3D SKETCH RECOGNITION USING THE MICROSOFT KINECT

by

Travis Bulgerin, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Computer Science
May 2014

Committee Members:

Yijuan Lu, Chair

Anne Ngu

Ziliang Zong

DEDICATION

Dedicated to my family and friends who have supported me in my goals to pursue higher education.

ACKNOWLEDGMENTS

Foremost, I would like to express my gratitude to my advisor Dr. Yijuan Lu for her continuous support and guidance throughout my research. Without her help and encouragement, this thesis would not have been possible.

I would also like to thank the rest of my thesis committee: Dr. Anne Ngu and Dr. Ziliang Zong for their insightful comments and support. My sincere thanks also go to Bo Li for his guidance throughout this process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The concept of sketch-based recognition has recently been used to enhance object categorization and speed up image retrieval. However, in each of the previous studies, the user was required to sketch on a two-dimensional plane. Currently there haven't been any studies on the performance of incorporating depth information into a sketch. The motivation behind this project is to develop software that will allow a user to draw in a three-dimensional space, incorporating this information, as well as determining whether this depth information will result in higher accuracies for object categorization. First, utilizing the Microsoft Kinect, software was developed to establish a virtual drawing board for three-dimensional sketching. Second, a new learning-based approach is proposed to allow for model feature extraction and recognition. The experimental results demonstrate the validity of the study as well as the effectiveness of the proposed solution.

CHAPTER I

INTRODUCTION

Since humanity's evolution of opposable thumbs, sketching has been used to convey information and depict the visual world. Sketching is one of the oldest forms of communication, and yet it is still difficult for a computer to understand the semantics behind a sketch, something humans have been able to do for thousands of years. Two of the most successful approaches to sketch recognition have been in measuring geometric similarities and semantic understanding. However, only recently has research begun to unveil the potential of this field.

Sketch recognition is the automated recognition of hand drawn images by a computer. The sketches are usually binary in color (black and white) and only contain the image to be recognized (no context around the sketch). Sketch recognition has a variety of useful applications, one of the most common being image retrieval. Before sketch recognition, image retrieval from a database full of thousands of images used to be a fairly complex task, as traditional databases operate on text. Therefore, each image needed to be tagged with some keywords used to represent the context of the image, which had to be input manually, making it very tedious. However, sketch-based recognition has helped to change that. By recognizing a hand drawn sketch and categorizing its context, users are able to search image databases much more efficiently.

In recent areas of research, the main challenges in sketch recognition have been in measuring geometric similarity and achieving semantic understanding. The first approach involves measuring the geometric similarities between a sketch based image and a ground truth image. This manner of recognition can significantly speed up image retrieval, since this measure of similarity can be cast as a nearest neighbor problem. However, it requires each sketch to be drawn perfectly to scale, which is not something humans normally do. Instead, we typically draw in an abstract way that is not very geometrically similar to the ground truth image (Eitz et al. 2012).

The semantic understanding approach to sketch-based recognition comes from a computer's ability to recognize higher levels of complexities within the sketch, and match it to a ground truth image. In this approach, a machine-learning algorithm is used to understand the semantic meaning of the sketch by learning from previously drawn sketches. This is a relatively new approach to the sketch based recognition problem and has had great success in the last two years.

The field of three-dimensional sketch-based recognition is a new area of study. Very little preliminary work exists in this field, allowing for many exciting and interesting results. The implications of development could be tremendous, as three-dimensional sketching allows for more detail in a user's drawing, which could result in higher accuracy of object categorization. Additionally, this project could lead to increases in speed for image retrieval, as clear categorization of sketches could narrow the database searches.

This project seeks to explore this area of three-dimensional sketch-based recognition, which allows a user to sketch a three dimensional object in a three dimensional space. In previous research, users had to draw a sketch on a two-dimensional plane. This can result in a loss of information along the z-axis, known as the depth information. Currently, there does not exist software to allow for a user to include the depth information in her sketch, as well as whether incorporating this information will result in increased objection categorization performance. The motivation behind this project is twofold. First, to develop software to establish a virtual drawing board that will allow a user to draw a three-dimensional sketch, utilizing the Microsoft Kinect. Second, to explore whether incorporating this depth information will result in higher accuracy of object categorization.

CHAPTER II

BACKGROUND

In this chapter we discuss the necessary background for understanding this thesis. First a brief description about sketch recognition is given. Then we discuss the technology used to capture the three-dimensional data points. Next, a mathematical filter is introduced that is used to smooth the coordinate data. Finally we give a brief introduction about machine learning, focusing primarily on support vector machines, which this project uses for the object categorization.

2.1 Sketch Recognition

Sketch recognition is the automated recognition of hand-drawn images by a computer. There are a number of different algorithms to accomplish this task, such as geometry-based, appearance-based, and gesture-based to name a few. Sketch recognition is a unique topic in computer science, as it does not belong to a particular field, but instead, it is a combination of multiple fields of computer science, including computer vision, artificial intelligence, and human-computer interaction.

In sketch-based recognition, a user draws a binary sketch of an object. This is typically a simple object, with no context drawn around it. In Figure II.1 we see an example of a typical hand-drawn sketch. This figure depicts a binary sketch of a barn drawn by the user. Notice the figure is only a picture of the barn, there is no context drawn around object, such as cows or tractors. This is what is meant by the object should be drawn with no surrounding context.

After a user sketches the binary image, it is the job of the sketch recognition algorithm to then classify this object into a particular category from a predefined set. This is typically done by extracting features from the sketch that make it unique to that particular category. For example, the features extracted from a sketch of a barn should be different from the same features extracted from the sketch of a car. A machine-learning algorithm to classify the sketch into its correct category then utilizes these features.

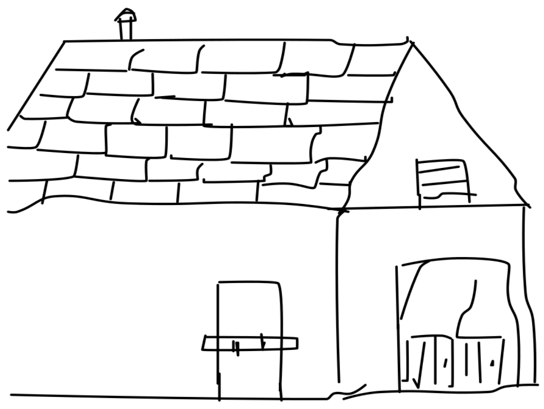

Figure II.1 A binary sketch of a barn (Eitz et al. 2012)

2.2 Microsoft Kinect

The Microsoft Kinect sensor is what is used to collect the 3D sketching data from each individual user in this project. This section will give a brief introduction to what the Kinect sensor is, and the hardware and software that is behind it.

The Microsoft Kinect is a motion sensing input device that was developed by Microsoft for the Xbox 360 video game console. The Kinect was designed to allow the user to interact with an application using a natural user interface using gestures and voice commands rather than a controller or keyboard/mouse.

The Kinect sensor is an advanced piece of technology with many interesting hardware features. As Figure II.2 demonstrates, the Kinect hardware consists of the following: a color VGA video camera, a depth sensor, and a multi-array microphone. The color VGA video camera is a normal RGB camera that is used to capture the surroundings just like a normal webcam. The depth sensor consists of an infrared projector and a monochrome CMOS sensor working together to capture the three-dimensional depth of the current room. This is the primary sensor utilized in this project; as the depth sensor is able to capture coordinate points in a three-dimensional space. The multi-array microphone is an array of four microphones that can be used to isolate a user's voice for input into voice commands. The video and depth sensors have a pixel resolution of 640x480 and run at 30 frames per second (Kinect for Windows Sensor Components and Specifications, 2014).

The Kinect software maps the layout of the room you are currently in and configures the space the user will be moving around in. The Kinect then detects and tracks 48 individual points on the user's body. This allows the Kinect to track individual portions of a user's body, such as a user's hands or feet. This is an extremely useful feature, as this project uses this software to track a user's hand in a three-dimensional space while she is drawing.

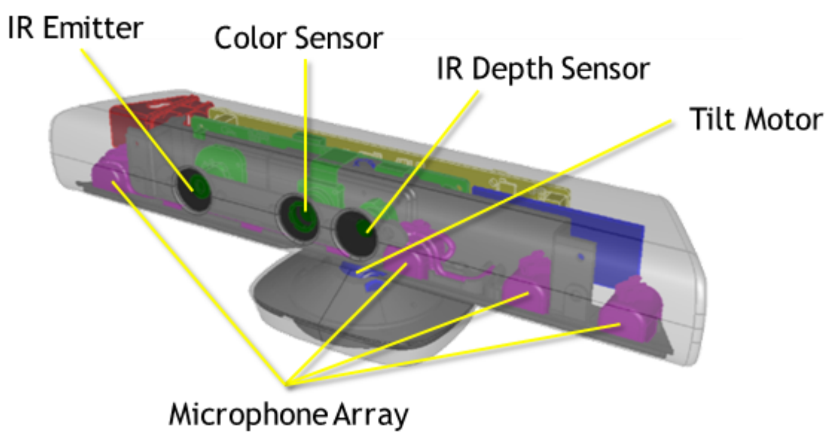

Figure II.2 The hardware of the Microsoft Kinect (Kinect for Windows Sensor Components and Specifications, 2014)

2.3 Kalman Filter

The Microsoft Kinect allows us to capture the three-dimensional data points from a user; however, this piece of equipment is not perfect. Specifically, there is a lot of noise that is included with the captured data. To solve this problem, a Kalman filter is used. A Kalman filter is an optimal estimator, meaning it infers parameters of interest from inaccurate and uncertain observations.

The primary purpose of a Kalman filter is to take a series of measurements observed over time and produce estimates of unknown variables which tend to be more precise than those based on a single variable alone. As Figure II.3 demonstrates, this can be broken into two steps: the measurement update and the prediction update.

