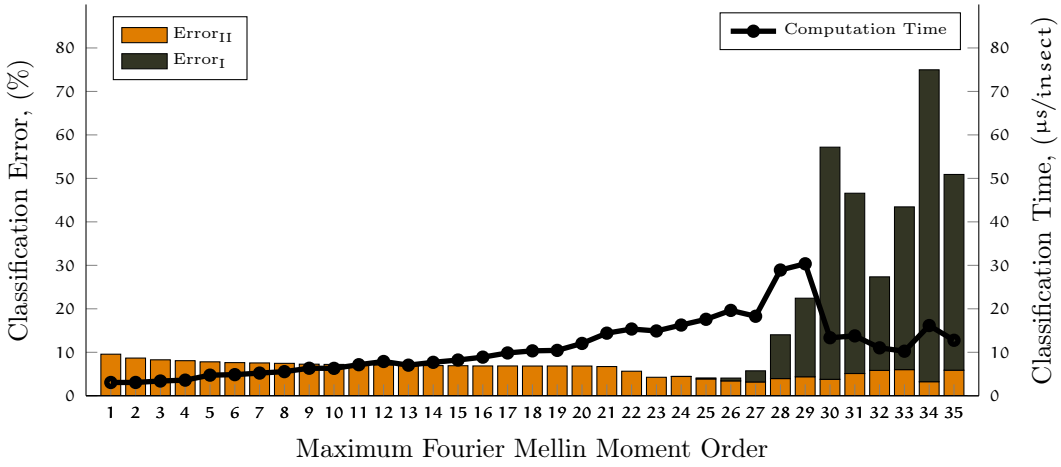


Figure 4.27: FMM classification performance for increasing moment order. $\mathsf{Error}_{\mathrm{I}}$ and $\mathsf{Error}_{\mathrm{II}}$ denote Type-I and Type-II error, respectively. Error rates are stacked, with the top of each stacked bar representing the total error rate.

For $p_{max} > 27$, the orthogonality of the FMM begins to break down due to decimation and the scale of images, creating a degenerate training matrix. This leads to a sudden increase in $\mathsf{Error}_{\mathrm{I}}$ rate, which were minimal for $p_{max} \leqslant 27$.

4.3.1.4 *ART Performance*

MPEG-7 Angular Radial Transform (ART) classification performance is nearly identical for all orders from $p_{max} = 1..10$ when using the SVM classifier. The maximum accuracy for ART features at order $p = 10$ is $\mathsf{Acc} = 91.0\%$, with error rates $\mathsf{Error}_{\mathrm{I}} = 0\%$ and $\mathsf{Error}_{\mathrm{II}} = 9.0\%$. Figure 4.28 shows the $\mathsf{Error}_{\mathrm{I}}$ and $\mathsf{Error}_{\mathrm{II}}$ rates and the time required to perform one classification for varied levels of p_{max} from 1 to 60. Classification time increases in a linear manner as p_{max} increases, which is as expected since the number of repetitions of ART is equal for every p_{max}.

ART features are artificially limited to $p_{max} = 10$ with 11 repetitions per order since there is no orthogonality of features and it is intended to be a encoded into

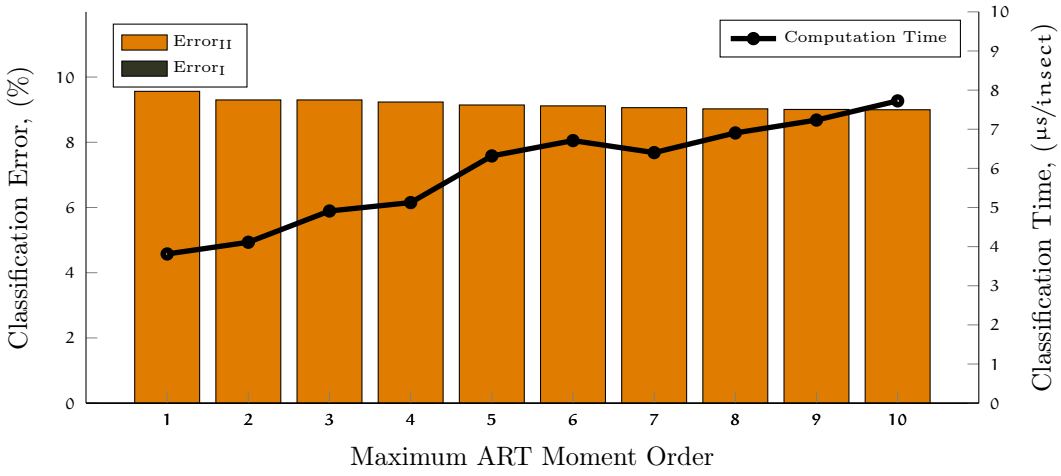


Figure 4.28: ART classification performance for increasing moment order. $\texttt{Error}_{\text{I}}$ and $\texttt{Error}_{\text{II}}$ denote Type-I and Type-II error, respectively. Error rates are stacked, with the top of each stacked bar representing the total error rate. Note: Although this figure shows the same information as in Figure 4.25 through Figure 4.27, both the left and right side axis scales have been changed such that the small trends are visible.

a fixed number of digits in MPEG images. These limits are such that ART feature vectors never grow large enough to induce a degenerate training matrix as is seen in the other three moment classes. When compared to the other classes, however, ART features prove much less capable of identifying pecan weevils, at least when using only ART features with SVM classifiers.

4.3.2 *Combined Feature Performance*

Combined feature vectors were created by concatenating feature vectors from individual moment types up to the order that produced the best classification performance as individual features. Table 4.11 shows the number of features in each combined feature set, the rates of $\texttt{Error}_{\text{I}}$ and $\texttt{Error}_{\text{II}}$, Accuracy, MCC score, time to train the classifier with 730 input vectors, and the time to classify an individual

feature vector. As individual feature types, ZM shows the performance at 98.0% accuracy with an MCC score of 0.877, followed closely by pZM and FMM. In combinations of two types of moments, the best performance comes with ZM combined with pZM, slightly improving the performance of ZM alone from $\mathsf{MCC} = 0.877$ to $\mathsf{MCC} = 0.888$. Surprisingly, combinations including ART improve upon the individual performance for the three other moment types despite the poor performance of ART on its own. The best overall performance for combined features is provided by the combination of ZM, pZM, and ART, with an accuracy of 98.3% and acMCC score of 0.900. The result is due to the significant reduction in $\mathsf{Error}_{\mathrm{II}}$ rate when compared to that of the individual ZM, pZM, and ART features, but comes at the cost of a small increase in $\mathsf{Error}_{\mathrm{I}}$. Training time is relatively constant in all combinations that do not include FMM. For reasons unknown, the inclusion of FMM features results in a four-order of magnitude increase in training time across all combinations. Classification time increases linearly with the number of features in the feature set.

4.3.3 *Comparison of Classifiers*

Two types of classifiers were used to identify insect shapes, including Support Vector Machines (SVMs) and Naive Bayes (NB). SVM is limited to binary classification problems, while NB can be used either as a binary or multi-class classifier. In this research, NB is used both in the binary and the multi-class forms. In the case of multi-class NB classification, accuracy is reported based on the accuracy of classification in classes, while the $\mathsf{Error}_{\mathrm{I}}$, $\mathsf{Error}_{\mathrm{II}}$, and MCC score are reported in the binary sense as there are not suitable multi-class analogs for these values. All results presented prior to this section have been based on SVM classification results.

Table 4.11: Performance metrics of Combined Moment Features with SVM classification

Feature	Feature Count	Error_{I} %	Error_{II} %	Acc %	MCC	Feature Computation Time ms	Classification Time µs
ZM, $p_{max} = 26$	196	0.07	1.97	98.0	0.877	84.1	8.87
pZM, $p_{max} = 21$	253	0.06	2.59	97.4	0.837	108.5	10.6
FMM, $p_{max} = 26$	378	0.66	3.40	95.9	0.745	162.2	15.6
ART, $p_{max} = 10$	110	0	8.90	91.1	0.226	**47.2**	**6.94**
ZM+pZM	443	0.17	1.69	98.1	0.888	190.0	22.3
ZM+FMM	573	0.27	2.40	97.3	0.836	245.8	31.7
ZM+ART	306	0.12	1.75	98.1	0.887	131.3	13.3
pZM+FMM	609	0.36	2.95	96.7	0.795	261.3	26.1
pZM+ART	363	0.08	2.09	97.8	0.869	155.7	16.6
FMM+ART	466	0.58	3.09	96.3	0.771	199.9	23.9
ZM+pZM+FMM	799	0.27	2.26	97.5	0.846	342.8	46.0
ZM+pZM+ART	553	**0.17**	**1.49**	**98.3**	**0.900**	237.2	30.79
ZM+FMM+ART	683	0.31	2.30	97.4	0.841	293.0	39.8
pZM+FMM+ART	719	0.31	2.59	97.1	0.822	308.5	40.7
ZM+pZM+FMM+ART	909	0.28	2.13	97.6	0.853	390.0	50.1

Note: Error_{I}, Error_{II}, Acc, and MCC represent Type-I error, Type-II error, classification accuracy, and Mathews Correlation Coefficient, respectively.

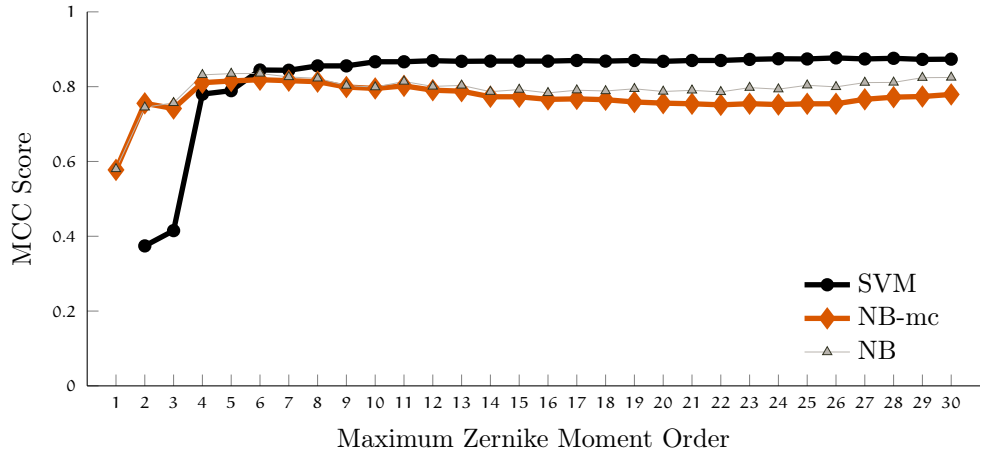


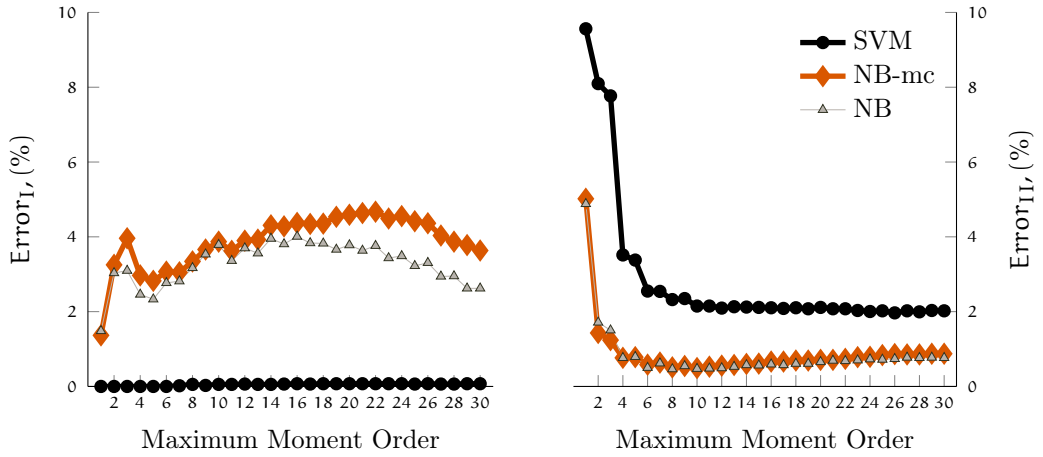
Figure 4.29: Comparison of classifier performance measured in MCC score for ZM. NB-mc denotes multi-class NB classifier, while NB in the legend denotes the binary case. The MCC score is undefined (0/0) for $p = 1$ with SVM.

4.3.3.1 *Classifier performance with ZM features*

Figure 4.29 shows the MCC scores for ZM of order $p = 1..30$. SVM performs better for all $p \geqslant 6$, achieving good classification results where $MCC > 0.844$ for maximum order $6 < p \leqslant 30$. NB classifiers, both binary and multi-class, still perform adequately with $MCC \approx 0.8$ for all orders $p \geqslant 3$. An interesting result is that, while SVM performs better than NB in terms of overall performance, the $Error_{II}$ rates are lower in all cases using NB classification. This means that, while SVM makes a better classification overall, it is more likely to predict that a true pecan weevil is not a pecan weevil than the NB classifier. Figure 4.30 shows the $Error_{I}$ and $Error_{II}$ rates for the three classifiers at varying moment order.

4.3.3.2 *Classifier performance with pZM features*

Figure 4.31 shows the MCC scores for the three classifier types using pZM features. Much the same as with ZM features, the SVM classifier outperforms the two NB classifiers for moment orders $p \geqslant 5$, although all three perform well enough to be



(a) Classifier $\mathsf{Error}_{\mathrm{I}}$ rates for varying ZM order

(b) Classifier $\mathsf{Error}_{\mathrm{II}}$ rates for varying ZM order

Figure 4.30: Comparison of classifier error rates for ZM for varied moment order with SVM, binary NB, and multi-class NB classifiers.

considered for further work. ForSVM the MCC score is consistently above 0.8, while it remains greater than 0.75 out to order $p = 25$. Figure 4.32 shows the $\mathsf{Error}_{\mathrm{I}}$ and $\mathsf{Error}_{\mathrm{II}}$ rates for the three classifiers at varying moment order. As with ZM, the NB classifiers provide fewer Type-II misclassifications with greater rates of $\mathsf{Error}_{\mathrm{I}}$ while the SVM classifier has virtually no Type-I error with greater rates of $\mathsf{Error}_{\mathrm{II}}$.

4.3.3.3 *Classifier performance with FMM features*

Figure 4.33 shows the MCC scores for the three classifiers using FMM features. In a reversal of performance, the NB features perform better for all moment orders except for the maximum, $p = 25$, with SVM performing much worse than with ZM or pZM. Performance reaches a maximum MCC score of 0.833 at order $p = 22$ for NB with multi-class classification and 0.842 for binary classification, which is significantly better than the maximum performance of $\mathsf{MCC} = 0.745$ with SVM. Figure 4.34 shows the $\mathsf{Error}_{\mathrm{I}}$ and $\mathsf{Error}_{\mathrm{II}}$ rates for the three classifiers at varying

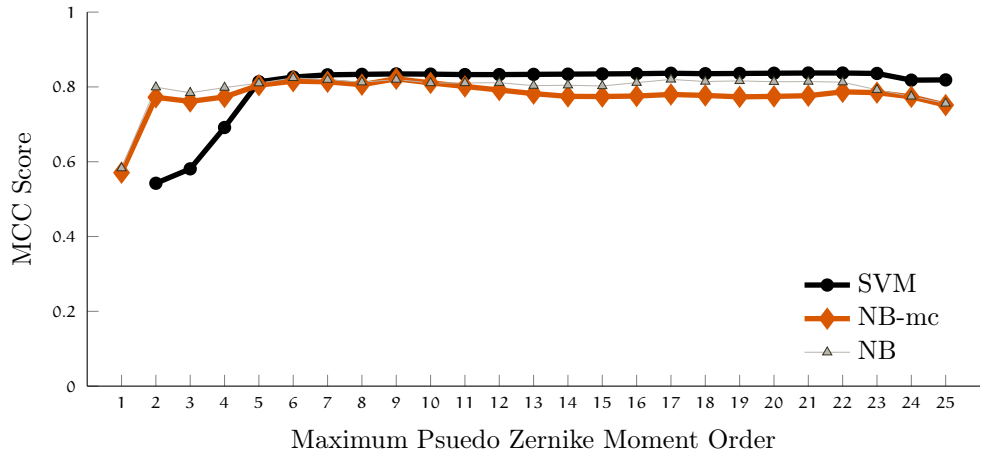
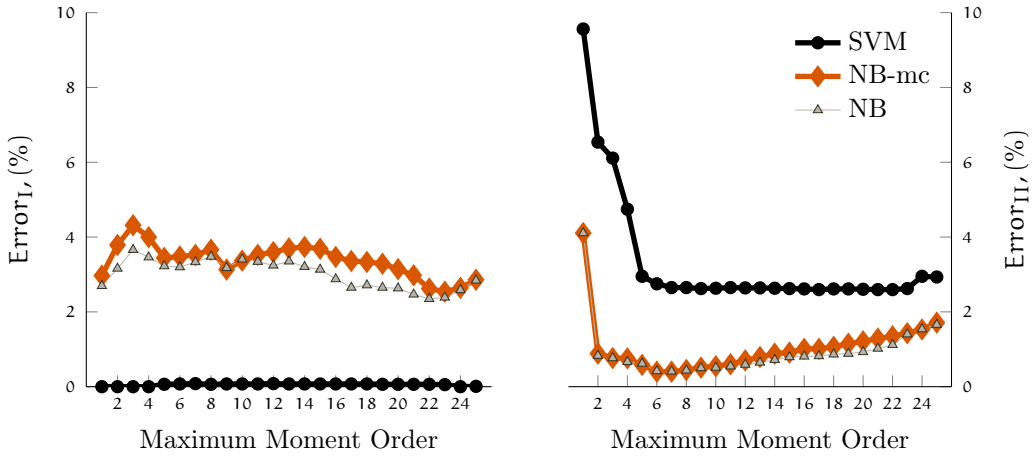


Figure 4.31: Comparison of classifier performance for pZM. NB-mc denotes the multi-class NB case, while NB in the legend denotes the binary case. The MCC score is undefined (0/0) for $p = 1$ with SVM.



(a) Classifier $\texttt{Error}_{\text{I}}$ rates for varying pZM order

(b) Classifier $\texttt{Error}_{\text{II}}$ rates for varying pZM order

Figure 4.32: Comparison of classifier error rates for pZM for varied moment order with SVM, binary NB, and multi-class NB classifiers

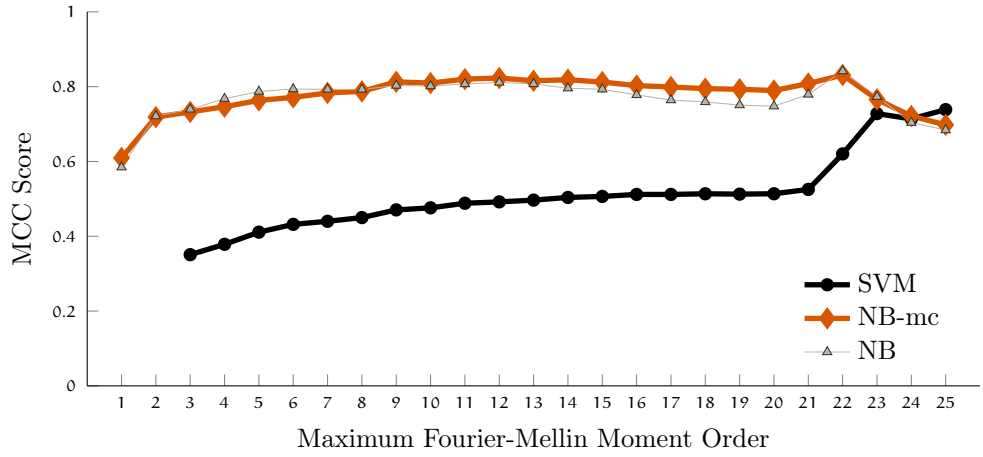


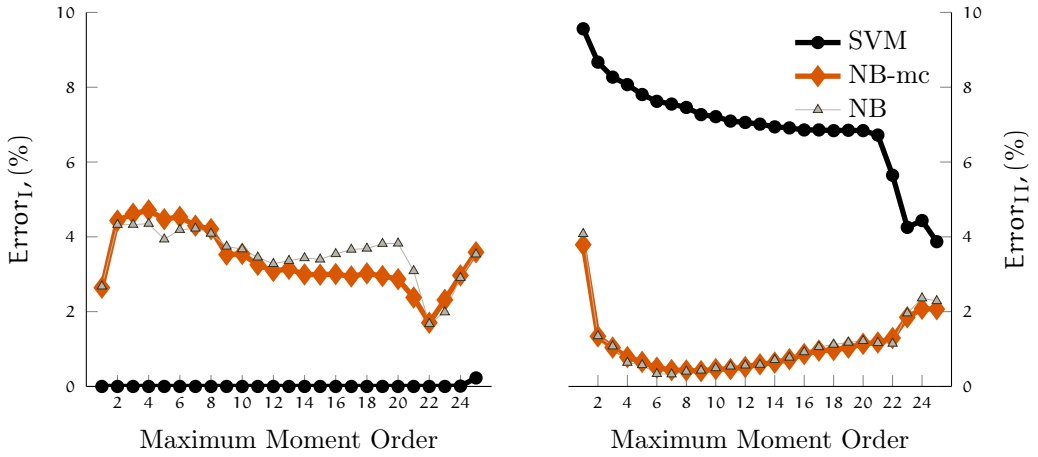
Figure 4.33: Comparison of classifier performance for ZM. NB-mc denotes the multi-class NB case, while NB in the legend denotes the binary case. The MCC score is undefined (0/0) for $p = 1..2$ with SVM.

moment order for FMM features. When compared with the same graphs for ZM and pZM, the major difference is that the Type-II error rates are twice as great when using FMM feature and the SVM classifier, while the Type-I error rates remain similar for all classifier types across the moment types.

4.3.3.4 *Classifier performance with ART features*

In the case of ART features, classification performance is significantly better with NB, and nearly on par with the maximum classification performance of combined feature sets using SVM classifiers[1]. Figure 4.35 shows the MCC scores for the three classifiers using ART features. MCC scores for ART features remain above 0.835 and 0.846 for NB classifiers using multi-class and binary classification, respectively, while failing to surpass $MCC = 0.226$ using the SVM classifier. The performance of NB with ART features is roughly as good as that of any of the combined feature sets using SVM. Interestingly, the strong classification performance begins at order $p = 1$ and continues to $p = 10$, with little change across the range.

1 See Table 4.11



(a) Classifier $\texttt{Error}_{\text{I}}$ rates for varying FMM order

(b) Classifier $\texttt{Error}_{\text{II}}$ rates for varying FMM order

Figure 4.34: Comparison of classifier error rates for FMM for varied moment order with SVM, binary NB, and multi-class NB classifiers

4.3.3.5 *Classifier performance with Combined features*

Table 4.12 and Table 4.13 show the performance of NB classifiers for binary and multi-class classification, respectively. For both cases, NB has higher $\texttt{Error}_{\text{I}}$ rates with lower $\texttt{Error}_{\text{II}}$ rates. Ultimate performance in terms of the MCC score is generally better for the binary classification even though the MCC score is computed as a binary performance metric for both types. Overall accuracy is much improved for the binary classifier since the between-class misclassifications of non-pecan weevil images count against the accuracy rate.

Although SVM generally performs better than NB for insect identification, the reduced rates of Type-II error could allow NB and SVM to be potentially used in tandem. This would mean using SVM classification in general, but checking the corresponding binary NB prediction in the case that the SVM predicts a non-pecan weevil. If both classifiers agree, the classification of non-pecan weevil would be

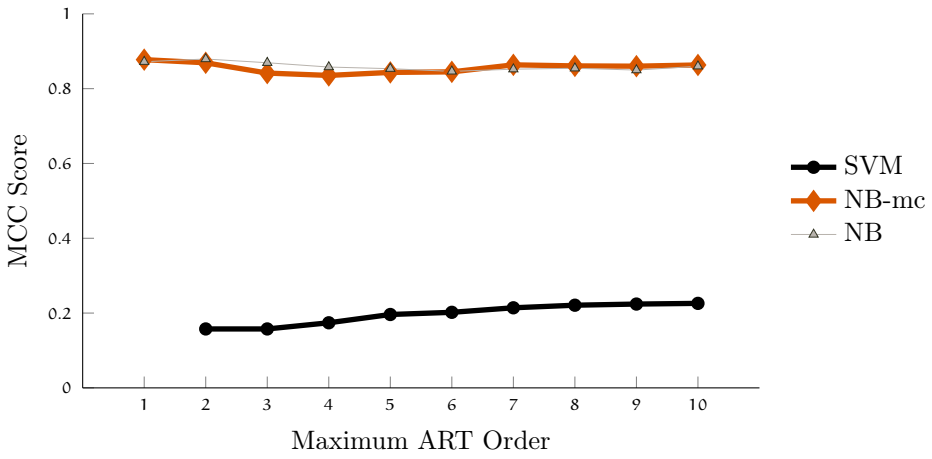


Figure 4.35: Comparison of classifier performance for ZM. NB-mc denotes the multi-class NB case, while NB in the legend denotes the binary case. The MCC score is undefined (0/0) for $p = 1$ with SVM.

selected. ART features, while providing the worst-case classification with SVM, perform well when the metric is Type-II error.

4.3.4 *Reduced Feature Performance*

SVM and NB classifiers generally work well for feature vectors with many members as long as the number of training samples used to train the classifier is much greater than the number of features. In the case of combined feature vector sets using all four types of moments, there are 909 unique features, which is a large number compared to many recognition problems. Since many of these features are orthogonal, the risk of overfitting a model is reduced, but there is still some concern about the *Curse of Dimensionality*[2] reducing classification performance due to the large feature set. Additionally, computing moment features is relatively

2 *The curse of dimensionality is the phenomena that causes models to break down when the number of variables is large in relation to the number of data samples. Many orthogonal features are generally good, but more is not always better.*

Table 4.12: Performance metrics of Combined Moment Features with binary NB classification

Feature	Feature Count	Error_{I} %	Error_{II} %	Acc %	MCC	Feature Computation Time ms	Classification Time µs
ZM, $p_{max} = 26$	196	3.31	0.74	96.0	0.799	84.1	7.5
pZM, $p_{max} = 21$	253	2.47	1.02	96.5	0.814	108.5	9.1
FMM, $p_{max} = 26$	378	3.85	2.57	93.6	0.652	162.2	13.0
ART, $p_{max} = 10$	110	**1.98**	0.65	**97.4**	**0.860**	**47.2**	**4.2**
ZM+pZM	443	3.03	0.72	96.2	0.811	190.0	15.3
ZM+FMM	573	3.56	1.51	94.9	0.738	245.8	20.2
ZM+ART	306	2.61	**0.50**	96.9	0.842	131.3	11
pZM+FMM	609	3.18	1.84	95.0	0.730	261.3	21.3
pZM+ART	363	2.17	0.64	97.2	0.852	155.7	13.1
FMM+ART	466	3.31	1.66	95.0	0.737	199.9	16.7
ZM+pZM+FMM	799	2.99	1.46	95.6	0.764	342.8	28.1
ZM+pZM+ART	553	2.67	0.62	96.7	0.833	237.2	19.5
ZM+FMM+ART	683	2.84	1.17	96.0	0.789	293.0	23.7
pZM+FMM+ART	719	2.60	1.29	96.1	0.790	308.5	24.6
ZM+pZM+FMM+ART	909	2.62	1.19	96.2	0.796	390.0	31.7

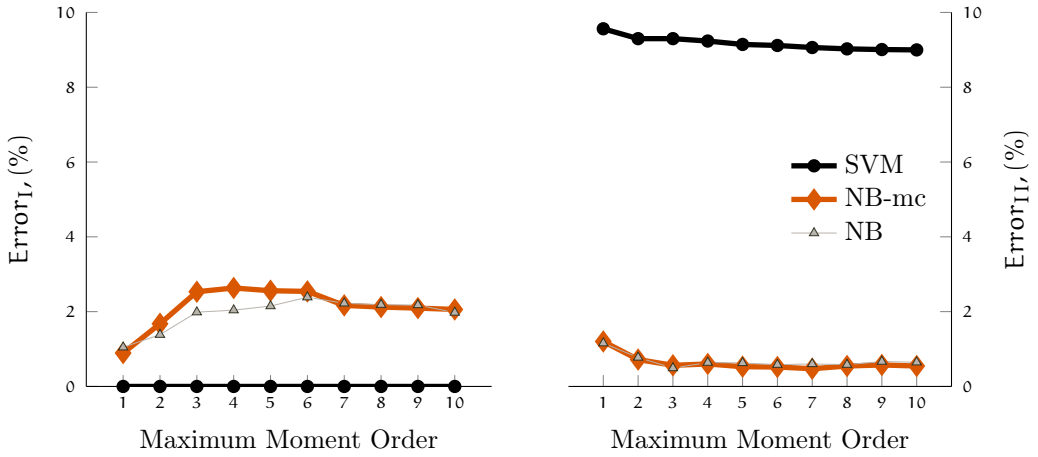
Note: Error_{I}, Error_{II}, Acc, and MCC represent Type-I error, Type-II error, classification accuracy, and Mathews Correlation Coefficient, respectively.