The measurement update is accomplished using the Bayes Rule. Essentially this update takes a prior Gaussian and a measurement Gaussian and multiplies them together to get a resultant Gaussian with a smaller covariance, which is the uncertainty. The mean and covariance of this new Gaussian is then used in the prediction update.

The prediction update is accomplished through Gaussian addition. Here we take the mean and covariance from the measurement update Gaussian and we update the mean and covariance by adding the mean and the covariance of the motion Gaussian to their respective counterparts. This resultant Gaussian is then fed into the update measurement as the prior, following the cyclical path shown in Figure II.3 (Bishop, 2006).

The result of applying the Kalman filter can be seen in Figure II.4. This graph shows the measured positions of a cannonball flight. As you can see, the measurements are filled with noise, as each of the measured data points seem to be jumping around sporadically. In this same graph, we can see the result of applying the Kalman filter to this noisy data. The Kalman filter filters out the noisy data captured and the end result is a smoothed curve in the shape of an upside parabola, which is the shape one would expect the path of a cannonball would make. Also in this graph, we can see the true path of the cannonball and notice that the Kalman filter curve is remarkably close to this true path.

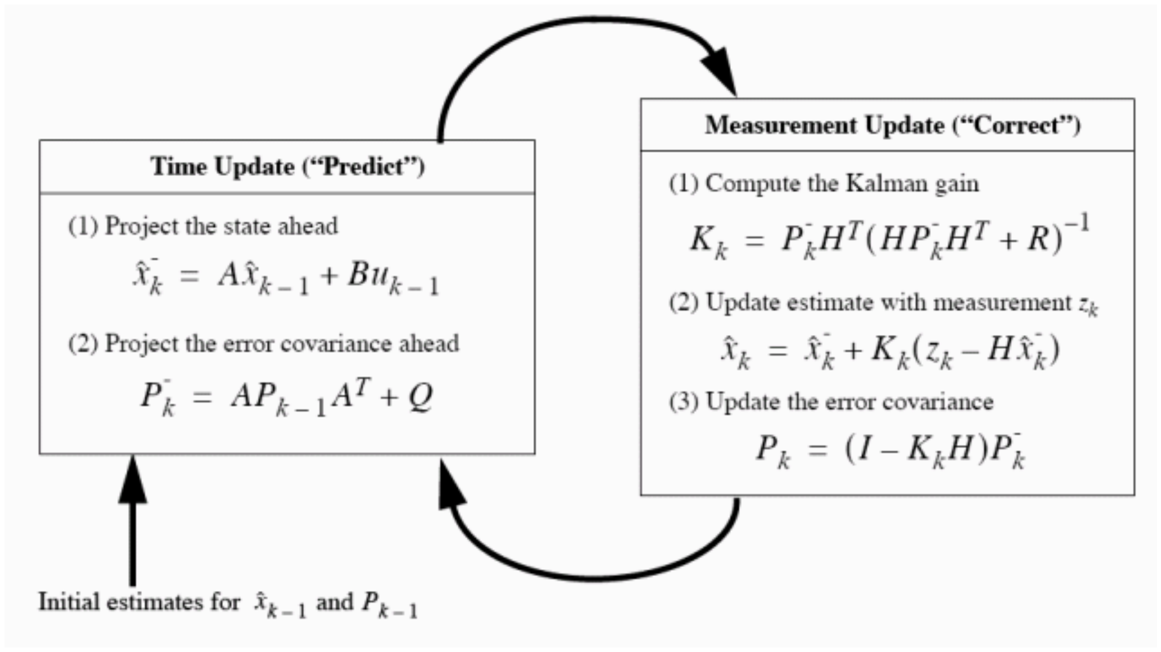
Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$
(2) Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_k H)P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Figure II.3 The measurement and prediction updates of the Kalman Filter (Fisher, 2014)

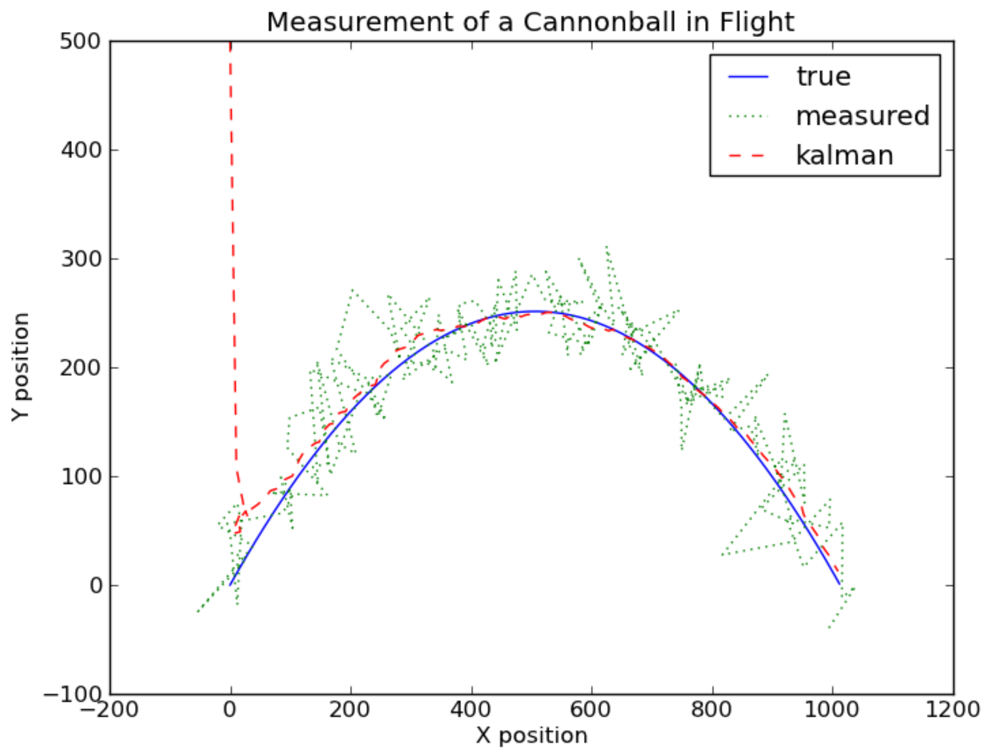

Figure II.4 The Kalman filter applied to a cannon ball flight (Czerniak, 2014)

2.4 Machine Learning

Machine learning is defined as the science of getting computers to act without being specifically programmed. More specifically, machine learning is the automatic discovery of regularities in data through the use of computer algorithms, then making decisions based on these regularities (Bishop, 2006). Within the field of machine learning there are four main subcategories that specify the type of algorithm: learning associations, supervised learning, unsupervised learning, and reinforcement learning. This thesis will focus on the supervised and unsupervised learning methods, as they are used in this project.

Supervised learning is a term used to classify a machine-learning algorithm where the training data is labeled. Training data is a set of data that is input to the adaptive model. If this data is labeled, meaning the ground truth of the data is known, then the algorithm is classified as a supervised learning algorithm. For example, a classic supervised learning algorithm is handwritten digit recognition. Take Figure II.5 below, where each of the digits one through nine is handwritten. The job of the supervised learning algorithm is to take one of these handwritten digits as input and output the identity of the digit. To do this, we must build an adaptive model that takes as input a handwritten digit and outputs the identity of this digit. To tune the parameters of this adaptive model, we must use a training set. The training set will consist of a large set of n handwritten digits $\{x_1, x_2, \ldots, x_n\}$. Each of these digits in the training set will be labeled with its ground truth, which classifies this as a supervised learning algorithm. The adaptive model is then trained on the training data, which is known as the learning phase. Once the adaptive model is trained it can then determine the identity of new digits (Harrington, 2012).

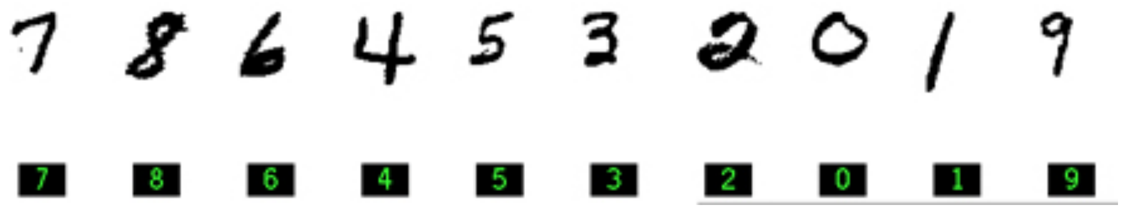

Figure II.5 The mapping of handwritten digits to their digital representations (Image and Documents Analytics, 2014)

Unsupervised learning is a term used to classify a machine-learning algorithm where the training data is not labeled. One of the most common unsupervised learning methods is *k*-Means clustering. In *k*-Means clustering, the goal is to partition the data into *k* clusters. In this algorithm, the examples given to the adaptive model are unlabeled, making this an unsupervised learning method. Instead, the *k*-Means clustering algorithm tries to find a hidden structure in the unlabeled data. *K*-Means accomplishes this by trying to find the minimum distance from the centers of the *k* clusters to each of the data points. In Figure II.6, the left image shows the graph of the unlabeled data, and the right image shows the result of *k*-Means clustering, with *k = 3*. The unlabeled data is split into three clusters based on the minimum distances of points to cluster centers (Harrington, 2012).