Table 4.13: Performance metrics of Combined Moment Features with multi-class NB classification

Feature	Feature Count	Error_{I} %	Error_{II} %	Acc %	MCC	Feature Computation Time ms	Classification Time µs
ZM, $p_{max} = 26$	196	4.36	0.87	93.0	0.754	84.1	21.0
pZM, $p_{max} = 21$	253	2.98	1.28	94.2	0.776	108.5	25.1
FMM, $p_{max} = 26$	378	3.92	2.31	92.6	0.668	162.2	37.5
ART, $p_{max} = 10$	110	**2.06**	**0.55**	**95.3**	**0.863**	**47.2**	**11.2**
ZM+pZM	443	3.57	0.98	93.7	0.774	190.0	43.8
ZM+FMM	573	3.76	1.41	93.2	0.737	245.8	56.9
ZM+ART	306	3.24	0.64	94.2	0.809	131.3	30.4
pZM+FMM	609	3.51	1.80	93.4	0.719	261.3	60.8
pZM+ART	363	2.66	0.77	94.6	0.824	155.7	36.4
FMM+ART	466	3.46	1.64	93.4	0.733	199.9	48.6
ZM+pZM+FMM	799	3.42	1.37	93.7	0.753	342.8	81.5
ZM+pZM+ART	553	3.21	0.77	94.1	0.801	237.2	56.4
ZM+FMM+ART	683	3.35	1.18	93.8	0.768	293.0	70.0
pZM+FMM+ART	719	3.16	1.38	93.9	0.761	308.5	73.1
ZM+pZM+FMM+ART	909	3.20	1.17	94.0	0.775	390.0	92.4

Note 1: Accuracy reported in this table is based on the multi-class classification accuracy, where false classification between non-pecan weevil classes is included in the rate. Error_{I}, Error_{II}, and MCC score are presented in the binary sense where false classification is only counted against the score when a pecan weevil is classified as non-pecan weevil, or vice versa.

Note 2: Error_{I}, Error_{II}, Acc, and MCC represent Type-I error, Type-II error, classification accuracy, and Mathews Correlation Coefficient, respectively.



(a) Classifier $\texttt{Error}_{\text{I}}$ rates for varying ART order

(b) Classifier $\texttt{Error}_{\text{II}}$ rates for varying ART order

Figure 4.36: Comparison of classifier error rates for ART for varied moment order with SVM, binary NB, and multi-class NB classifiers

computationally expensive, taking as much or more time as actual training of the classifier of performing classifications based on the features.

In this section, two methods for reducing the feature vector length are presented. Both methods are based on dimensionality reduction techniques commonly used in pattern recognition to project the feature space into a smaller number of dimensions. Rather than projecting the features to a new space in a manner that uses all of the original features, these methods select individual variables based on their contribution to describing either the variance of the training set or the between-class variance of the feature set. Any reduction in the number of features will be beneficial in the case that there is minimal impact on the classification performance as the reduced feature set will require fewer features to be computed, reduced computation for classification, and reduced memory requirements for either storage or data transfer.

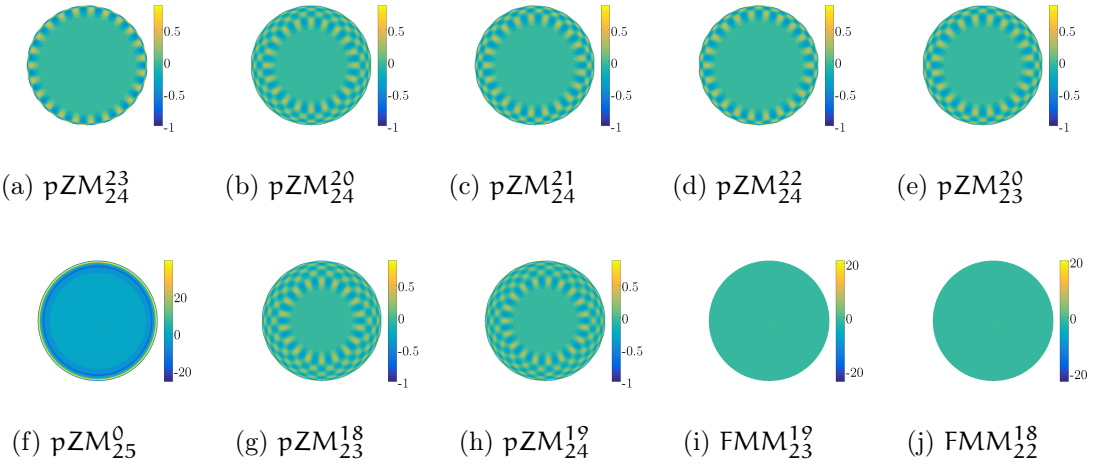
4.3.4.1 *Results of PCA–based Feature Reduction*

The Principle Component Analysis (PCA)–based method for feature reduction developed for this research begins with the largest number of possible features, computes the scatter matrix of the training samples, solves the generalized eigenvalue problem to obtain the eigenvectors and eigenvalues of the scatter matrix, then selects the best dimension from the overall feature set as the feature with the largest eigenvector projection for the largest eigenvalue. The selection process is iterated on the remaining, un-selected features until all features have been selected, thus creating an ordered list of features in the best PCA direction sense. This direction is the feature that accounts for the majority of the variance in the samples, regardless of if that variance is noise or if it is useful to discern between classes.

The first feature selected by the PCA–based method is the pZM of order $p = 25$ and repetition $q = 4$, otherwise written as pZM_{25}^{4}. With this single feature, accuracy rates of 93.9%, 94.2%, and 92.2% are achieved using SVM, binary NB , and multi-class NB, respectively. The corresponding MCC scores were 0.581, 0.623, and 0.648. Figure 4.37 shows the real portion of the basis functions for the first ten features selected by the PCA–based feature selection algorithm.

Upon examination of the first features selected, it is likely that the values of these moments account for mainly noise in the model rather than provide good discernment between classes, which is a known problem in any method that relies on PCA. With that in mind, however, the performance of the single-member feature vector as a classifier shows that at least some of the variance described by pZM_{25}^{4} is inherent to the model classes.

Figure 4.38 shows the performance of the SVM, binary NB , and multi-class NB classifiers in terms of MCC score as features are added to the feature vector based

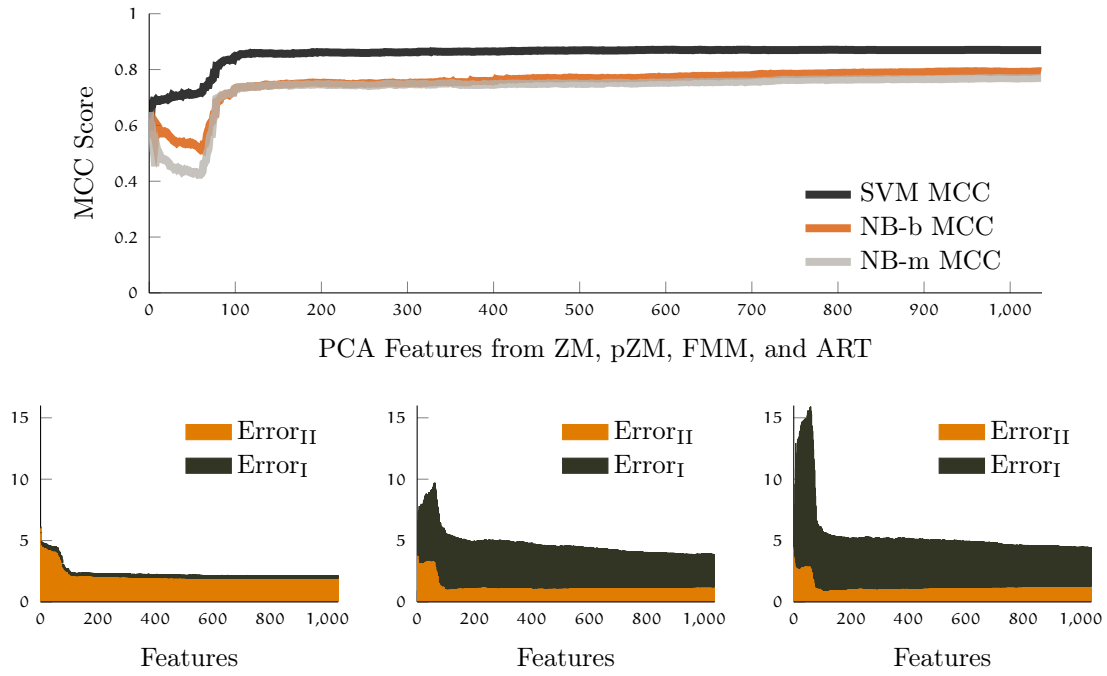


(a) pZM_{24}^{23} (b) pZM_{24}^{20} (c) pZM_{24}^{21} (d) pZM_{24}^{22} (e) pZM_{23}^{20}

(f) pZM_{25}^{0} (g) pZM_{23}^{18} (h) pZM_{24}^{19} (i) FMM_{23}^{19} (j) FMM_{22}^{18}

Figure 4.37: Basis functions for the ten best PCA features. These represent the ten best moment basis functions to explain the variance in the full $1,829$ sample data set of insect shapes.

on the best PCA direction. The classification performance declines as PCA features two through ten are added to the feature vector despite these features describing more of the variance than those that are selected subsequently. This is evidence that much of the information carried by the second through tenth PCA features is noise when considered as classification features. For SVM classification, performance continues to rise after the tenth feature is added, while both binary and multi-class NB continue to decline in performance until the sixty-first feature is added. Peak classification performance occurs with 584 features with the SVM classifier, providing an accuracy of 97.9% and corresponding MCC score of 0.871. In the case that further reduction of feature vector length is desired for computational or storage reasons, 99% of this maximum performance can be reached with 274 features (MCC $= 0.863$, 97.7% accurate), while 95% of the maximum can be reached with only 86 features (MCC $= 0.831$, 97.25% accurate). Complete ordering $1,036$ of features using the PCA method takes 237 seconds.



(a) SVM Classifier (b) Binary NB Classifier (c) Multi-class NB Classifier

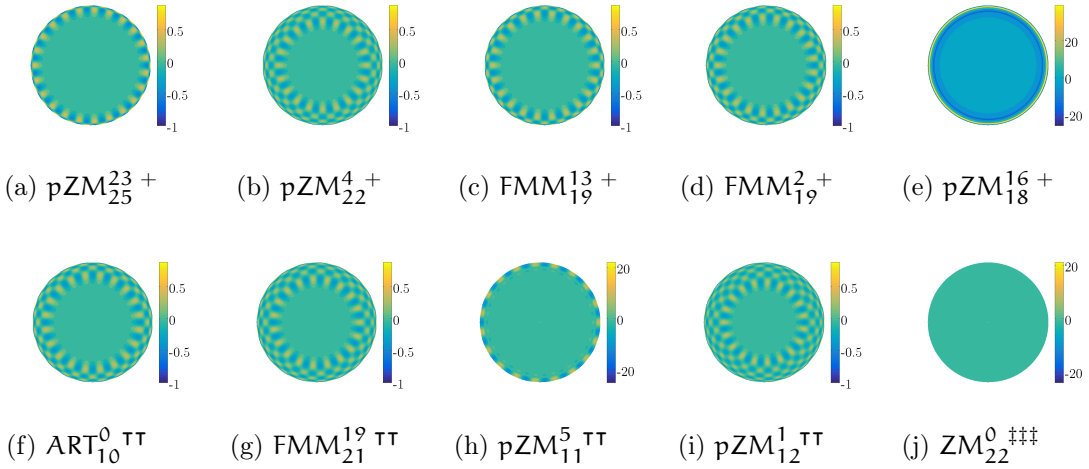
Figure 4.38: Performance results based on PCA dimension reduction. The top figure shows the MCC scores for SVM, binary NB, and multi-class NB. Sub-figures a, b, and c show the Type-I and Type-II errors in percent for the three classifiers as the number of features increases. Error rates are stacked, with the top of the stack representing the total error rate.

4.3.4.2 *Results of FMDA–based Feature Reduction*

The FMDA–based method is very similar to the PCA–based method for dimensional reduction, but it takes into account both the scatter matrix of the entire data set and the scatter matrices from for each individual class, seeking to maximize the ratio of the between class and within class scatter. For each iteration, this method selects up to one fewer than the total number of classes as opposed to a single feature as is done in the PCA–based method. In the proposed algorithm, the features that are selected with each iteration are the first $c-1$ features that have the maximum projection in the eigenvectors correspondting to the $c-1$ greatest eigenvalues. If there are repeated features among the $c-1$ largest eigenvalues, the repeated feature is only represented once in the ordered list of selected features. FMDA–based feature reduction can be done either in a multi-class sense where $c > 2$, or in a binary sense where $c \equiv 2$. In the binary sense, one feature is selected per iteration as in the PCA–based method, but the between class scatter is considered in the selection rather than solely relying on the scatter matrix of the entire data set.

Multi-Class FMDA–based Feature Reduction

The first iteration of the multi-class FMDA–based method for feature reduction selects the features pZM_{25}^{23}, pZM_{22}^{4}, FMM_{19}^{13}, FMM_{19}^{2}, and pZM_{18}^{16}, all of which are unique from any of the first ten features selected by PCA. With these five features, accuracy rates of 95.2%, 93.0%, and 90.0% are achieved using SVM, binary NB , and multi-class NB, respectively. The corresponding MCC scores were 0.688, 0.587, and 0.556, representing both improvement and decline when compared to the first feature selected in PCA. When compared to the first five features selected by PCA, the performance of multi-class FMDA–selected features is slightly better,



(a) $\mathrm{pZM}_{25}^{23\,+}$ (b) $\mathrm{pZM}_{22}^{4\,+}$ (c) $\mathrm{FMM}_{19}^{13\,+}$ (d) $\mathrm{FMM}_{19}^{2\,+}$ (e) $\mathrm{pZM}_{18}^{16\,+}$

(f) $\mathrm{ART}_{10}^{0\,\tau\tau}$ (g) $\mathrm{FMM}_{21}^{19\,\tau\tau}$ (h) $\mathrm{pZM}_{11}^{5\,\tau\tau}$ (i) $\mathrm{pZM}_{12}^{1\,\tau\tau}$ (j) $\mathrm{ZM}_{22}^{0\,\ddagger\ddagger\ddagger}$

Figure 4.39: Basis functions for the ten best multi-class FMDA features. These represent the ten best moment basis functions to maximize the between class scatter versus the within class scatter in the full 1,829 sample data set of insect shapes. Symbol $^{+}$ denotes first iteration selection, $^{\tau\tau}$ denotes second iteration selection, and ‡‡‡ denotes third iteration selection.

with improvements in MCC score of 0.11%, 3.33%, and 8.31% for SVM, multi-class NB , and NB NB classification, respectively. The real portion of the basis functions for the first ten features selected by multi-class FMDA are shown in Figure 4.39.