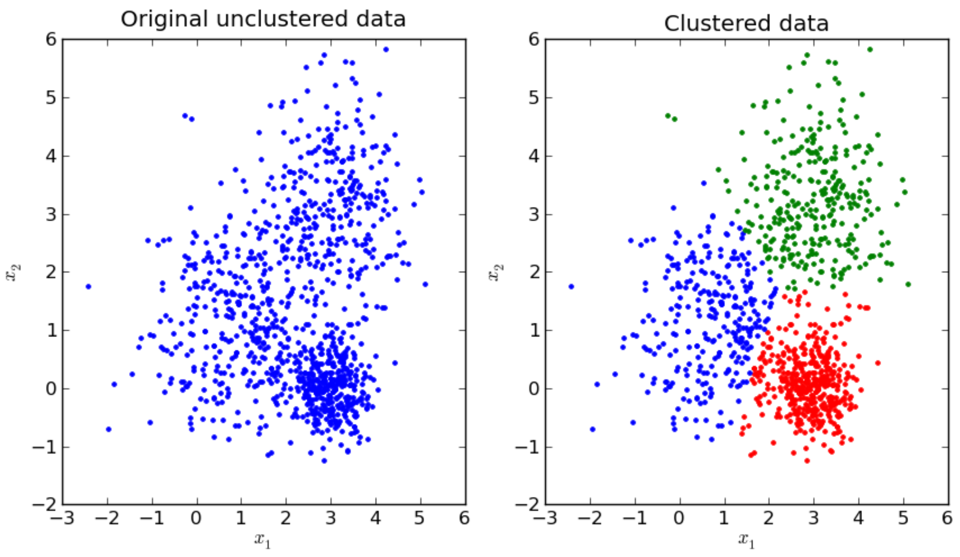

Figure II.6 The result of running *k*-means on unlabeled data (K-Means, 2014)

Support Vector Machines is a typical supervised learning algorithm, and it will be discussed in detail here as this project makes use of this algorithm for sketch classification.

Support Vector Machines work under the assumption that data is linearly separable. If the data meets this constraint, then the support vector machine separates this data by drawing a hyper plane to separate this data. In a simple two-dimensional example, this hyper plane would be a line, and in a three-dimensional example this hyper plane would be a plane. Generally speaking, for data containing of *n* dimensions, we need *n-1* dimensions to separate the data. This hyper plane is known as the decision boundary, meaning everything on one side of the plane belongs to one class and everything on the other side of the plane belongs to a different class. The idea of a hyper plane raises the issue of the optimal way to draw it to separate the data. In Figure II.7, we have two clusters of data: blue squares and red circles. However, as this figure demonstrates, there are a number of ways to draw a hyper plane to separate the blue squares from the red circles.

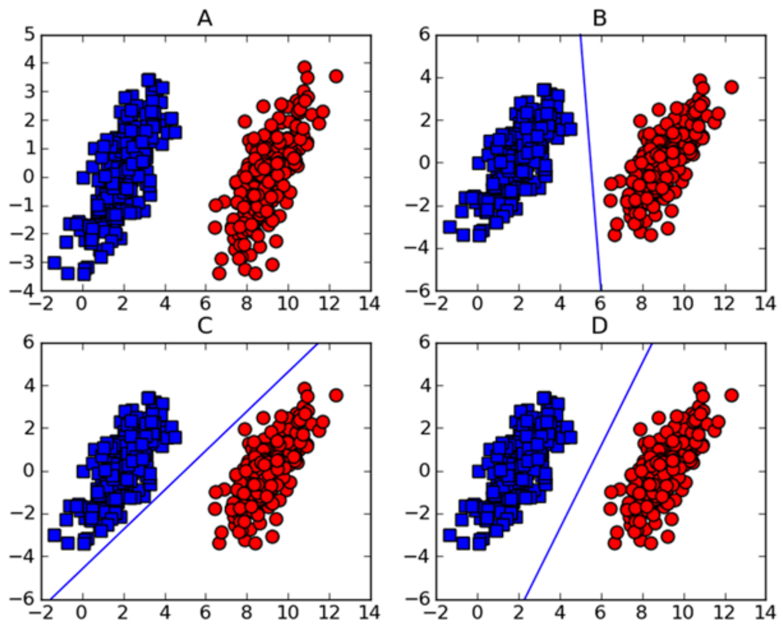

Figure II.7 A demonstration of the numerous ways the hyper plane can be drawn (Harrington, 2012)

The Support Vector Machines algorithm draws the optimal hyper plane to separate the data. This is done by finding the closest points to the hyper plane, called the support vectors, and maximizing the distance between these support vectors and the hyper plane. This distance is known as the margin. In Figure II.8 the left image illustrates the support vectors in relation to the hyper plane. The image on the right of the figure demonstrates the distance between a particular support vector and the hyper plane, known as the margin.

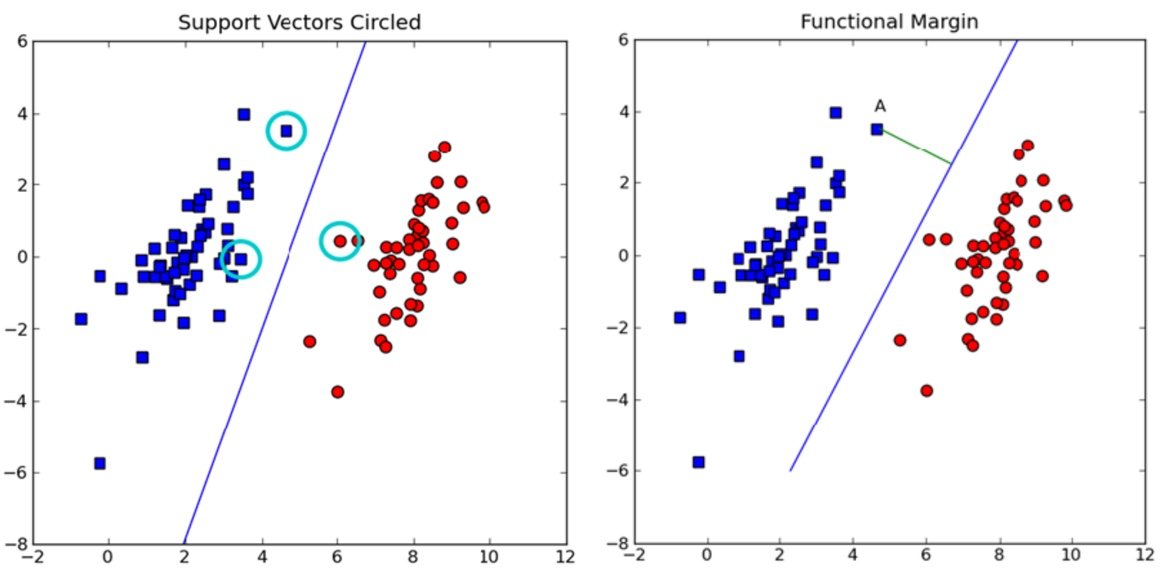

Figure II.8 A visual representation of the support vectors (left) and the margin (right) (Harrington, 2012)

As stated earlier, Support Vector Machines operate under the assumption that the data is linearly separable. However, if the data is not linearly separable, a kernel is used in conjunction with the Support Vector Machines algorithm. A kernel is used to map the feature space from a lower dimension to a higher dimension feature space. This mapping allows nonlinearly separable data in a lower dimension to be solved in a higher dimension that is linearly separable. Figure II.9 demonstrates data that is not linearly separable in a two-dimension feature space, but is linearly separable in a three-dimension feature space.

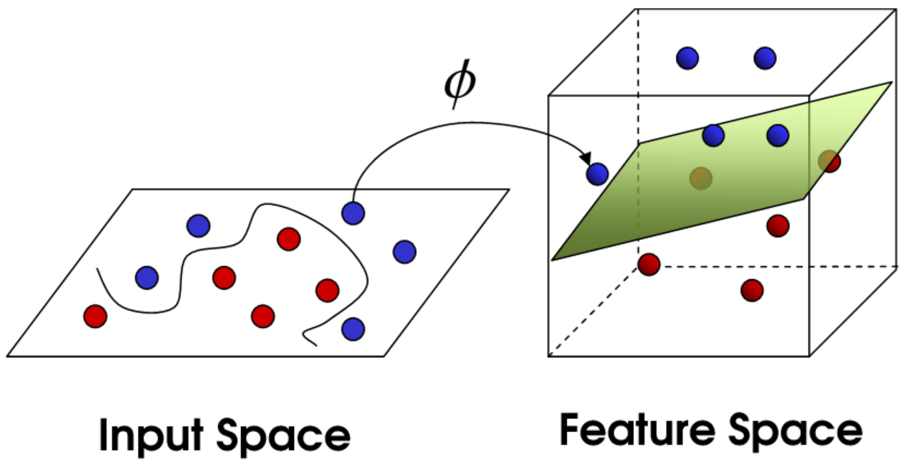

Figure II.9 A mapping from a lower dimension to a higher dimension feature space

Support Vector Machines have many benefits that make it ideal for this project. In particular, the support vector machine algorithm has a low generalization error. It is computationally inexpensive and easy to interpret the results when compared to other classification algorithms. However, some disadvantages of using Support Vector Machines include that it only natively handles binary classification, and is sensitive to tuning parameters and kernel choice. However, each of these limitations can be overcome. Specifically, in order to handle multi-class classification, the techniques of *one-versus-one* or *one-versus-all* can be employed. To help in selecting optimal tuning parameters a grid search can be performed (Harrington, 2012).

CHAPTER III

RELATED WORK

The subject of three-dimensional sketch-based recognition is a new and exciting field. With technology such as the Kinect, more and more people are able to explore the field of three-dimensional computer vision for a reasonable cost. However, since this field and technology is cutting-edge, little preliminary work exists in this domain. Previous work does exist relating to two-dimensional sketch-based recognition, and in this section we will be exploring a few of these works. The first method involves measuring the geometric similarities between a sketch and some ground truth image. The second method is more cutting-edge, as it was published only two years ago, and involves attempting to achieve a semantic understanding of the sketch. After these methods are explored, we will then look at another piece of work that deals with constructing a histogram of oriented gradients for a three-dimensional model, which is useful for feature extraction.