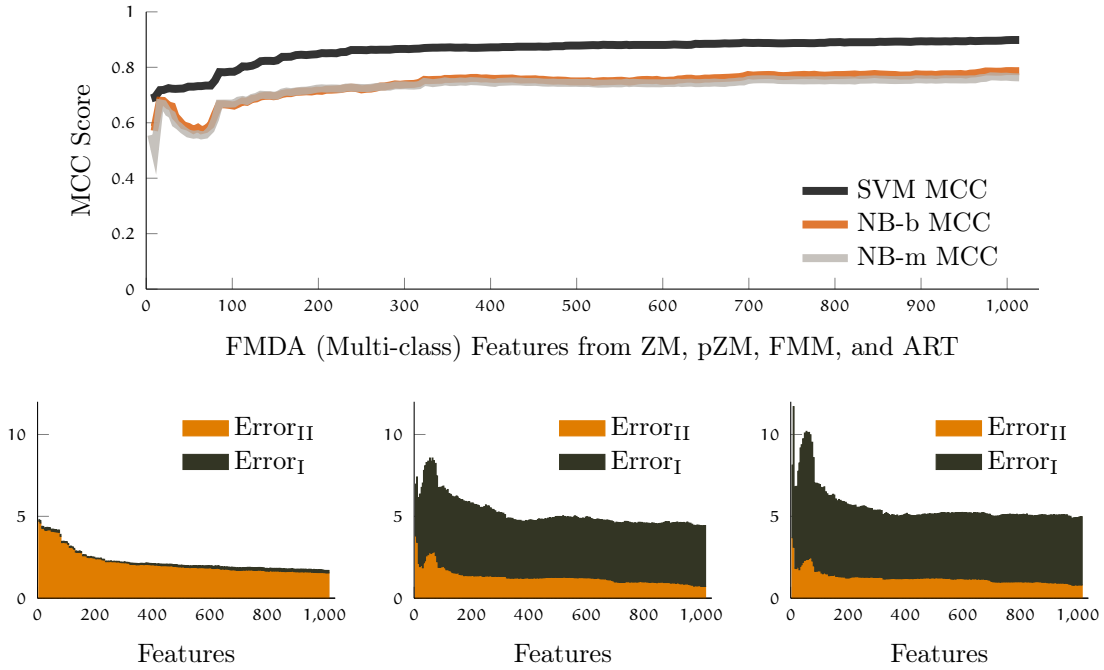
Figure 4.40 shows the performance of the SVM, binary NB , and multi-class NB classifiers in terms of MCC score as features are added to the feature vector based on the best multi-class FMDA direction. Although not noticeable in the figure, there are far fewer data points in each of the graphs for multi-class FMDA than in the case of PCA–based selection. Each of the iterations of multi-class FMDA selects between one and six features as there are seven identified classes in the data set, and each iteration is represented by one point in the figures. This results in only 190 data points compared to 1,036 in the case of PCA or binary FMDA.

Multi-class FMDA–based feature selection results in a more smoothly monotonic increase in performance as features are added to the feature vector than does Prin-

ciple Component Analysis (PCA)–based selection. This comes at a reduced relative increase rate in classification performance such that the maximum performance in terms of MCC score does not occur until 186 out of 190 selections whereas it occurs at 584 iterative selections out of a possible 1,036 for the PCA–based method. The maximum performance in the case of multi-class FMDA feature selection includes 1,014 out of 1,036 possible unique features. This maximum classification performance does improve on that of the PCA–based though, with an increase in MCC score from 0.899 compared to 0.871 for PCA. Performance of the NB classifiers is worse than that that of SVM, with maximum MCC scores of 0.786 and 0.768 for binary and multi-class NB, respectively. Complete ordering 1,036 of features using the multi-class FMDA method takes 62 seconds.

Using the features selected by FMDA with the SVM classifier, the number of features can be reduced from 1,014 to 794 while still achieving 99% of the maximum performance with an accuracy rate of 98.3% and an MCC score of 0.891. For 95% of the maximum performance, the feature set can be reduced further to 226 dimensions with accuracy of 97.6% and MCC score of 0.855. Ninety percent of maximum performance requires 128 features and achieves accuracy of 97.0% and MCC score of 0.812.

The application of multi-class FMDA–based for this data set suffers due to the small number of samples available in the non-pecan weevil classes, where there are only five available examples of spiders or moths to be used between the training and testing data sets. This small number of samples in some classes leads to the within-class scatter matrix for the non-weevil classes to be ill conditioned, resulting in imprecise solutions to the generalized eigenvalue problem. Additional samples from these would be expected to improve performance, not only of classification,



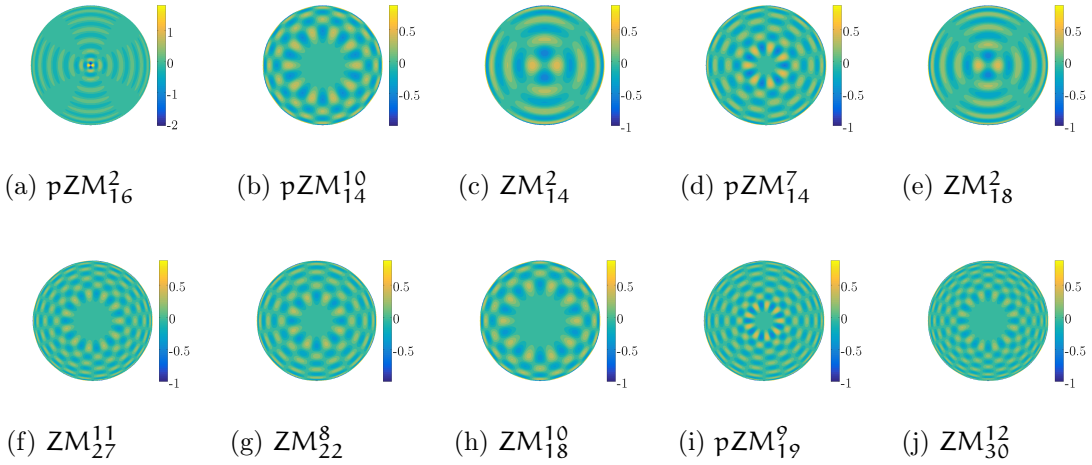
(a) SVM Classifier (b) Binary NB Classifier (c) Multi-class NB Classifier

Figure 4.40: Performance results based on multi-class FMDA dimension reduction. The top figure shows the MCC scores for SVM, binary NB, and multi-class NB. Sub-figures a, b, and c show the Type-I and Type-II errors in percent for the three classifiers as the number of features increases. Error rates are stacked, with the top of the stack representing the total error rate.

but of feature selection for reduced dimensionality as the likelihood of a singular scatter matrix would decline for larger numbers of samples in the matrix.

Results of Binary FMDA–based Feature Reduction

Binary FMDA–based feature reduction uses the same algorithm as the multi-class case, but pools all non-pecan weevil samples into a single class. This results in a binary, two class case for feature selection. Since FMDA projects a data set in the $c - 1$ directions that best separate the classes, the binary form of FMDA–based feature reduction selects exactly one feature per iteration. This is the same as in the case of PCA–based feature selection, but with the distinction of considering the class



(a) pZM_{16}^{2} (b) pZM_{14}^{10} (c) ZM_{14}^{2} (d) pZM_{14}^{7} (e) ZM_{18}^{2}

(f) ZM_{27}^{11} (g) ZM_{22}^{8} (h) ZM_{18}^{10} (i) pZM_{19}^{9} (j) ZM_{30}^{12}

Figure 4.41: Basis functions for the 9th through 18th best multi-class FMDA features. These represent the moment basis functions that maximize the between class scatter versus the within class scatter in the full $1,829$ sample data set of insect shapes. The first eight features can be seen in Figure 4.39.

membership of the samples and the discernment of classes when making feature selections. As such, the binary FMDA–based method takes more computational resources than the multi-class FMDA or PCA based methods.

The first feature selected by the binary FMDA–based method for feature reduction was pZM of order $p = 25$ and repetition $q = 23$, otherwise written as pZM_{25}^{23}, which is the same as that selected by the multi-class method. With this single feature, accuracy rates of 93.9%, 94.2%, and 92.2% are achieved using SVM, binary NB , and multi-class NB, respectively. The corresponding MCC scores were 0.581, 0.623, and 0.648, which are exactly the same as the performance with the first feature selected in PCA. The first ten features selected by binary FMDA are pZM_{25}^{23}, FMM_{19}^{2}, ART_{10}^{0}, pZM_{18}^{16}, pZM_{22}^{4}, FMM_{19}^{13}, pZM_{12}^{1}, ZM_{22}^{0}, pZM_{16}^{2}, and pZM_{14}^{10}. Eight of the first ten features are shown in Figure 4.39, since they were also selected by the multi-class FMDA method. Figure 4.41 shows the real portion of the basis functions for the remaining two features, plus the following eight selected features.

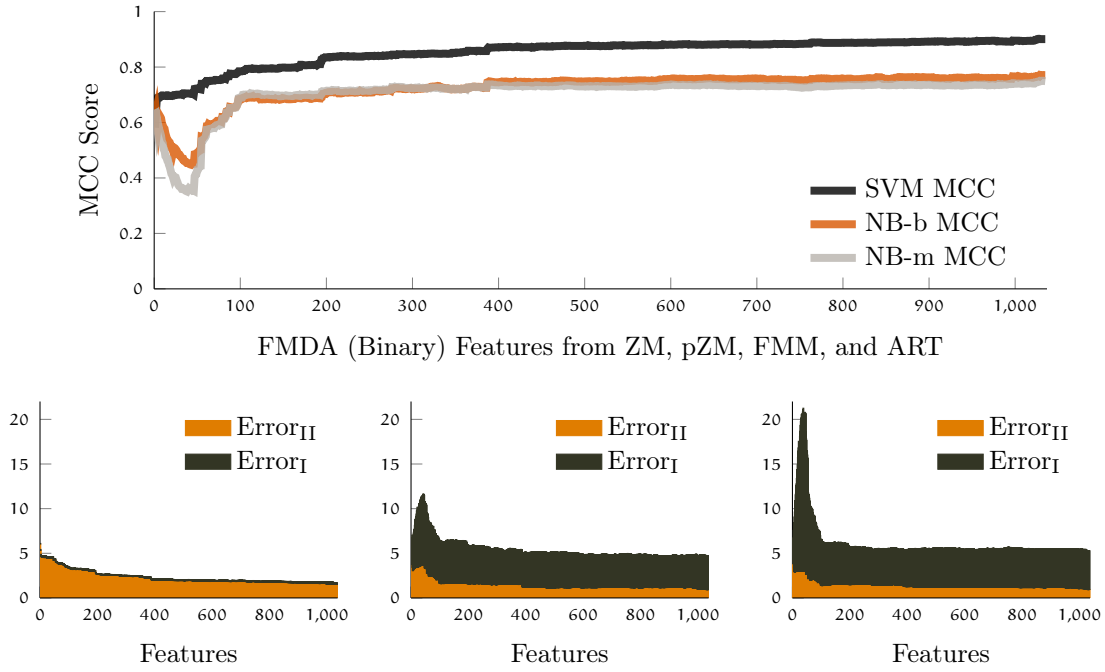
Figure 4.42 shows the performance of the SVM, binary NB , and multi-class NB classifiers in terms of MCC score as features are added to the feature vector based on the best binary FMDA direction. The maximum classification performance occurs with 1,027 of the 1,036 possible features included in the model, with a corresponding accuracy of 98.4% and MCC score of 0.902. If further reduction in the number of features is desired, 99% of maximum performance occurs with 956 features, achieving 98.2% and an MCC score of 0.895. Ninety five percent and ninety percent of maximum performance require 364 and 25 features, respectively, and provide accuracy of 97.7% and 95.40% with MCC scores of 0.859 and 0.825. Complete ordering 1,036 of features using the multi-class FMDA method takes 264 seconds.

4.3.5 *Moment Feature Computation*

The kernel for computing angular-radial moments was tested with four use cases, and compared to the Lans code used in the work of Ashaghathra (2008). The four use cases include:

1. Pre-allocated variables with one image passed to the engine at a time in a for-loop
2. Pre-allocated variables with a stack of images in a single 3-D matrix passed to the engine
3. No pre-allocation of variables with a stack of images in a single 3-D matrix passed to the engine
4. Pre-allocated variables, stacked 3-D matrix of images, running in a parfor parallel loop

The major variables tested are the impacts of pre-allocation of the moment variables versus no pre-allocation, the impacts of single images passed to the kernel versus a



(a) SVM Classifier (b) Binary NB Classifier (c) Binary NB Classifier

Figure 4.42: Performance results based on binary FMDA dimension reduction. The top figure shows the MCC scores for SVM binary NB. Sub-figures a, b, and c show the Type-I and Type-II errors in percent for the SVM, binary NB, and multi-class NB classifiers, respectively, as the number of selected features increases. Error rates are stacked, with the top of the stack representing the total error rate.

stack of images passed as a 3-D matrix, and the impacts of parallel computing for stacks of images. Each of these variables is tested for a varying number of moments to be computed.

Pre-allocation of memory allows the moment engine to compute moments significantly faster than the Lans methods, starting with a single image at a time. This is regardless of whether the images are passed to the function as individual images or in a 3-D stack of images. For the single image moment computation, the pre-allocated engine passed a single image requires 3.33 milliseconds compared to 14.0 milliseconds for the Lans method. When passed as a 3-D stack of images, even the single-image stack results in further time savings, requiring only 1.43 milliseconds. The major difference between the pre-allocated version using a for-loop and the pre-allocated version passed a stack of images is the amount of computational overhead that is required. In the for-loop case, this overhead is incurred for each moment computation. For the 3-D stack version this overhead is only computed once for the entire 3-D stack of images.

For groups of images larger than four, the pre-allocated engine settles to a roughly constant computational time, requiring approximately 0.742 milliseconds for the per-image calculation and 0.429 milliseconds for the stacked image 3-D matrix form. The reference Lans implementation requires 5.70 milliseconds per moment. The result is a 87% reduction in computation time for per-image calculation and a 92.5% reduction in computation time for the stacked image, 3-D matrix version when compared to the Lans method.

The proposed engine takes more computational time when not pre-allocated for groups of images up to eight when compared to the Lans method, and reaches equivalent computational efficiency when compared to the per-image, pre-allocated

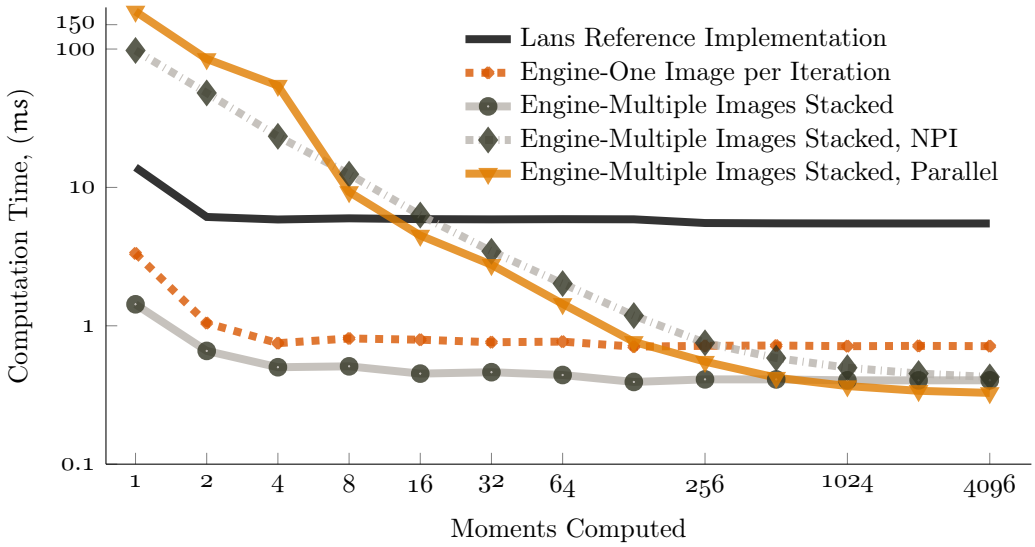


Figure 4.43: Moment Feature Computation Time. The angular-radial moment engine computation time for a single 128×128 moment computation. The engine can be run with or without pre-allocated variables for the pixel rings, angular basis functions, and radial basis functions. The reference implementation is the Lans code base that was used in the work of Ashaghathra (2008).

case for groups of images equal to or greater than 128. Improvements from pre-allocation of variables are finally equalized without using pre-allocation when the number of images grows to $4,096$.

The parallel version of the engine performs worse than any other scenario for groups of image moments up to four, then surpasses the non-parallel, not pre-allocated version for groups of images eight or larger. It is faster than the Lans method for groups of images equal or greater than 16, and is roughly equivilent to the pre-allocated per-image case for groups of 128 images or larger. The parallel method only surpasses the series calculation in efficiency for groups of 1024 images or larger, and only slightly bests the serial version even with the largest of groups. At $4,096$ moment calculations, the parallel version requires 0.327 milliseconds per moment compared to 0.404 milliseconds for the series version. This represents 19.0%

improvement in efficiency at $4,096$ moments, compared to $12,900\%$ more computational time when the number of images is only one.

In sum, the moment computation engine provides much better computational efficiency than the Lans method as long as it is used as intended: with variables pre-allocated and with large, stacked groups of images passed as one 3-D matrix.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCLUSIONS

The research presented in this dissertation was carried out with the goal of enabling automation of pecan weevil emergence monitoring during the late growing season. The major contributions of this study were that it was the first to use an instrumented weevil trap system to monitor weevil emergence with time granularity of minutes rather than days, it was the first to use images of live pecan weevil in their natural posture for identification purposes, and it was the first to include a unified, efficient kernel for feature calculation. The system components developed in this work should be sufficient for the implementation of a network of instrumented traps, including insect detection and identification components.

The research objectives and research hypothesis introduced in Chapter 1 have been investigated thoroughly in this dissertation. Conclusions based on each of the four research objectives and their respective research hypotheses follow in the next sections.

Image histograms for insect presence detection

In evaluation of the efficacy of using image histogram features to detect either the presence or absence of an insect in the trap, it was found that there was enough information in the histograms alone to detect insect presence with an accuracy greater than 92% with KNN classification and cumulative histograms. The optimal number of histogram bins for insect presence detection based on 8-bit images was found to be between nine and twenty, with sub-sampling of images prior to computing histograms resulting in a great improvement in computation speed with little to no impact on classification performance. Adjusting the contrast of images prior to classification was also found to improve classification performance, although at a cost of computational intensity. In summary, the research hypotheses that field collected images contain enough information in their histograms to determine if insects were present was true.

Insect identification with angular-radial moment features

In this research, it was found that any of the angular-radial moments evaluated could be used to successfully discern pecan weevils from other insects that were either imaged in instrumented traps or assumed to potentially be present in pecan growing regions of the country. Used as features individually, correct recognition rates with ZM, pZM, FMM, and ART features and SVM classification were found to be 98%, 97.3%, 95.9%, and 91.0%, respectively. ZM features proved to be the best individual features and required fewer features to be included in the feature vector for maximum performance. Used as combined features, the best classification performance was found to be with ZM, pZM, and ART, with an accuracy rate of 98.3% and

a Matthews Correlation Coefficient (MCC) score of 0.900 using SVM classification. This combined feature set required 553 features for maximum performance.

Support Vector Machine (SVM) classification was found to provide better performance that Naive Bayes (NB) when compared based on MCC scores for nearly all cases, most notably when using ZM or pZM features. For FMM, NB classification was found to provide better performance until the maximum moment order reaches twenty four, while NB was shown to provide better performance with ART features of any order. SVM classification was found to be more prone to Type-II error with virtually no Type-I error when using ZM or pZM features, while NB classification was shown to have less Type-II error than SVM no matter the feature.

PCA and FMDA feature reduction

Two methods for reducing the number of variables in the angular-rotary moment feature vector were developed in this dissertation, with one relying on principles of PCA and the other on FMDA. Both methods were found capable of reducing the number of features from the feature vector with losses in classifier performance that were minimal. The PCA–based method was shown capable of reducing the number of features from the maximum of 1,036 to 86 while still maintaining performance of 97.3% accuracy and 0.831 MCC score, while selecting a single feature that could provide 93.9% accuracy and 0.647 MCC score. The multi-class FMDA–based method selects multiple features in each iteration. The first five selected features alone provided 95.2% accuracy and 0.688 MCC score, while the method was shown to provide 97.0% accuracy and an MCC score of 0.812 using only 128 features. In the binary case of the FMDA method, the first selected feature provided 94.2% accuracy

and 0.648 MCC score, while the method could reduce the number of features to twenty five with 95.4% accuracy and 0.825 MCC score.

The feature reduction methods were also shown useful in removing "bad" features from the feature vector. The binary FMDA–based method provided the maximum classification performance and MCC score of any combination in the study when using 1,027 of the available 1,036 features, with metrics of 98.4% accuracy and 0.902 MCC score. In other words, FMDA based feature selection provided the maximum insect classification performance by discarding the nine "worst" features from the feature set.

Moment Computation Kernel

The proposed moment computation kernel was shown to provide large savings in computation time. For a single moment computation, the engine reduced the computation time by an order of magnitude when compared with the functions used in the work of Ashaghathra (2008). Pre-allocation of variables for the kernel provided a two-order of magnitude improvement in computation speed for the kernel that required over a thousand images to be processed in the un-allocated function before reaching parity in computation time.

Limitations of the current work

Over the course of completing the work presented in this dissertation, working software has been created in the MATLAB environment using both original code and functions from the Image Processing and Statistics toolboxes from the Mathworks corporation. Many of these functions are not readily available for embedded

systems designers, and would need to be implemented in another programming environment in order for embedded systems to utilize the algorithms.

5.2 RECOMMENDATIONS AND FUTURE WORK

In order to deploy a network of instrumented, connected camera traps, modifications to the current system, as well as additional research and development may be warranted. Recommended future work includes:

- Improvements to the instrumented trap design
  - Design with a fixed-mount, fixed-focus camera and curved imaging chamber such that the field of view of the camera captures the same scene in every image. This would improve the ability of the insect detection classifier to detect insects as the noise introduced by moving camera and changes in focus would be removed.
  - Additional image capture triggering is warranted. Whether an optical sensor or some other technology, capturing images only when a sensor is triggered could reduce system power consumption. This is important since the camera is the greatest consumer of power in the trap.
- Additional data for insect detection and identification
  - Collect more images of living non-pecan weevil insects for training and testing. Due to the nature of the orchard where the images used in this study were collected, the number of pecan weevils is very large compared to then number of other insects. This may be true in any orchard since pecan weevil climb into traps out of instinct, but the addition of non-pecan weevil training data could improve classification performance.
  - Collect additional images of the imaging chamber with insects present. Much like the previous bullet, the nature of the data collection method resulted in many more images of an empty chamber than with insects since most of the time there is not an insect entering the trap. Additional data with insects present could improve performance of the presence detection classifiers

BIBLIOGRAPHY

Aggelopoulou, a. D., Bochtis, D., Fountas, S., Swain, K. C., Gemtos, T. a., and Nanos, G. D. (2011). Yield prediction in apple orchards based on image processing. *Precision Agriculture*, 12(3):448–456. (Cited on page 12.)

Agilent Technologies (2008). N9340B handheld RF spectrum analyzer technical overview. Technical report, Agilent Technologies, Santa Clara, California, USA.

Agrawal, D. P. and Zeng, Q.-A. (2010). *Introduction to wireless and mobile systems*. Cengage Learning.

Al-Saqer, S., Weckler, P., Solie, J., Stone, M., and Wayadande, A. (2011). Identification of pecan weevils through image processing. *American Journal of Agricultural and Biological Science*, 6.

Antonio-Javier Garcia-Sanchez, Felipe Garcia-Sanchez, J. G.-H. (2011). Wireless sensor network deployment for integrating video-surveillance and data-monitoring in precision agriculture over distributed crops. *Computers and Electronics in Agriculture*, Volume 75(2):288–303. (Cited on page 13.)

Arduino.cc. *Arduino Mega Board*. Arduino.cc, Ivrea, Italy.

Ashaghathra, S. M. (2008). *Identification of Pecan Weevils Through Image Processing*. ProQuest. (Cited on pages 3, 15, 16, 41, 69, 75, 116, 141, 144, and 149.)

Bailey, D. R., Selker, J. S., Owen, J. S., and Wagner, J. (In Review). Wireless network performance at a production container nursery. *Applied Engineering in Agriculture*.

Baluja, J., Diago, M. P., Balda, P., Zorer, R., Meggio, F., Morales, F., and Tardaguila, J. (2012). Assessment of vineyard water status variability by thermal and multispectral imagery using an unmanned aerial vehicle (UAV). *Irrigation Science*, 30(6):511–522. (Cited on page 12.)

Bergerman, M., Singh, S., and Hamner, B. (2012). Results with autonomous vehicles operating in specialty crops. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1829–1835. IEEE. (Cited on page 12.)

Berni, J., Zarco-Tejada, P. J., Suárez, L., and Fereres, E. (2009). Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle. *Geoscience and Remote Sensing, IEEE Transactions on*, 47(3):722–738. (Cited on page 13.)

Bhatia, A. and Wolf, E. (1954). On the circle polynomials of zernike and related orthogonal sets. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 50, pages 40–48. Cambridge Univ Press. (Cited on page 16.)

Bloem, S., Mizell III, R. F., and OBrien, C. W. (2002). Old traps for new weevils: new records for curculionids (coleoptera: Curculionidae), brentids (coleoptera: Brentidae) and anthribids (coleoptera: Anthribidae) from jefferson co., florida. *Florida Entomologist*, 85(4):632–644.

Bloom, B. S., Engelhart, M., Furst, E. J., Hill, W. H., and Krathwohl, D. R. (1956). Taxonomy of educational objectives: Handbook i: Cognitive domain. *New York: David McKay*, 19:56.

Boethel, D. and Eikenbary, R. (1979). Pecan weevil : seasonal emergence of larvae from three pecan cultivars in oklahoma. *Georgia Entomological Society. Journal*, 14(1):75–83.

Calcote, V. and Hyder, D. (1981). Pecan weevil preference for various pecan cultivars. *Journal of economic entomology*, 74(2):223–226.

Celebi, M. and Aslandogan, Y. (2005). A comparative study of three moment-based shape descriptors. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 1, pages 788–793Vol. 1. (Cited on page 16.)

Chen, Q.-S., Defrise, M., and Deconinck, F. (1994). Symmetric phase-only matched filtering of Fourier-mellin transforms for image registration and recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(12):1156–1168. (Cited on pages 16 and 41.)

Choras, R. (2007). Feature extraction of gray-scale handwritten characters using Gabor filters and zernike moments. In Kurzynski, M., Puchala, E., Wozniak, M., and Zolnierek, A., editors, *Computer Recognition Systems 2*, volume 45 of *Advances in Soft Computing*, pages 340–347. Springer Berlin Heidelberg.

CMUCAM Team (2007). Cmucam3 datasheet. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA.

Coopmans, C. (2009). AggieNav: A small, well integrated navigation sensor system for small unmanned aerial vehicles. In *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference 2009, DETC2009*, volume 2009, pages 635–640. ASME. (Cited on page 12.)

Cottrell, T. and Wood, B. (2003). Pecan weevil management: past, present and toward a future strategy. *Southwestern entomologist. Supplement*, (27):75–84. (Cited on pages 7 and 9.)

Cottrell, T. E. and Wood, B. W. (2008). Movement of adult pecan weevils curculio caryae within pecan orchards [electronic resource]. *Agricultural and forest entomology*, 10(4):363–373. (Cited on page 7.)

Digi International Inc. (2012). ZigBee RF module development kit. Technical report, Digi International® Inc. , Minnetonka, Minnesota, USA.

Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons. (Cited on pages 17 and 18.)

Dukes, M. and Scholberg, J. (2005). Soil moisture controlled subsurface drip irrigation on sandy soils. *Applied Engineering in Agriculture*, 21(1):89–101. (Cited on page 12.)