3.1 Geometric Similarity Sketch-based Recognition

The idea of measuring the geometric similarities is by far the most popular method of sketch-based recognition existing today. There have been numerous studies on this topic and this section will discuss a few of them.

In 2005, Chalechale et al. introduced a method for sketch-based retrieval using angular partitioning. In this method, ground truth images are first converted to gray scale and then Canny edge detection is performed to identify strong edges of the image. This method operates under the assumption that the edge maps of the ground truth images are closer in similarity to the sketch-based images. Then each image (the sketch-based and the ground truth) is angularly partitioned into *k* slices. Figure III.1 below shows the angular partitioning of *k* slices (left) as well as the angular partitioning of a sketch-based image (right). The number of edges in each slice is then used as a feature of the image. The similarities of the final feature vectors of the sketch-based image and the ground truth image are then measured using Manhattan distance of the two vectors.

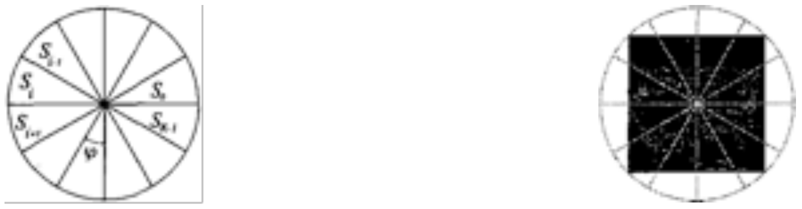

Figure III.1 Angular partition of *k* slices (left) and the effect of angular partitioning on a sketch (right)

In 2011, Shrivastava et al. introduced a method to compare visually similar images across different visual domains. For example, comparing a sketch to a ground truth image, or another example is comparing a painting to a sketch. This approach also measures the geometric similarities of the different images. First a method is introduced that estimates the relative importance of different features in the query image based upon the notion of data-driven uniqueness. The basic idea behind the approach is that the features of an image that exhibit high uniqueness would also be the ideal features to best discriminate this image against the rest of the data. The features are then extracted using a histogram of oriented gradient.

While both works previously discussed displayed positive results in mapping a two-dimensional sketch to a ground truth image, they both relied on the geometry of the sketch, which works great if the sketch is drawn perfectly to scale. However, most humans do not draw perfectly to scale and, instead, tend to draw in a more abstract way while exaggerating certain features. For example, in Figure III.2 we see a user sketch of a bunny rabbit. Even though this sketch is easy to decipher to the human brain, either of the above algorithms would have trouble with this image because the rabbit is not drawn to scale, seen specifically in the oversized ears and whiskers (Eitz, 2012).

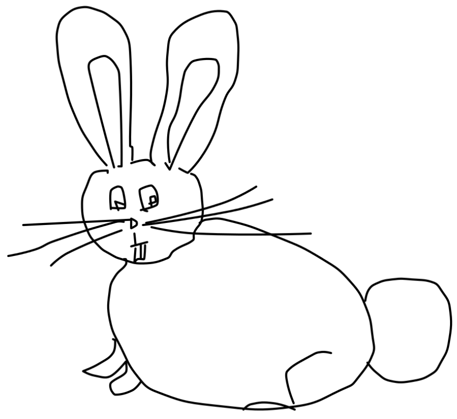

Figure III.2 A sketch of a bunny rabbit with exaggerated features (Eitz, 2012)

Several image synthesis systems exist to allow users to create novel, realistic imagery using sketch exemplars to combat this issue of exaggerated features and scaling (Eitz et al., 2011; Chen et al., 2009; Lee et al., 2011). While these systems help with the accuracy of the above algorithms, lines must still be drawn to scale, and, thus, do little to help users who are incapable of doing this.

3.2 Semantic Understanding Sketch-based Recognition

In 2012, Eitz et al. made one of the first attempts to develop an algorithm that actually understands the semantic meaning of a sketch, as opposed to just geometric similarity. If an algorithm were able to understand the semantics of a sketch, then the problem presented earlier of having to draw exactly to scale would be irrelevant.

In this work the authors first attempted to define an exhaustive set of object categories for sketches to be placed in. The object categories must be recognizable from their shape alone, yet also specific enough to not have any major subcategories. For example, the category of animal is too broad because it contains many subcategories, but the category of dog is more specific and, therefore, appropriate. After defining a taxonomy of 250 object categories, the project then collected 20,000 human sketches, asking each human to sketch an object of a particular category. Then each sketch is represented as a large number of local features that encode local orientation estimates. Each of the locally computed gradients is then assigned to a bin based on its orientation angle to create a histogram of oriented gradients. Figure III.3 shows an example of an image divided into blocks, with each block containing a gradient. As the figure shows, these gradients are then binned according to their orientation angle. This figure shows an example of eight bins. Usually only four are used, however, with co-terminal angles below the x-axis being assigned to the same bin as their counterparts above the x-axis.

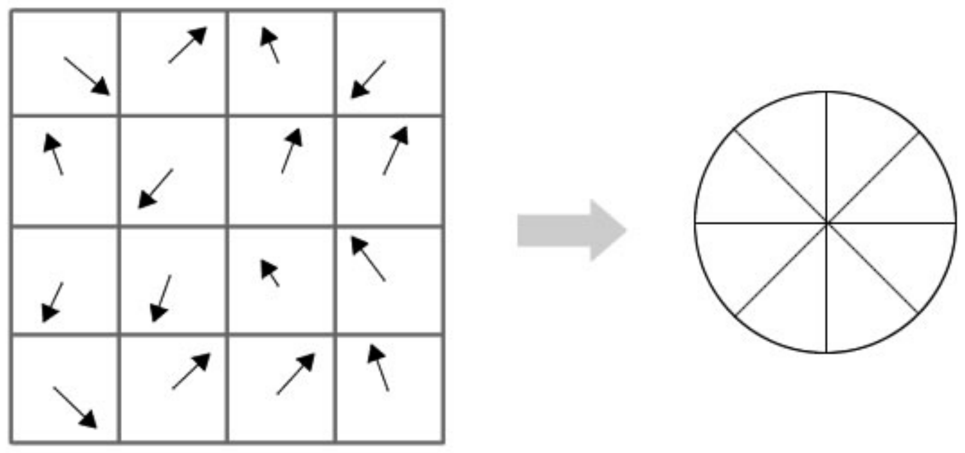

Figure III.3 Image gradients binned according to orientation (Zang, 2014)

After the features were extracted using the histogram of oriented gradients technique above, a visual vocabulary using local features randomly sampled from the dataset of sketches is built using *k*-means clustering. Finally, each sketch is represented as a frequency histogram of visual words.

A number of unsupervised machine learning algorithms were used in the results of this project. Figure III.4 shows how each unsupervised algorithm performed, showing the Support Vector Machines algorithm, utilizing a radial basis kernel, outperforming the other algorithms with 56% accuracy.

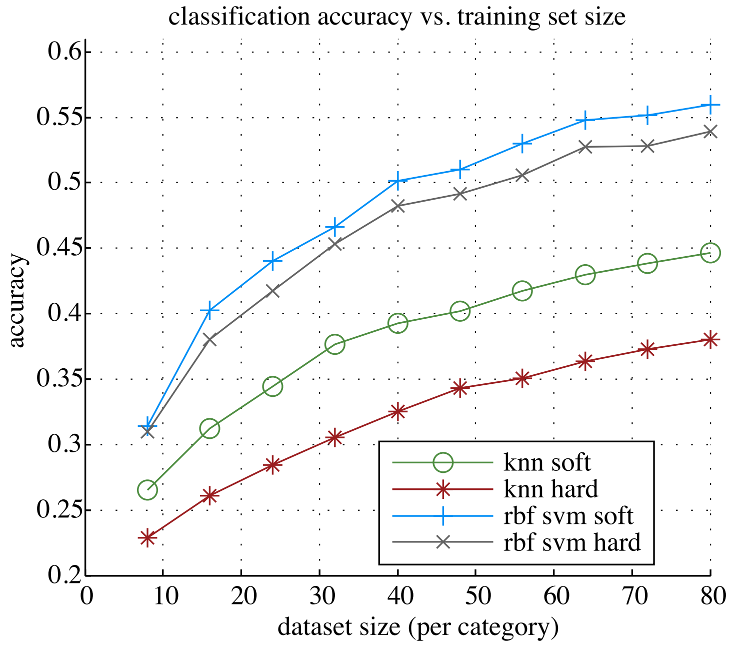

Figure III.4 Accuracies of the unsupervised learning algorithms (Eitz, 2012)

3.3 Three-Dimensional Histograms of Oriented Gradients

While histograms of oriented gradients are fairly straightforward on a two-dimensional image, they can become very complex when trying to extrapolate this idea to the third dimension. The next section does not involve any research in sketch-based recognition and instead focuses on how to construct a histogram of oriented-gradients for a three-dimensional model, which this project relies on to perform feature extraction.

In 2012, Scherer et al. introduces the concept of a three-dimensional histogram of oriented-gradients by adapting the current two-dimensional algorithm. Even though this algorithm is being extrapolated to the third-dimension, the key concepts are still the same: we need to extract gradients from the three-dimensional model, and secondly, we need to organize these gradients into bins using appropriate histograms over uniformly spaced grid cells.

The calculation of the image gradient involves the convolution of the image with a one-dimensional *[-1, 0, 1]* filter mask. This approximates the partial first-order derivate:

$$\begin{aligned} \nabla f(x,y) &= \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)^T \\ &\approx \begin{pmatrix} f(x+1,y) - f(x-1,y) \\ f(x,y+1) - f(x,y-1) \end{pmatrix} \end{aligned}$$

To extend this to the third-dimension, a three-dimensional Euclidean distance field is calculated based on the following formula:

$$f(x, y, z) = min \; || \; x - center(x, y, z) \; ||_2$$

This formula is simply calculating the minimum distance from the center of each grid cell to the boundaries of the cell. Once this distance field is calculated, the gradient can be carried out analogously to the gradient computation on images by convolving the distance field with the *[-1, 0, 1]* filter mask in three-dimensions.

The second step of organizing the gradients into bins using appropriate histograms involves the conversion of the x, y, and z coordinates into spherical coordinates using the following equations:

$$\begin{pmatrix} \theta \\ \phi \\ r \end{pmatrix} = \begin{pmatrix} \arccos \frac{z}{\sqrt{x^2+y^2+z^2}} \\ \operatorname{atan2}(x, y) \\ \sqrt{x^2+y^2+z^2} \end{pmatrix}$$

This results in a two-dimensional histogram of oriented gradients as opposed to the usual one-dimensional, as we are now binning the gradient based on two values (phi and theta) as opposed to just one. Figure III.5 illustrates the formula graphical representation of each of these values in relation to the Cartesian coordinate system.

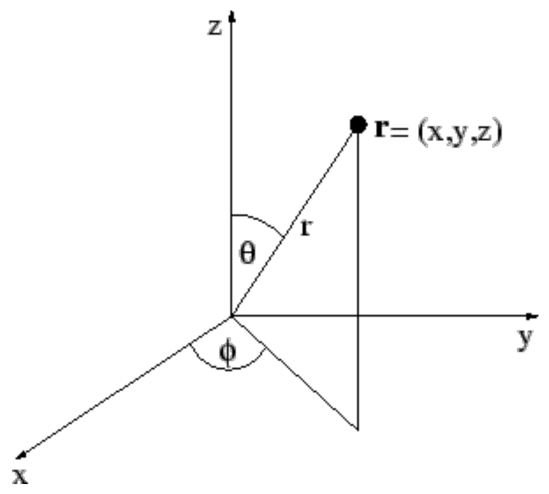

Figure III.5 A graphical representation of the spherical coordinate system

CHAPTER IV

THREE-DIMENSIONAL SKETCH-BASED RECOGNITION

There have been a fair number of studies over the years analyzing the results of numerous two-dimensional sketch-based algorithms. However, to the best of our knowledge, there doesn't exist a study on the results of three-dimensional sketch-based algorithms. Specifically, it is often difficult to display depth information in a two-dimensional sketch and it is even harder for an algorithm to successfully deduce this information on a two-dimensional plane. The question of whether incorporating the third dimension of a sketch can yield an improvement on categorization accuracy has not been answered. Therefore, to answer this question, we much first construct a software program that allows a user to draw a three-dimensional model. Secondly, we must come up with a novel algorithm for feature extraction and object categorization.

4.1 Framework

In order to explore the relationship of depth information in sketch recognition, all the three-dimensional data for each particular category is first collected from a user study and then thoroughly analyzed for feature extraction. Different from traditional two-dimensional methods for feature extraction, a new learning-based approach is proposed for feature extraction and object categorization in the third dimension.

The learning framework is demonstrated in Figure IV.1. This figure segments the learning algorithm into two main stages: the training stage and the testing stage. As discussed in section 2.4, during the training stage of the Support Vector Machines algorithm, we are trying to learn the parameters *w* and *b* (from the equation $w^T x + b$), such that the margin between the support vectors and the hyper plane is as large as possible. The training stage then produces a predictive model that can be used in the testing stage. In the testing stage, given a new three-dimensional sketch, we are able to input its feature vectors into the predictive model to output the expected category of the sketch.

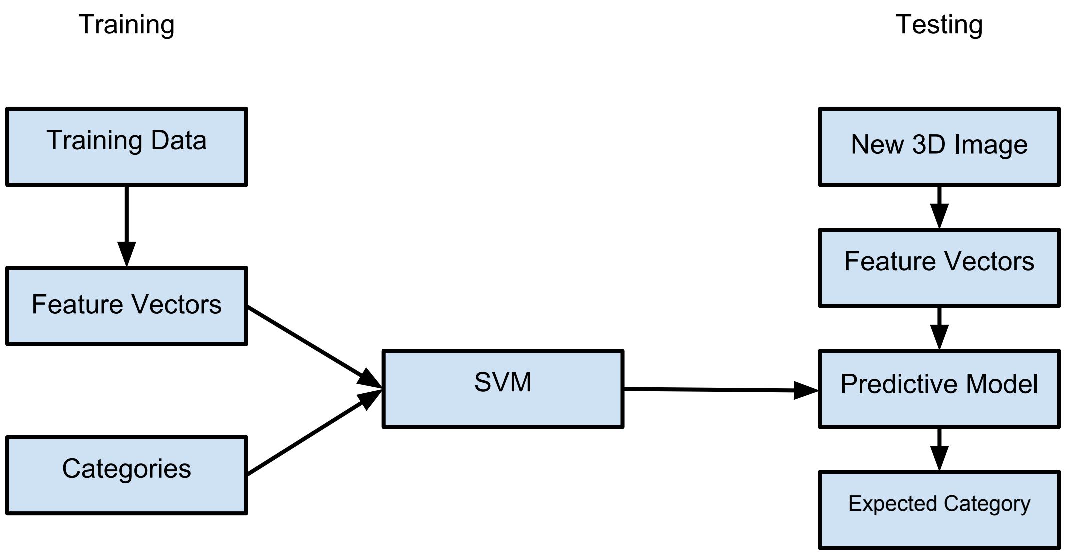

Figure IV.1 Learning framework flowchart

4.2 Three-Dimensional Data Collection

4.2.1 Three-Dimensional Drawing Board Design

In order to collect the three-dimensional data from users, we first had to develop a three-dimensional drawing board to allow user's to draw in a three-dimensional space. The newly designed software allows the user to draw an object in the three-dimensional space around them, while tracking the hand movements of the user, storing the x, y, and z coordinates of each hand movement. This interface also supports voice commands, allowing a user to pause during the middle of a drawing, or completely restart a sketch using only their voice.

To accomplish the task of building such a complex system, the Microsoft Kinect was used. The Microsoft Kinect contains a RGB video camera, allowing the user to see exactly what they are drawing in the surrounding three-dimensional space, as well as a depth sensor to help capture the depth information of each sketch. The Kinect comes with a software development kit, which contains APIs for hand tracking as well as voice recognition, allowing for the development of our system.

Figure IV.2 shows the screenshot of the user interface developed for this project. The user interface was built using Qt, a cross platform application and UI framework. The user interface allows the user to tell the system which hand they will be drawing with (the right or left), as well as shows the user the state of their current drawing. The user can interact with the application using voice commands as well. The application allows for the user to say ‘pause’ to stop recording data, as well as ‘restart’ to start the sketch over.

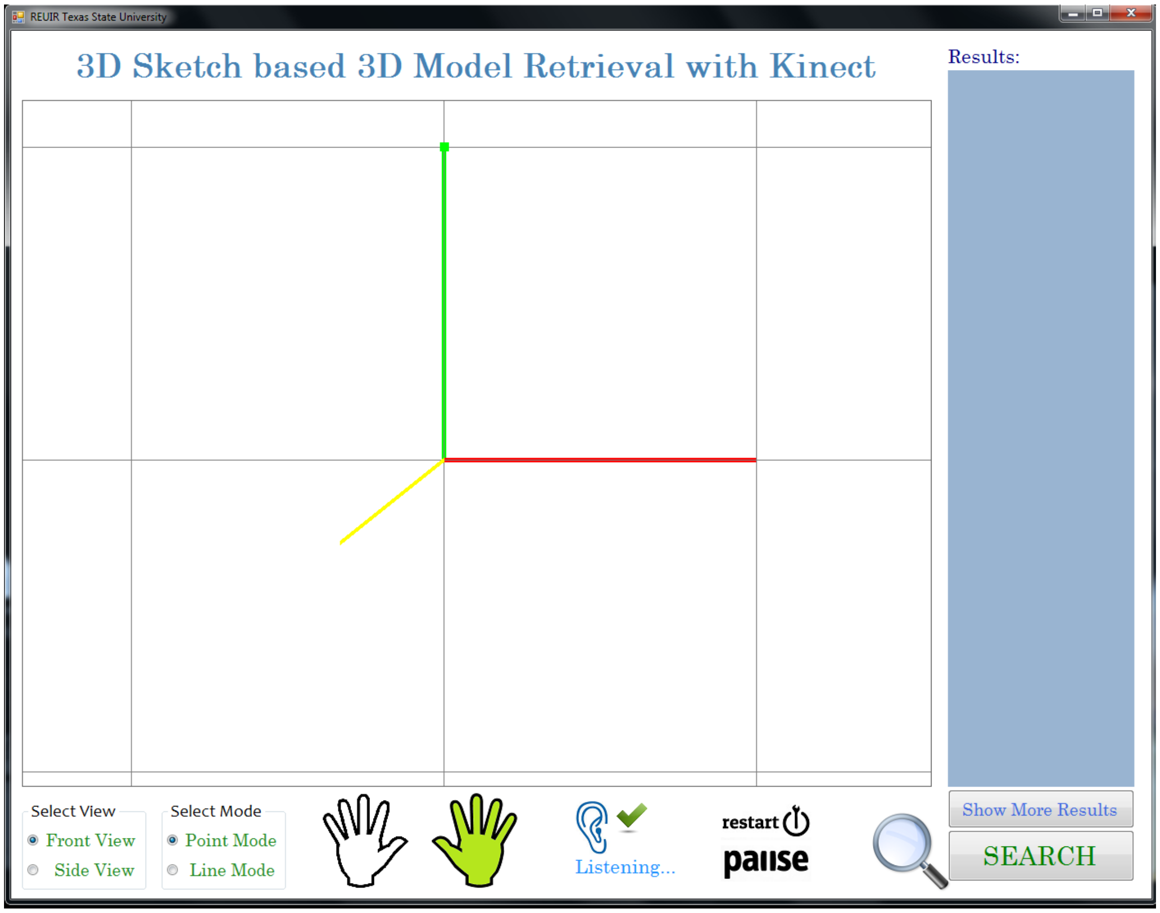

Figure IV.2 3D sketch user interface

4.2.2 Object Categorization Data Set

A subset of the object categories in the data set used in the work of Eitz et al. (2012) was selected for this experiment. The decision of choosing only a subset of the object categories in the previous work was twofold. First, to have 250 categories of objects, over 20,000 sketches from numerous users were collected for the training data phase. This study didn't have the time and resources to obtain that many sketches. Second, the subset of categories was handpicked where depth information would actually help categorize the object. Therefore, the following object categories were selected: basket, cup, megaphone, suitcase, lamp, mailbox, and house.