Eikenbary, R., McNew, R., Hall, M., Criswell, J., Taylor, G., Hedger, G., Morrison, R., Coppock, S., and Smith, M. (1978). Sampling methods for adult pecan weevils. *Oklahoma State University. Cooperative Extension Service. O.S.U. extension facts. Science serving agriculture*, (7175).

Evans, R. G., Iversen, W. M., and Kim, Y. (2011). Integrated decision support, sensor networks, and adaptive control for wireless site-specific sprinkler irrigation. 28(3):–.

Fernández, J., Palomo, M., Dıaz-Espejo, A., Clothier, B., Green, S., Girón, I., and Moreno, F. (2001). Heat-pulse measurements of sap flow in olives for automating irrigation: tests, root flow and diagnostics of water stress. *Agricultural Water Management*, 51(2):99–123. (Cited on page 12.)

Fernández, J., Romero, R., Montaño, J., Diaz-Espejo, A., Muriel, J., Cuevas, M., Moreno, F., Girón, I., and Palomo, M. (2008). Design and testing of an automatic irrigation controller for fruit tree orchards, based on sap flow measurements. *Crop and Pasture Science*, 59(7):589–598. (Cited on page 12.)

Franzen, A. J., Weckler, P. R., and Wang, N. (2010). Automated in-orchard imaging of pecan weevils for insect recognition. In *ASABE Annual International Meeting Paper*, number 1009637, Pittsburgh, Pennsylvania.

Franzen, A. J., Weckler, P. R., and Wang, N. (2011). Evaluation of single-insect proximity detection methods for use in ultra-low-power systems. In *ASABE Annual International Meeting Paper*, number 1111838, Loiusville, Kentucky. (Cited on page 14.)

Franzen, A. J., Weckler, P. R., and Wang, N. (2012). Wireless signal path loss and transmission success rates in orchard environments. In *ASABE Annual International Meeting Paper*, number 121336701, Dallas, Texas.

Guarnieri, A., Maini, S., Molari, G., and Rondelli, V. (2011). Automatic trap for moth detection in integrated pest management. (Cited on page 13.)

Guo, X., Zheng, L., Li, M., Zhang, Y., Deng, X., and An, X. (2013). A movable data acquisition and production management system for apple orchard. In *ASABE International Meeting*, number 131620114. ASABE.

Hall, M., McNew, R., Eikenbary, R., and Hedger, G. (1981). Impact of pecan weevil on pecan production in a pest-managed commercial orchard. *Environmental entomology*, 10(5):668–672.

Hall, M., Morrison, R., Hedger, G., Eikenbary, R., and McNew, R. (1983). Comparison of a cone emergence trap procedure with th eminotoring of tagged nuts for timing insecticide applications for pecan weevil (coleoptera: Curculionidae) control. *Environmental entomology*, 12(2):505–509.

Hamner, B., Singh, S., and Bergerman, M. (2010). Improving orchard efficiency with autonomous utility vehicles. (Cited on page 12.)

Harris, M., Jackman, J., Aguirre, L., and Ring, D. (1981). Longevity of post-emergent adult pecan weevil in the laboratory and field. *Environmental entomology*, 10(2):201–205.

Harris, M., Neeb, C., Jackman, J., Ring, D., and Cutler, B. (1980). Pecan weevil management in texas. *Pecan quarterly*, 14(2):3.

Harris, M. and Ring, D. (1979). Biology of pecan weevil from oviposition to larval emergence. *Southwestern entomologist*, 4(2):73–85.

Hosny, K. (2012a). Accurate pseudo zernike moment invariants for grey–level images. *Imaging Science Journal, The*, 60(4):234–342. (Cited on pages 15 and 53.)

Hosny, K. M. (2007). Exact and fast computation of geometric moments for gray level images. *Applied mathematics and computation*, 189(2):1214–1222. (Cited on page 15.)

Hosny, K. M. (2008). Fast computation of accurate zernike moments. *Journal of Real–Time Image Processing*, 3(1–2):97–107. (Cited on page 15.)

Hosny, K. M. (2010a). Fast and accurate method for radial moment's computation. *Pattern Recognition Letters*, 31(2):143–150. (Cited on page 15.)

Hosny, K. M. (2010b). New set of rotationally legendre moment invariants. *International Journal of Electrical, Computer, and Systems Engineering*, 4(3):735–747. (Cited on page 15.)

Hosny, K. M. (2010c). A systematic method for efficient computation of full and subsets zernike moments. *Information Sciences*, 180(11):2299–2313. (Cited on page 15.)

Hosny, K. M. (2011a). Accurate orthogonal circular moment invariants of gray–level images. *Journal of Computer Science*, 7(5):715. (Cited on pages 15 and 43.)

Hosny, K. M. (2011b). Image representation using accurate orthogonal gegenbauer moments. *Pattern Recognition Letters*, 32(6):795–804. (Cited on page 15.)

Hosny, K. M. (2012b). Fast computation of accurate Gaussian–Hermite moments for image processing applications. *Digital Signal Processing*, 22(3):476–485. (Cited on pages 15 and 16.)

Hosny, K. M., Shouman, M. A., and Salam, H. M. A. (2011). Fast computation of orthogonal Fourier–mellin moments in polar coordinates. *Journal of Real–Time Image Processing*, 6(2):73–80. (Cited on pages 15 and 41.)

Jennic, NXP Semiconductors (2010). Jn5139 module datasheet. Technical report, Jennic, NXP Semiconductors, Eindhoven, The Netherlands.

Johnson, D. T., Mulder, Jr, P. G., McCraw, B. D., Lewis, B. A., Jervis, B., Carroll, B., and McLeod, P. J. (2002). Trapping plum curculio conotrachelus nenuphar (herbst)(coleoptera: Curculionidae) in the southern United States. *Environmental entomology*, 31(6):1259–1267.

Khotanzad, A. and Hong, Y. H. (1990). Invariant image recognition by zernike moments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(5):489–497. (Cited on page 16.)

Lang, Z. (1999). Automatic steering control of plantation tractor based on image processing. In *Proceedings of SPIE*, volume 3543, page 246. (Cited on page 13.)

Leskey, T. C. and Wright, S. E. (2004). Monitoring plum curculio, conotrachelus nenuphar (coleoptera: Curculionidae), populations in apple and peach orchards in the mid-atlantic. *Journal of economic entomology*, 97(1):79–88.

Li, Z., Wang, N., Hong, T., and Franzen, A. J. (2010). Experimental path-loss models for 2.4GHz in-field wireless sensor network. In *ASABE Annual International Meeting Paper*, number 1008576, Pittsburgh, Pennsylvania.

Majone, B., Viani, F., Filippi, E., Bellin, A., Massa, A., Toller, G., Robol, F., and Salucci, M. (2013). Wireless sensor network deployment for monitoring soil moisture dynamics at the field scale. *Procedia Environmental Sciences*, 19(0):426–435. <ce:title>Four Decades of Progress in Monitoring and Modeling of Processes in the Soil-Plant-Atmosphere System: Applications and Challenges</ce:title>. (Cited on page 13.)

Moruzzi, T. L., Fuller, T. K., DeGraaf, R. M., Brooks, R. T., and Li, W. (2002). Assessing remotely triggered cameras for surveying carnivore distribution. *Wildlife Society Bulletin*, pages 380–386. (Cited on page 13.)

Mukundan, R. and Ramakrishnan, K. (1995). Fast computation of legendre and zernike moments. *Pattern Recognition*, 28(9):1433–1442. (Cited on page 16.)

Mulder, P. and Grantham, R. A. (2007). *Biology and control of the pecan weevil in Oklahoma*. Division of Agricultural Sciences and Natural Resources, Oklahoma State University. (Cited on page 7.)

Mulder, P., Reid, W., Grantham, R., Landgraf, S., Taliaferro, L., Payton, M., and Knutson, A. (2003). Evaluations of trap designs and a pheromone formulation used for monitoring pecan weevil, curculio caryae. *Southwestern Entomologist Supplement*, 27:85–99. (Cited on pages 9 and 10.)

Mulder, P. G., Harris, M. K., and Grantham, R. A. (2012). Biology and management of the pecan weevil (coleoptera: Curculionidae). *Journal of Integrated Pest Management*, 3(1):A1–A9. (Cited on pages 2, 5, 6, 7, 8, 9, 10, 11, and 12.)

Nelson, S. and Payne, J. (1982a). Pecan weevil control by dielectric heating. *Journal of microwave power*, 17(1):51–55.

Nelson, S. and Payne, J. (1982b). RF dielectric heating for pecan weevil control. *Transactions of the ASAE - American Society of Agricultural Engineers*, 25(2):456.

Nordström, K., Barnett, P. D., and O'Carroll, D. C. (2006). Insect detection of small targets moving in visual clutter. *PLoS biology*, 4(3):e54. (Cited on page 14.)

Oerke, E.-C., Fröhling, P., and Steiner, U. (2011). Thermographic assessment of scab disease on apple leaves. *Precision Agriculture*, 12(5):699–715. (Cited on page 12.)

Papakostas, G., Boutalis, Y., Karras, D., and Mertzios, B. (2010). Efficient computation of zernike and pseudo-zernike moments for pattern classification applications. *Pattern Recognition and Image Analysis*, 20(1):56–64. (Cited on page 16.)

Payne, J., Lockwood, D., and Ellis, H. (1979). Biology and distribution of the pecan weevil in Georgia and tennessee. *Pecan South*, 6(1):30–33.

Phillips Semiconductor, N. (2008). Lpc2106 Arm7Tdmi microcontroller. Technical report, Phillips Semiconductor, NXP Semiconductor, Eindhoven, The Netherlands.

Pierce, F. and Elliott, T. (2008). Regional and on-farm wireless sensor networks for agricultural systems in eastern washington. *Computers and Electronics in Agriculture*, 61(1):32–43. <ce:title>Emerging Technologies For Real-time and Integrated Agriculture Decisions</ce:title>. (Cited on page 13.)

Ree, B., Knutson, A., and Harris, M. (2005). Controlling the pecan weevil. *Available electronically from http://hdl. handle. net/1969*, 1:86939.

Reid, W. (2002). Current pest management systems for pecan. *HortTechnology*, 12(4):633–639.

Reyes, J. F., Correa, C., Esquivel, W., and Ortega, R. (2012). Development and field testing of a data acquisition system to assess the quality of spraying in fruit orchards. *Computers and Electronics in Agriculture*, 84:62–67.

Ridgway, C. and Chambers, J. (1996). Detection of external and internal insect infestation in wheat by near-infrared reflectance spectroscopy. *Journal of the Science of Food and Agriculture*, 71(2):251–264. (Cited on page 14.)

Ring, D., Haensly, T., Cutler, B., and Harris, M. (1981). A gravity cage for monitoring pecan nuts, leaves and pecan weevil dropping from pecan trees. *Pecan quarterly*, 15(1):23–26.

Ring, D., Snow, J., Payne, J., and Grauke, L. (1991). Tree species used as hosts by pecan weevil (coleoptera: Curculionidae). *Journal of economic entomology*, 84(6):1782–1789.

Rowe, A., Rosenberg, C., and Nourbakhsh, I. (2002). Cmucam: a lowoverhead vision system. In *Proceedings, IROS 2002*.

Schraer, S., Biggerstaff, M., Jackman, J., and Harris, M. (1998). Pecan weevil (coleoptera: Curculionidae) emergence in a range of soil types. *Environmental entomology*, 27(3):549–554. (Cited on page 7.)

Shapiro-Ilan, D., Fuxa, J., Wood, B., and Gardner, W. (2007). Methods and materials for control of insects such as pecan weevils [electronic resource]. *United States Department of Agriculture patents*, (US 7,241,612 B).

Shen, J. (1997). Orthogonal Gaussian-Hermite moments for image characterization. *Proc. SPIE, Intelligent Robots and Computer Vision XVI: Algorithms Techniques, Active Vision, and Materials Handling, Pittsburgh, USA*, pages 875–894.

Singh, C., Pooja, S., and Upneja, R. (2011). On image reconstruction, numerical stability, and invariance of orthogonal radial moments and radial harmonic transforms. *Pattern recognition and image analysis*, 21(4):663–676.

Smith, M. W. and Mulder, P. G. (2009). Oviposition characteristics of pecan weevil. *Southwestern entomologist*, 34(4):447–455. (Cited on page 6.)

Subramanian, V., Burks, T. F., and Arroyo, A. (2006). Development of machine vision and laser radar based autonomous vehicle guidance systems for citrus grove navigation. *Computers and electronics in agriculture*, 53(2):130–143. (Cited on page 13.)

Tedders, W., Mizell III, R., and Wood, B. (1996). Influence of trap color on pecan weevil monitoring. *J. Entomol. Sci*, 31:414–419.

The MathWorks Inc. (2014). *MATLAB with Image Processing and Statistics Toolboxes Release 2014b*. Natick, Massachusetts. (Cited on page 22.)

The MathWorks Inc. (2015). *MATLAB with Image Processing and Statistics Toolboxes Release 2015a*. Natick, Massachusetts. (Cited on page 22.)

Tumbo, S., Salyani, M., Whitney, J., Wheaton, T. A., and Miller, W. M. (2002). Investigation of laser and ultrasonic ranging sensors for measurements of citrus canopy volume. (Cited on page 12.)

Twain, M. (1898). *Following the Equator: A Journey Around the World*. American Publishing Company.

USDA (2015). Pecan report. (Cited on page 1.)

Vick, K. W., Webb, J., and Litzkow, C. A. (1991). Insect detection using a pitfall probe trap having vibration detection. US Patent 5,005,416. (Cited on pages 14 and 16.)

Vougioukas, S., Anastassiu, H., Regen, C., and Zude, M. (2013). Influence of foliage on radio path losses (pls) for wireless sensor network (WSN) planning in orchards. *Biosystems Engineering*, 114(4):454–465. <ce:title>Special Issue: Sensing Technologies for Sustainable Agriculture</ce:title>.

Walia, E., Singh, C., and Goyal, A. (2012). On the fast computation of orthogonal Fourier–mellin moments with improved numerical stability. *Journal of Real-Time Image Processing*, 7(4):247–256.

Wang, N., Zhang, N., and Wang, M. (2006). Wireless sensors in agriculture and food industry: Recent development and future perspective. *Computers and Electronics in Agriculture*, 50(1):1–14.

Wang, X. and Liao, S. (2013). Image reconstruction from orthogonal Fourier-mellin moments. In *Image Analysis and Recognition*, pages 687–694. Springer. (Cited on page 41.)

Watson, A. T., O'Neill, M. A., and Kitching, I. J. (2004). Automated identification of live moths (macrolepidoptera) using digital automated identification system (daisy). *Systematics and Biodiversity*, 1(3):287–300. (Cited on page 14.)