4.2.3 User Study

This study consisted of 10 users, where each user drew a sketch of each one of the object categories, resulting in the collection of 70 three-dimensional sketches. The users consisted of four females and six males with an average age of 32 years old. In order to protect a participant's confidentiality, each was assigned a number. All data collected during this study was recorded using this user number, not by name.

Before the start of each data collection session with a new user, the user was allowed to sketch a few practice drawings to help accustom themselves with the drawing rules of the application as well as drawing in a three-dimensional space. After a brief period of training time, the user was asked to sketch each one of the predefined categories. The order of the sketches each user had to draw was predetermined, with easier sketches being presented first, and harder ones being presented last. The hope here was that as time went on and the user became more accustomed to drawing in a three-dimensional space, the more complex images would become easier to sketch. While the user is drawing, the program tracks the user's hand: recording the x, y, and z coordinates of each movement. Figure IV.3 demonstrates the process of data collection, showing a subject interacting with the software to draw their sketch.

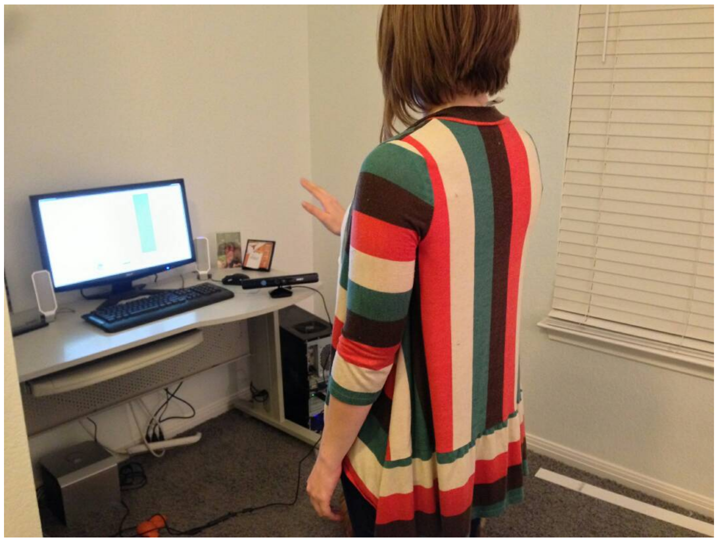

Figure IV.3 Data collection process

4.3 Three-Dimensional Model Feature Extraction

The process of feature extraction involves transforming the input data into a reduced representation of a set of features. These features should be similar between objects from the same category, while also being different from objects of different categories. The feature extraction code is arguably the most important algorithm of this project, as the machine-learning algorithm uses these feature vectors to differentiate objects from different categories.

Figure IV.4 outlines the algorithm of the feature extraction code used in this project. First, the x, y, and z coordinates of a model are loaded into a three-dimensional matrix. Then, a weighted PCA analysis is performed on the data set, making it rotation invariant. The coordinates are then normalized inside a bounding cube with dimensions *n x n x n*. Voxelization is then performed on the data points, assigning a value of 1 if a point exists there, and a 0 if it does not: creating a binary model of the object. The gradients in the x, y, and z directions are then computed on this binary model using the following equations:

$$\begin{aligned} \nabla f(x,y) &= (\partial f / \partial x,\ \partial f / \partial y,\ \partial f / \partial z) \\ &= f(x+1, y, z) - f(x-1, y, z) \\ &\quad\ f(x, y+1, z) - f(x, y-1, z) \\ &\quad\ f(x, y, z+1) - f(x, y, z-1) \end{aligned}$$

The gradients are then converted into spherical coordinates:

$$\begin{pmatrix} \theta \\ \phi \\ r \end{pmatrix} = \begin{pmatrix} \arccos \frac{z}{\sqrt{x^2+y^2+z^2}} \\ \operatorname{atan2}(x,y) \\ \sqrt{x^2+y^2+z^2} \end{pmatrix}$$

Then a two-dimensional histogram is created for each cell of the model, based on the values of theta and phi. Each of these histograms is then concatenated together to form block vectors. Finally, each block vector is then normalized and concatenated together to form the final feature vector of the model.

```
1.  Choose size of cell and block
2.  FOREACH Model $m$ DO
3.      Perform PCA analysis
4.      Normalize coordinates inside bounding box
5.      Perform voxelization of the coordinates in a 3D matrix
6.      GradientField $\mathbf{G} = \mathrm{conv3}(\mathbf{D}, [-1, 0, 1])$
7.      FOREACH Gradient $\boldsymbol{g}$ IN $\mathbf{G}$ DO
8.          transformToSphericalCoordinates( $\boldsymbol{g}$ )
9.          insert $\boldsymbol{g}$ into its CellHistogram $h_c(\theta, \varphi)$
10.     FOREACH CellHistogram $h_c$ DO
11.         append $h_c$ to its BlockVector b
12.     FOREACH BlockVector b DO
13.         append normalize(b) to featureVector
14.     save featureVector
```

Figure IV.4 Feature extraction pseudo code (Scherer, 2010)

4.4 Three-Dimensional Sketch Learning

This section will discuss the machine-learning method used in this project on the features we extracted in the previous section. Specifically, we will look at the Support Vector Machines algorithm and the kernel used in conjunction with it.

The Support Vector Machines algorithm takes as input a feature vector of a particular model, along with the label of the model. The feature vector is a concatenation of the histogram of oriented gradients for each block of the model. The feature dimension of the final feature variable is dependent on certain parameters that will be tuned in Chapter V, such as the cell and block size chosen. Letting the variable *blocks* represent the total number of blocks, *cells* the number of cells per block, and *bins* the number of bins per histogram, the feature dimension can be calculated using the following formula:

$$Feature\ Dimension = blocks * cells * bins$$

The Support Vector Machines algorithm then trains on a subset of the data collected to build a predictive model. In the training phase of the learning algorithm, the three-dimensional model and the ground truth (the category to which the model belongs) are used as input. In order to prevent the problem of creating a model that over fits the training data, *k*-fold cross validation is used. In *k*-fold cross validation, the model is trained using *k-1* of the folds as training data and the resulting model is then validated on the remaining part of the data. Figure IV.5 shows a visual representation of this process. In this example, we are letting *k* be equal to the number of observations in the original sampling. A single observation from the original sample is used as the validation data and the remaining observations are used as the training data. This process is then repeated for *k* cycles. This particular kind of cross validation is also known as leave-one-out cross validation and is useful when only small data sets are available.

In order to combat the data not being linearly separable, a kernel is used to map the current feature space to a higher dimension so that it may be linearly separated. In this particular project, a radial basis function kernel is used.

After the training is complete, we now have a predictive model. This model can take as input a new three-dimensional model and it will output the expected category to which the model belongs.

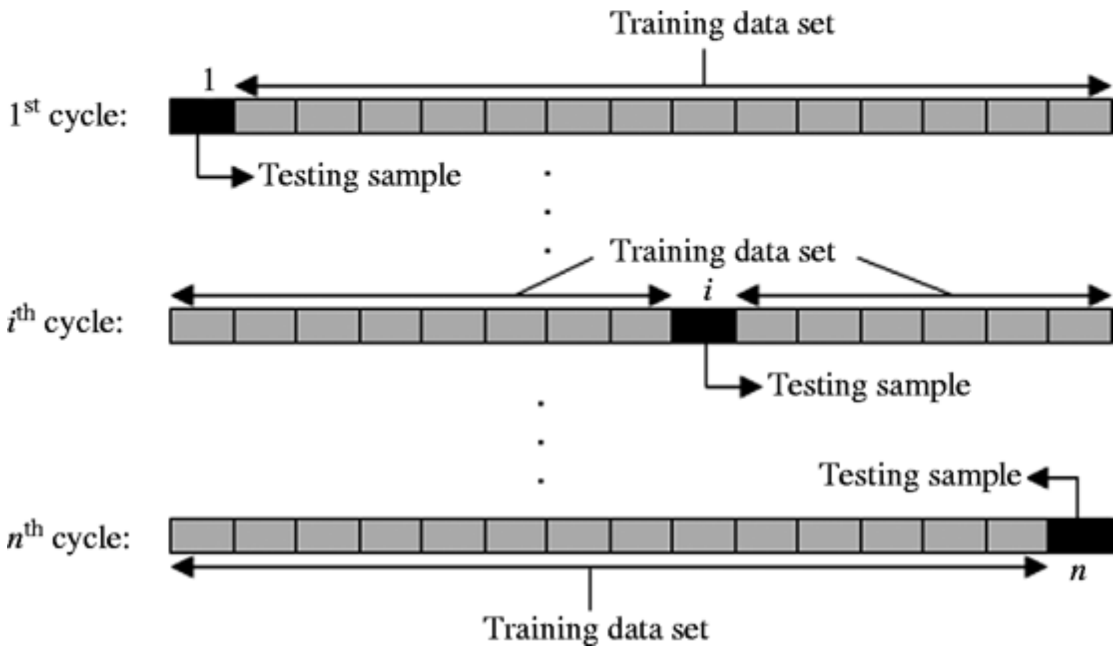

Figure IV.5 Leave-one-out cross validation (Ghosh, 2011)

CHAPTER V

EXPERIMENT AND RESULT

5.1 Experiment Design

In order to evaluate the machine learning algorithm introduced in Chapter IV, the results of this approach were evaluated against the results of running a two-dimensional HOG algorithm against the two-dimensional projections of each sketch. First, a set of categories was selected for the user to sketch. This resulted in these seven categories: cup, megaphone, suitcase, lamp, basket, mailbox, and house. These categories were specifically chosen to allow for good results when incorporating depth information, as well as being simple objects that are easy to draw in a three-dimensional space. The accuracies of the three-dimensional sketches of each of these categories are then compared to the accuracies of the exact same sketches projected onto a two-dimensional plane, demonstrating the importance of incorporating the depth information.