Wen, C. and Guyer, D. (2012). Image-based orchard insect automated identification and classification method. (Cited on page 13.)

Xin, Y., Pawlak, M., and Liao, S. (2007). Accurate computation of zernike moments in polar coordinates. *Image Processing, IEEE Transactions on*, 16(2):581–587.

Yang, X., Guo, X., Li, M., Sun, C., Hao, L., Qu, L., and Wang, Y. (2013). An empirical model for 2.4 ghz radio propagation in a gala apple orchard and evaluation of the model performance by simulation. *Transactions of the ASABE*, 56(4):1599–1611.

Yoo, S.-E., Kim, J.-E., Kim, T., Ahn, S., Sung, J., and Kim, D. (2007). A2S: Automated agriculture system based on WSN. In *Consumer Electronics, 2007. ISCE 2007. IEEE International Symposium on*, pages 1–5.

Zernike, v. F. (1934). Beugungstheorie des schneidenver-fahrens und seiner verbesserten form, der phasenkontrastmethode. *Physica*, 1(7):689–704. (Cited on pages 16 and 43.)

Zhang, C. and Kovacs, J. M. (2012). The application of small unmanned aerial systems for precision agriculture: a review. *Precision Agriculture*, 13(6):693–712.

Zhenjiang, M. (2000). Zernike moment-based image shape analysis and its application. *Pattern Recognition Letters*, 21(2):169–177. (Cited on page 16.)

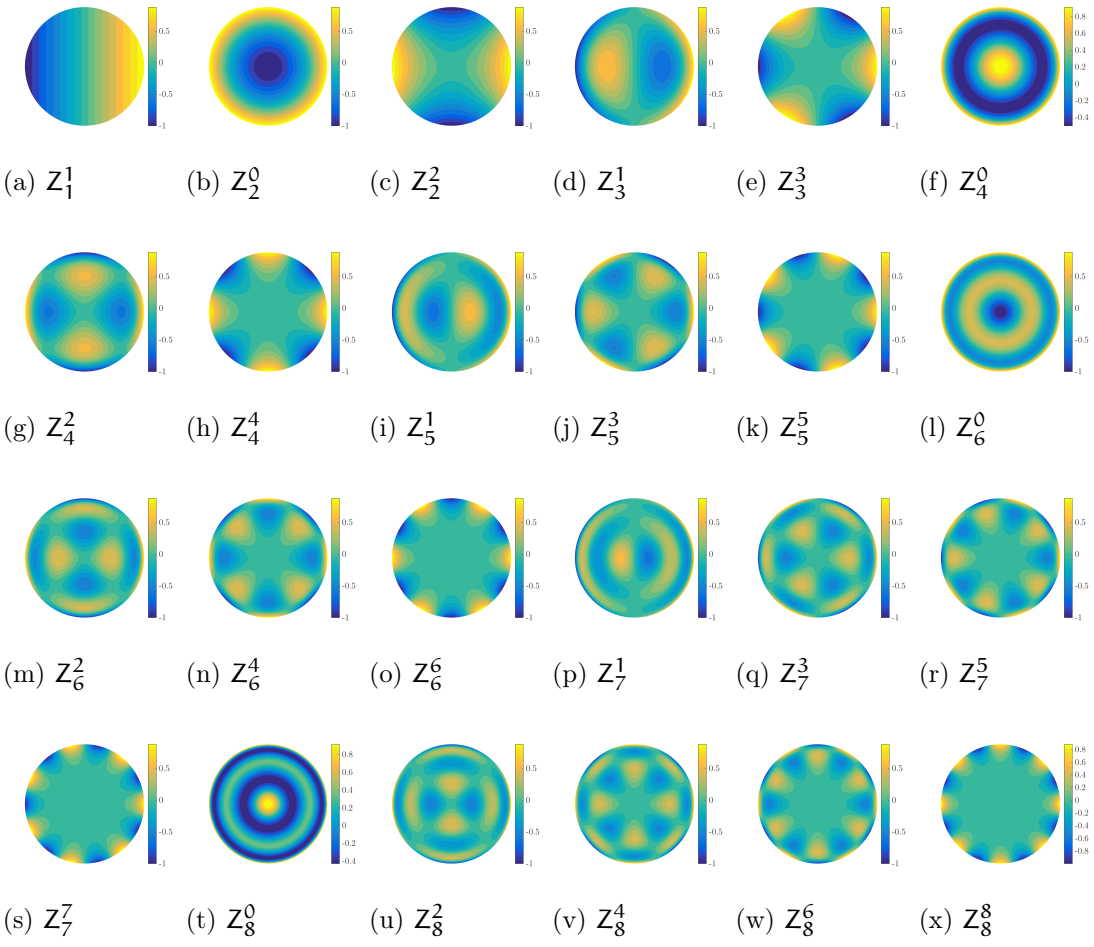
Zhu, H., Shu, H., Liang, J., Luo, L., and Coatrieux, J.-L. (2007a). Image analysis by discrete orthogonal racah moments. *Signal Processing*, 87(4):687–708. (Cited on page 16.)

Zhu, H., Shu, H., Zhou, J., Luo, L., and Coatrieux, J.-L. (2007b). Image analysis by discrete orthogonal dual hahn moments. *Pattern Recognition Letters*, 28(13):1688–1704. (Cited on page 16.)

APPENDICES

APPENDIX A

ZERNIKE MOMENT BASIS FUNCTIONS



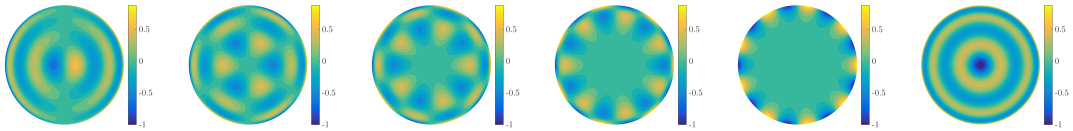
(a) Z_1^1 (b) Z_2^0 (c) Z_2^2 (d) Z_3^1 (e) Z_3^3 (f) Z_4^0

(g) Z_4^2 (h) Z_4^4 (i) Z_5^1 (j) Z_5^3 (k) Z_5^5 (l) Z_6^0

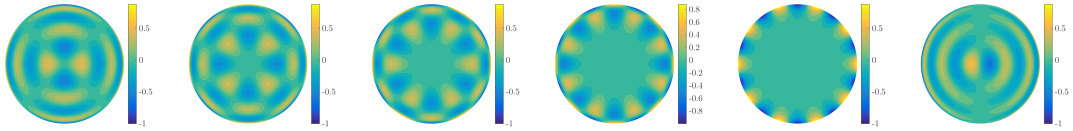
(m) Z_6^2 (n) Z_6^4 (o) Z_6^6 (p) Z_7^1 (q) Z_7^3 (r) Z_7^5

(s) Z_7^7 (t) Z_8^0 (u) Z_8^2 (v) Z_8^4 (w) Z_8^6 (x) Z_8^8

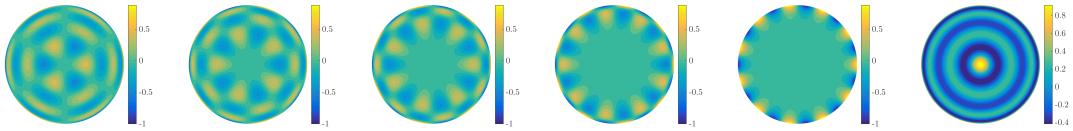
Figure A.1: Real components of the ZM basis functions Z_1^1 through Z_8^8. The negative repetition values are excluded since they are equivalent to rotations of the positive repetition.



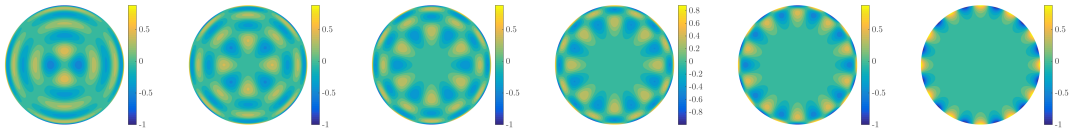
(a) Z_9^1 (b) Z_9^3 (c) Z_9^5 (d) Z_9^7 (e) Z_9^9 (f) Z_{10}^0



(g) Z_{10}^2 (h) Z_{10}^4 (i) Z_{10}^6 (j) Z_{10}^8 (k) Z_{10}^{10} (l) Z_{11}^1



(m) Z_{11}^3 (n) Z_{11}^5 (o) Z_{11}^7 (p) Z_{11}^9 (q) Z_{11}^{11} (r) Z_{12}^0

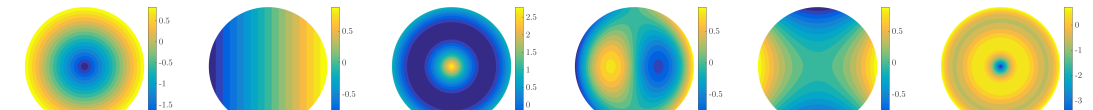


(s) Z_{12}^2 (t) Z_{12}^4 (u) Z_{12}^6 (v) Z_{12}^8 (w) Z_{12}^{10} (x) Z_{12}^{12}

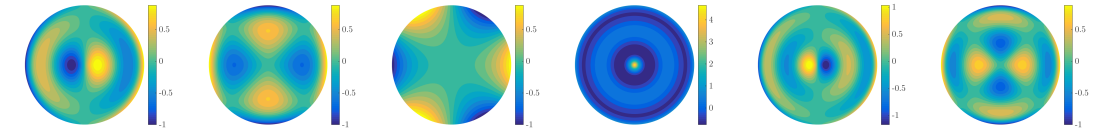
Figure A.2: Real components of the ZM basis functions Z_9^1 through Z_{12}^{12}. The negative repetition values are excluded since they are equivalent to rotations of the positive repetition.

APPENDIX B

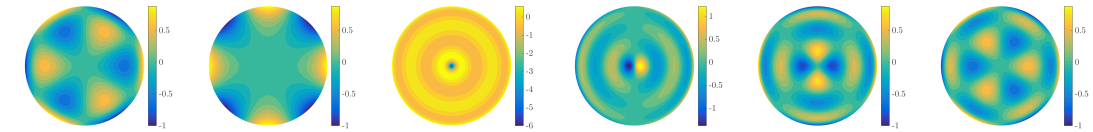
PSEUDOZERNIKE MOMENT BASIS FUNCTIONS



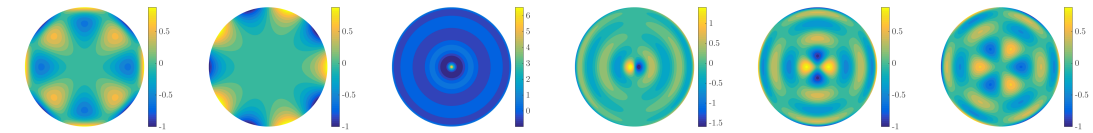
(a) pZ_1^0 (b) pZ_1^1 (c) pZ_2^0 (d) pZ_2^1 (e) pZ_2^2 (f) pZ_3^0



(g) pZ_3^1 (h) pZ_3^2 (i) pZ_3^3 (j) pZ_4^0 (k) pZ_4^1 (l) pZ_4^2

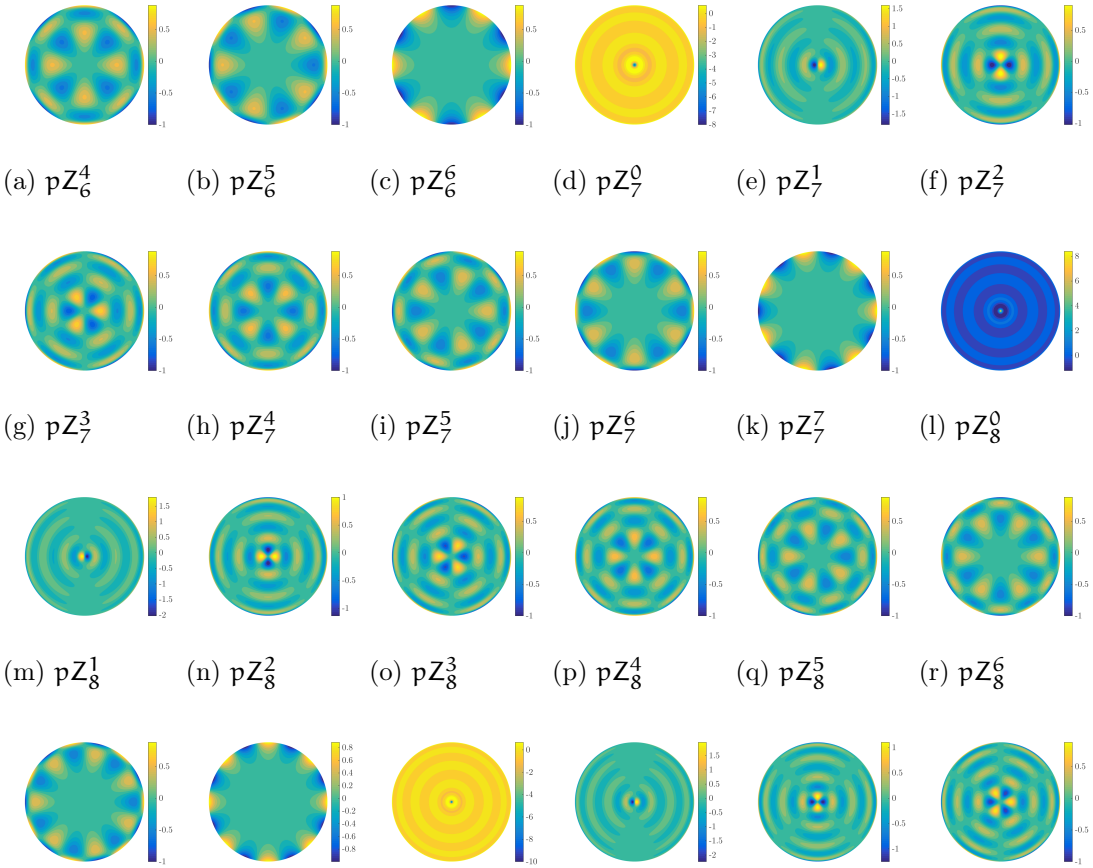


(m) pZ_4^2 (n) pZ_4^4 (o) pZ_5^0 (p) pZ_5^1 (q) pZ_5^2 (r) pZ_5^3



(s) pZ_5^4 (t) pZ_5^5 (u) pZ_6^0 (v) pZ_6^1 (w) pZ_6^2 (x) pZ_6^3

Figure B.1: Real components of the pZM basis functions pZ_1^0 through pZ_6^3. The negative repetition values are excluded since they are equivalent to rotations of the positive repetition.



(a) pZ_6^4 (b) pZ_6^5 (c) pZ_6^6 (d) pZ_7^0 (e) pZ_7^1 (f) pZ_7^2

(g) pZ_7^3 (h) pZ_7^4 (i) pZ_7^5 (j) pZ_7^6 (k) pZ_7^7 (l) pZ_8^0

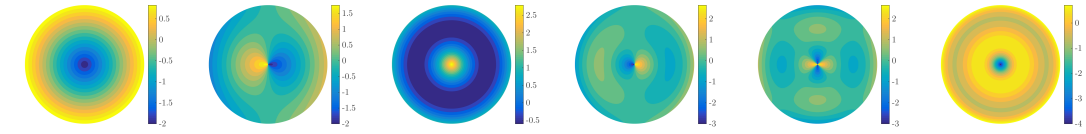
(m) pZ_8^1 (n) pZ_8^2 (o) pZ_8^3 (p) pZ_8^4 (q) pZ_8^5 (r) pZ_8^6

(s) pZ_8^7 (t) pZ_8^8 (u) pZ_9^0 (v) pZ_9^1 (w) pZ_9^2 (x) pZ_9^3

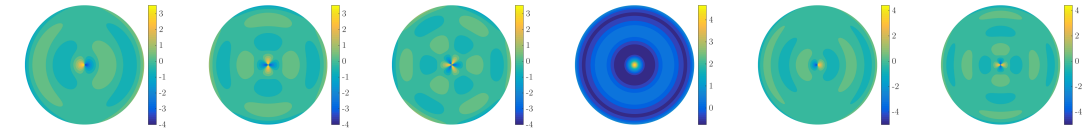
Figure B.2: Real components of the pZM basis functions pZ_6^4 through pZ_9^3. The negative repetition values are excluded since they are equivalent to rotations of the positive repetition.

APPENDIX C

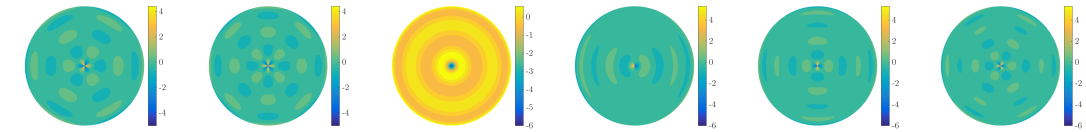
FOURIER MELLIN MOMENT BASIS FUNCTIONS



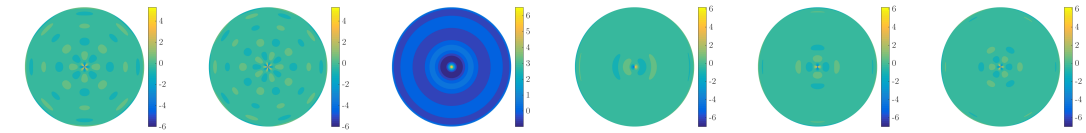
(a) FMM_1^0 (b) FMM_1^1 (c) FMM_2^0 (d) FMM_2^1 (e) FMM_2^2 (f) FMM_3^0



(g) FMM_3^1 (h) FMM_3^2 (i) FMM_3^3 (j) FMM_4^0 (k) FMM_4^1 (l) FMM_4^2

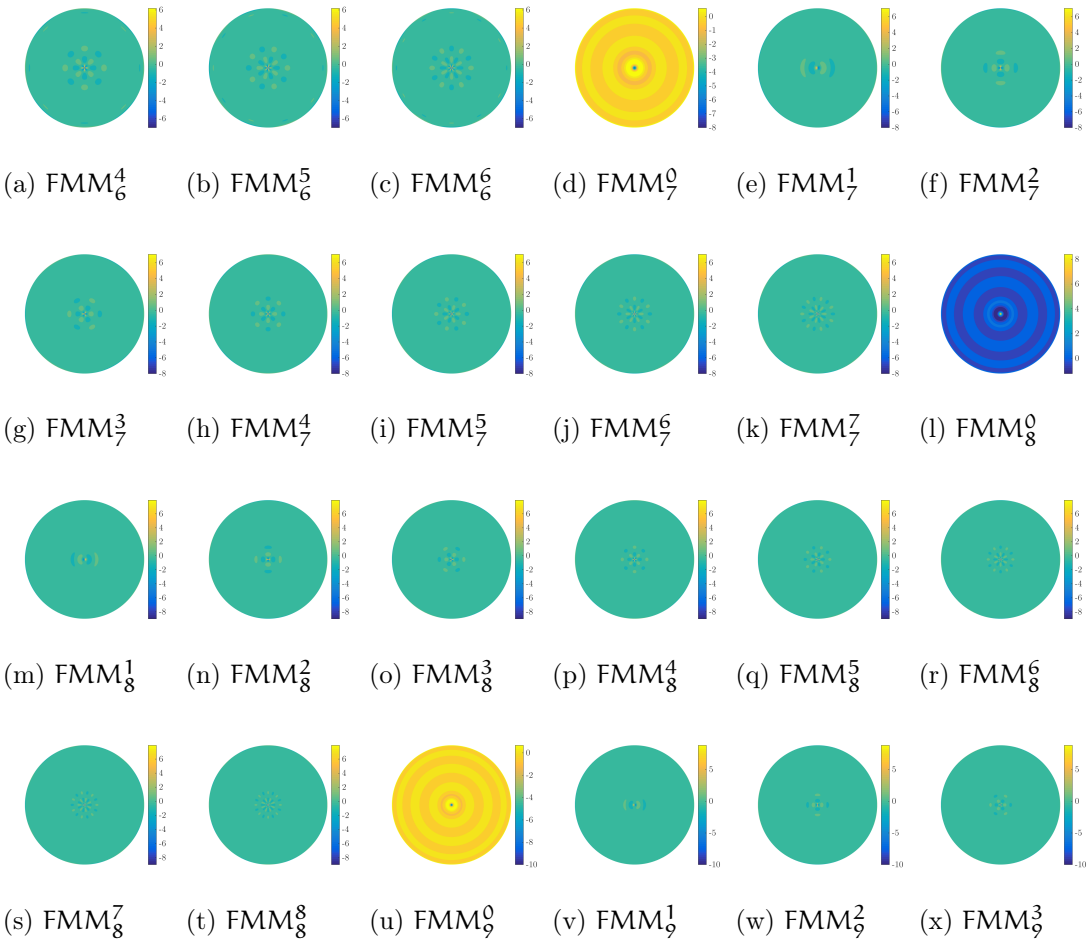


(m) FMM_4^2 (n) FMM_4^4 (o) FMM_5^0 (p) FMM_5^1 (q) FMM_5^2 (r) FMM_5^3



(s) FMM_5^4 (t) FMM_5^5 (u) FMM_6^0 (v) FMM_6^1 (w) FMM_6^2 (x) FMM_6^3

Figure C.1: Real components of the FMM basis functions FMM_1^0 through FMM_6^3. The negative repetition values are excluded since they are equivalent to rotations of the positive repetition.



(a) FMM_6^4 (b) FMM_6^5 (c) FMM_6^6 (d) FMM_7^0 (e) FMM_7^1 (f) FMM_7^2

(g) FMM_7^3 (h) FMM_7^4 (i) FMM_7^5 (j) FMM_7^6 (k) FMM_7^7 (l) FMM_8^0

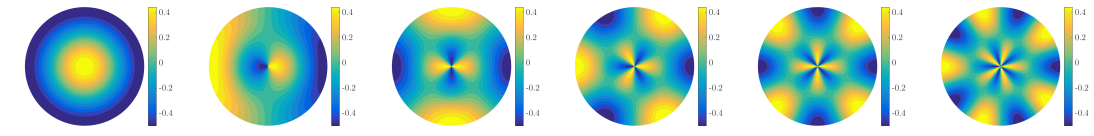
(m) FMM_8^1 (n) FMM_8^2 (o) FMM_8^3 (p) FMM_8^4 (q) FMM_8^5 (r) FMM_8^6

(s) FMM_8^7 (t) FMM_8^8 (u) FMM_9^0 (v) FMM_9^1 (w) FMM_9^2 (x) FMM_9^3

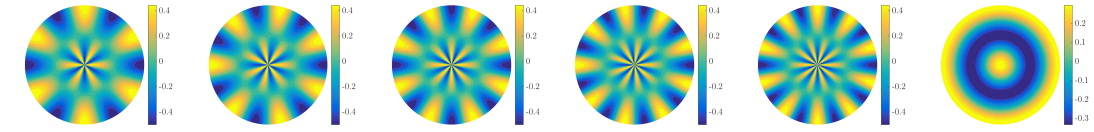
Figure C.2: Real components of the FMM basis functions FMM_6^4 through FMM_9^3. The negative repetition values are excluded since they are equivalent to rotations of the positive repetition.

APPENDIX D

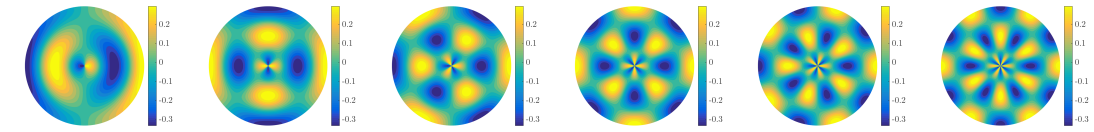
MPEG-7 ART MOMENT BASIS FUNCTIONS



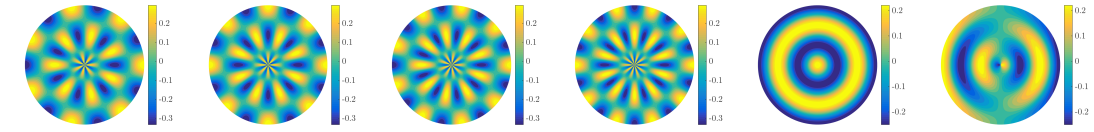
(a) ART_1^0 (b) ART_1^1 (c) ART_1^2 (d) ART_1^3 (e) ART_1^4 (f) ART_1^5



(g) ART_1^6 (h) ART_1^7 (i) ART_1^8 (j) ART_1^9 (k) ART_1^{10} (l) ART_2^0

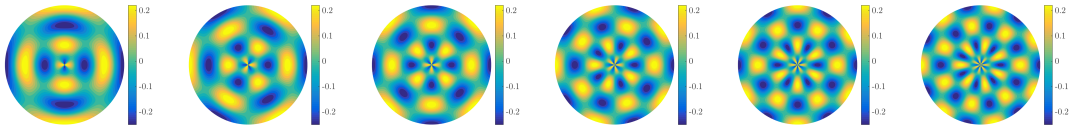


(m) ART_2^1 (n) ART_2^2 (o) ART_2^3 (p) ART_2^4 (q) ART_2^5 (r) ART_2^6

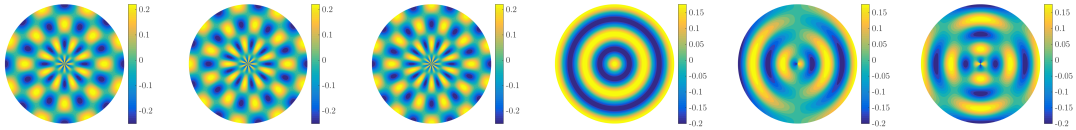


(s) ART_2^7 (t) ART_2^8 (u) ART_2^9 (v) ART_2^{10} (w) ART_3^0 (x) ART_3^1

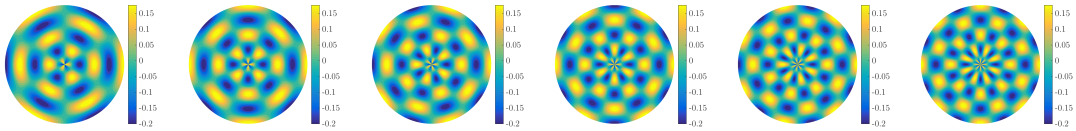
Figure D.1: Real components of the ART basis functions ART_1^0 through ART_3^1. The negative repetition values are excluded since they are equivalent to rotations of the positive repetition.



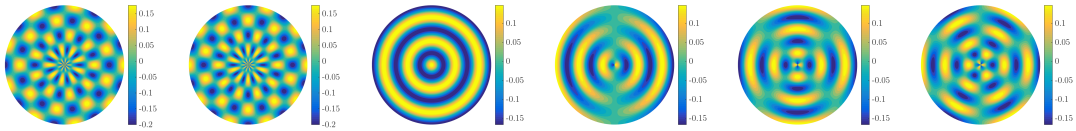
(a) ART_3^2 (b) ART_3^3 (c) ART_3^4 (d) ART_3^5 (e) ART_3^6 (f) ART_3^7



(g) ART_3^8 (h) ART_3^9 (i) ART_3^{10} (j) ART_4^0 (k) ART_1^4 (l) ART_4^2



(m) ART_4^3 (n) ART_4^4 (o) ART_4^5 (p) ART_4^6 (q) ART_4^7 (r) ART_4^8



(s) ART_4^9 (t) ART_4^{10} (u) ART_5^0 (v) ART_5^1 (w) ART_5^2 (x) ART_5^3

Figure D.2: Real components of the ART basis functions ART_3^2 through ART_5^3. The negative repetition values are excluded since they are equivalent to rotations of the positive repetition.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

`http://code.google.com/p/classicthesis/`

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured at:

`http://postcards.miede.de/`

VITA

Aaron John Franzen

Candidate for the Degree of

Doctor of Philosophy

Thesis: In-Orchard imaging of pecan weevil and efficient identification using orthogonal polar moment descriptors

Biographical:

Born in Gothenburg, Nebraska, USA. Son of the late Rodney M Franzen and Ruth A Ostergard.

Education:

Completed the requirements for the Doctor of Philosophy in Biosystems & Agricultural Engineering at Oklahoma State University, Stillwater, Oklahoma in December, 2015.

Completed the requirements for the Master of Science in Mechanical & Aerospace Engineering at the University of Florida, Gainesville, Florida in 2007.

Completed the requirements for the Bachelor of Science in your Agricultural Engineering at the University of Nebraska–Lincoln, Lincoln, Nebraska in 2003.

Experience: Alumni Research Fellow, University of Florida, 2003-2007; Research Engineer, Oklahoma State University, 2007-2014

Professional Memberships: American Society of Agricultural & Biological Engineering, Alpha Epsilon, Tau Beta Pi