In this experiment, there are a number of parameters that need to be fine-tuned in order to obtain good results. This experiment tests a different number of each of these parameters in order to obtain the best results. In particular, a number of different cells sizes were tested in order to obtain the optimal number of cells to divide the model. Along the same line, a number of different block dimensions were also tested for organizing cells. Finally, different gamma values for the RBF (radial basis function) kernel were tested for the machine learning section of the algorithm.

5.2 Evaluation Metrics

In order to evaluate the performance of the three-dimensional sketch recognition model a confusion matrix is used. A confusion matrix is a table layout that allows for the visualization of the performance of this algorithm. Figure V.1 shows an example of a confusion matrix. Here, each cell of the matrix is labeled as true positive, false positive, false negative, true negative. The true positive cell contains the number of correctly predicted labeled models. The false positive cell contains the number of models that were falsely labeled as this category. The false negative cell contains that number of models that are of this particular category, but incorrectly labeled as a different category. The true negative cell contains the number of models that were correctly predicted as not belonging to this particular category.

	p' (Predicted)	n' (Predicted)
p (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

Figure V.1 An example of a confusion matrix (Caraciolo, 2014)

5.3 Experiment Results

5.3.1 Bounding Box Dimensions

In order to build the optimal sketch recognition program, the first variable that was fine-tuned was the bounding box dimensions. In the feature extraction code, the very first step is to encapsulate the model inside a bounding box of particular dimensions to ensure the features extracted from all models are consistent. Figure V.2 illustrates the accuracies of using different dimensions. As the graph shows, using a block dimension of *64 x 64 x 64* seems to work the best.

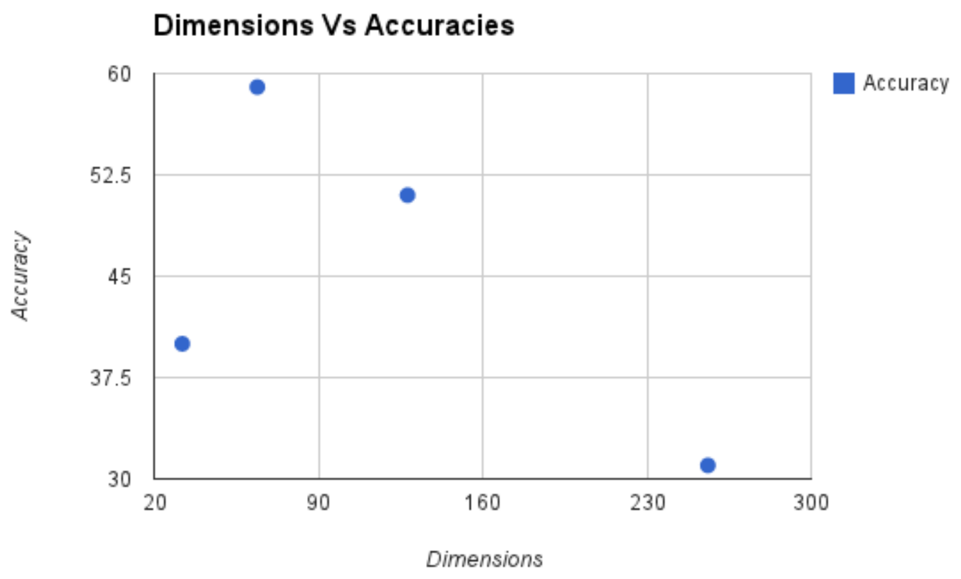

Figure V.2 The accuracies of the various bounding box dimensions

5.3.2 Gamma Parameter

The second variable to be fine-tuned was the gamma parameter for the RBF kernel. In order to find the optimal value for gamma, a grid search was performed. A grid search is used for parameters that are not directly learnt within a classifier. Instead these parameters can be set by searching a parameter space for the best cross validation score based on some scoring metric. In this particular experiment the scoring metric of accuracy was used, but there are other metrics that can be used based on the needs of the experiment, such as precision, recall, etc. Figure V.3 illustrates the accuracies of using various gamma parameters for the RBF kernel. As the graph illustrates, a gamma value of 0.0001 gives the highest accuracy with 59%.

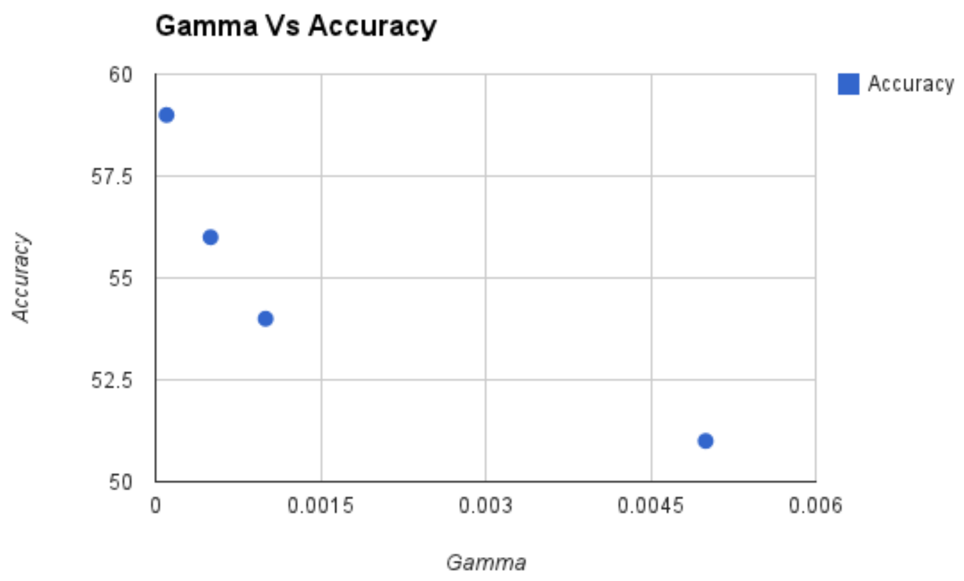

Figure V.3 The accuracies of various gamma values

5.3.3 Confusion Matrix

After learning the optimal values for the bounding box dimensions and the gamma parameter, next a confusion matrix is constructed, showing the performance of the algorithm in regards to classifying each object to the correct category. In Table 1 we see the confusion matrix for all seven categories. The matrix, *C*, can be evaluated as entry $C_{i,j}$ is equal to the number of observations known to be in category *i* but predicted to be in group *j*.

Table 1 Confusion matrix of each category

	Basket	Cup	House	Lamp	Mailbox	Megaphone	Suitcase
Basket	9	0	0	0	0	1	0
Cup	1	4	1	2	0	2	0
House	0	1	4	1	0	1	3
Lamp	1	2	0	6	0	1	0
Mailbox	0	3	0	0	6	0	1
Megaphone	0	3	0	0	0	7	0
Suitcase	0	0	2	0	2	1	5

5.3.4 Precision, Recall, F-Measure

The precision, recall, and f-measure can be computed from the values of the confusion matrix above. Table 2 below displays these values for each of the specified categories.

Table 2 Precision, Recall, & F-Measure of the categories

	Basket	Cup	House	Lamp	Mailbox	Megaphone	Suitcase
Precision	82%	31%	57%	67%	75%	54%	56%
Recall	90%	40%	40%	60%	60%	70%	50%
F_1 Score	86%	35%	47%	63%	67%	61%	53%

5.3.5 Accuracy

After constructing the confusion matrix for each of the categories, the accuracies for each of the categories can now be calculated. Figure V.4 illustrates the accuracies for each of the respective categories.

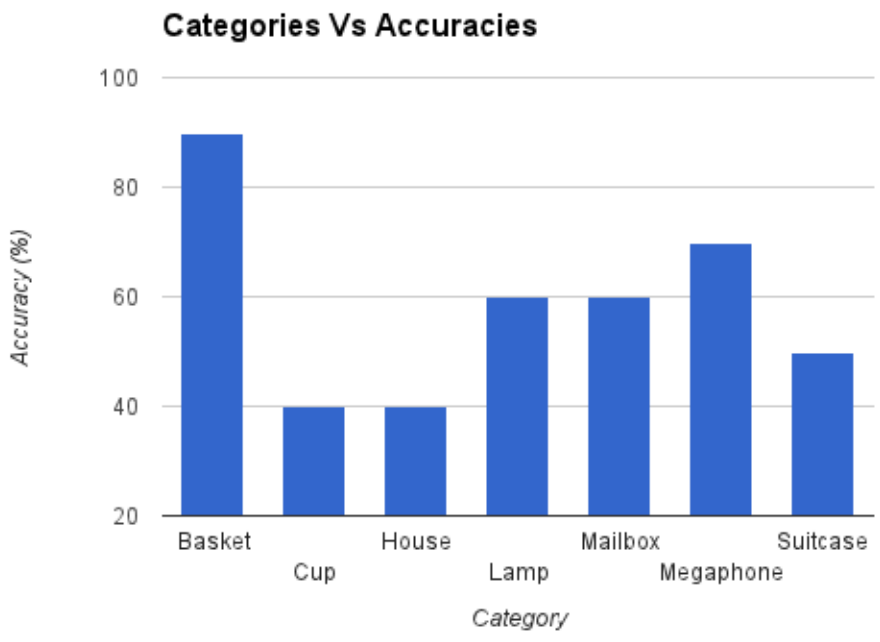

Figure V.4 Accuracies of all the categories

5.3.6 Analysis

In order to analyze the importance of the depth information of each model, this experiment was run again utilizing only the two-dimensional data for each model. Specifically, only the x and y coordinate data for each model were used for feature extraction. Table 3 shows the confusion matrix for the two-dimensional categories and Table 4 shows the precision, recall, and f-measure of each category.

Table 3 Confusion matrix of the two-dimensional categories

	Basket	**Cup**	**House**	**Lamp**	**Mailbox**	**Megaphone**	**Suitcase**
Basket	6	0	0	0	1	3	0
Cup	1	0	3	3	1	1	1
House	3	2	1	0	0	2	2
Lamp	2	4	2	0	0	1	1
Mailbox	1	0	0	0	4	2	3
Megaphone	1	0	1	0	0	8	0
Suitcase	0	1	1	2	2	0	4

Table 4 Precision, Recall, & F-Measure of the two-dimensional categories

	Basket	Cup	House	Lamp	Mailbox	Megaphone	Suitcase
Precision	43%	0%	12.5%	0%	50%	47%	36%
Recall	60%	0%	10%	0%	40%	80%	40%
F_1 Score	50%	0%	11%	0%	44%	59%	38%

These tables, when compared to Table 1 and Table 2, illustrate the importance of the third dimension as a feature. Specifically, the three-dimensional models outperformed all of the two-dimensional sketches in terms of precision and all but one of the two-dimensional sketches in terms of recall. The overall accuracy of the two-dimensional sketches was 33%, compared to the 59% we get from the three-dimensional sketches. Figure V.5 further solidifies the importance of using the depth information of sketches, showing the accuracy comparison of each category for the three-dimensional and two-dimensional sketches.

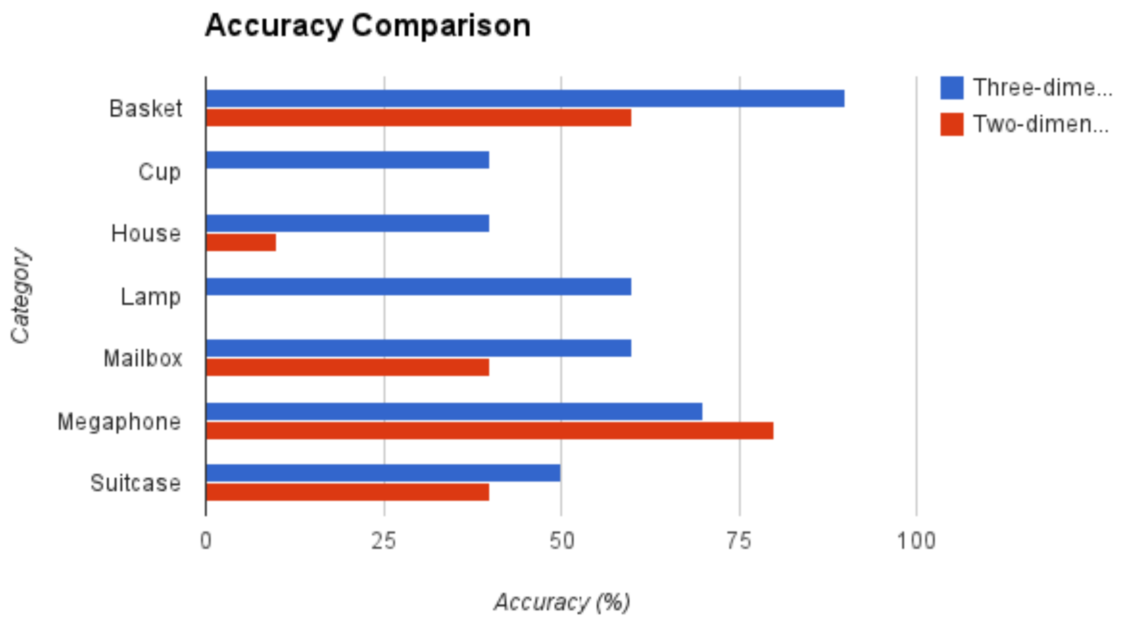

Figure V.5 Accuracy comparison of three-dimensional vs. two-dimensional sketches

CHAPTER VI

CONCLUSION

The area of sketch-based recognition has become a very popular research topic in recent years due to its vast applications. However, all of these previous approaches require the user to sketch on a two-dimensional plane, losing depth information that could potentially aid in sketch recognition. Recently, with technologies such as the Microsoft Kinect, three-dimensional image capture has become more affordable and accessible; making the idea of three-dimensional sketch-based recognition a reality.

In this thesis, a novel approach to sketch recognition was explored by utilizing the third dimension. First, software was developed to allow for a user to draw three-dimensional sketches. Specifically, a visual drawing board was developed using the Microsoft Kinect and drawing rules were established to allow a user to draw a sketch in a three-dimensional space. This information was then utilized to form a learning-based algorithm to categorize three-dimensional sketches into specific categories.

The results of this experiment demonstrate that incorporating the depth information of a sketch can lead to an improvement in the overall accuracy of classifying a sketch. In particular, this algorithm was able to successfully categorize 59% of a set of individual categories, a vast improvement over the 33% accuracy of the exact same sketches projected onto a two-dimensional plane.

In future work, further improvement on three-dimensional sketch-based recognition may be achieved in the data collection process of the algorithm. In particular a database of only 70 three-dimensional models was used for this study. I believe the accuracy can be improved if the algorithm has a larger dataset to learn from. Also, accuracy may be improved in the simplification of the software and drawing rules presented to capture the data. Specifically, after running these experiments, it is apparent that it is difficult for most people to draw images in a three-dimensional space. Simplification of the drawing rules, along with improvements in three-dimensional capture software, could potentially allow for users to become more comfortable with drawing in a three-dimensional space, as well as allow for the drawing of more complex objects.

REFERENCES

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Caraciolo, M. (n.d.). Artificial Intelligence in Motion. *Tools for Machine Learning Performance Evaluation: Confusion Matrix -*. Retrieved March 6, 2014, from http://aimotion.blogspot.com/2010/08/tools-for-machine-learning-performance.html

Chalechale, A., Naghdy, G., & Mertins, A. (2005). Sketch-Based Image Matching Using Angular Partitioning. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *35*(1), 28-41.

Chen, T., Cheng, M., Tan, P., Shamir, A., & Hu, S. (2009). Sketch2Photo. *ACM Transactions on Graphics*, *28*(5), 1.

Crosstabs and Analysis of Proportions in NCSS. (n.d.). *NCSS Crosstabs and Analysis of Proportions in NCSS Comments*. Retrieved March 6, 2014, from http://www.ncss.com/software/ncss/crosstabs-and-analysis-of-proportions-in-ncss/

Czerniak, G. (n.d.). Greg Czerniak's Website. - *Kalman Filters for Undergrads 1*. Retrieved February 20, 2014, from http://greg.czerniak.info/guides/kalman1/

Eitz, M., Richter, R., Hildebrand, K., Boubekeur, T., & Alexa, M. (2011). Photosketcher: Interactive Sketch-Based Image Synthesis. *IEEE Computer Graphics and Applications*, *31*(6), 56-66.

Eitz, M., Hays, J., & Alexa, M. (2012). How do humans sketch objects?. *ACM Transactions on Graphics*, *31*(4), 1-10.

Eitz, M., Richter, R., Boubekeur, T., Hildebrand, K., & Alexa, M. (2012). Sketch-based shape retrieval. *ACM Transactions on Graphics*, *31*(4), 1-10.

Fisher, R. (n.d.). 1 The Discrete Kalman Filter. *1 The Discrete Kalman Filter*. Retrieved February 20, 2014, from http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/WELCH/kalman.1.html

Ghosh, A., Guha, T., Bhar, R., & Das, S. (2011). Pattern classification of fabric defects using support vector machines. *International Journal of Clothing Science and Technology*, *23*(2/3), 142-151.

Harrington, P. (2012). *Machine learning in action*. Shelter Island, NY: Manning Publications.

Image and Documents Analytics. (n.d.). - *OCR Technologies*. Retrieved February 15, 2014, from http://www.research.ibm.com/haifa/dept/imt/dpm/focus_ocr.shtml

K-Means. (n.d.). *K-Means ‚Äî PyPR v0.1rc3 documentation*. Retrieved February 17, 2014, from http://pypr.sourceforge.net/kmeans.html

Kinect for Windows Sensor Components and Specifications. (n.d.). *Kinect for Windows Sensor Components and Specifications*. Retrieved February 14, 2014, from http://msdn.microsoft.com/en-us/library/jj131033.aspx

Lee, Y. J., Zitnick, C. L., & Cohen, M. F. (2011). ShadowDraw. *ACM Transactions on Graphics*, *30*(4), 1.

Scherer, M., Walter, M., & Schreck, T. (2010). Histograms of Oriented Gradients for 3D Object Retrieval. P*roceedings of the WSCG 2010, Plzen, Czech Republic*, *1*, 41-48.

Shrivastava, A., Malisiewicz, T., Gupta, A., & Efros, A. A. (2011). Data-driven visual similarity for cross-domain image matching. *ACM Transactions on Graphics*, *30*(6), 1.

Zang, A. (n.d.). SIFT: Scale Invariant Feature Transform. *Assignment 3*. Retrieved February 25, 2014, from http://w3.impa.br/~zang/IP2012/sift_p7.html