WORKPLACE POSTURE ASSESSMENT AND BIOFEEDBACK WITH KINECT

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

Matthew Crussell

April 2017

COMMITTEE MEMBERSHIP

TITLE:	Workplace Posture Assessment and Biofeedback with Kinect
AUTHOR:	Matthew Crussell
DATE SUBMITTED:	April 2017
COMMITTEE CHAIR:	Jane Zhang, Ph.D. Professor of Electrical Engineering
COMMITTEE MEMBER:	Xiao-Hua Yu, Ph.D. Professor of Electrical Engineering
COMMITTEE MEMBER:	John A. Saghri, Ph.D. Professor of Electrical Engineering

ABSTRACT

Workplace Posture Assessment and Biofeedback with Kinect

Matthew Crussell

With the prevalence of computing, many workers today are confined to desk within an office. By sitting in these positions for long periods of time, workers are prone to develop one of many musculoskeletal disorders (MSDs), such as carpal tunnel syndrome. In order to prevent MSDs in the long term, workers must employ good sitting habits. One promising method to ensure good workplace posture is through camera monitoring. To date, camera systems have been used in determining posture in a clean environment. However, an occluded and cluttered background, which is typical in an office setting, imposes a great challenge for a computer vision system to detect desired objects. In this thesis, we design and propose components that assess good posture using information gathered from a Microsoft Kinect camera. To do so, we generate a data set of posture captures to test and train, applying crowd-sourced voting to determine ratings for a subset of these captures. Leveraging this data set, we apply machine learning to develop a classification tool. Finally, we explore and compare the usage of depth information in conjunction with a traditional RGB sensor array and present novel implementations of a wrist locating method.

ACKNOWLEDGMENTS

To my late grandfather, Colin Crussell. Though you didn't have the opportunity to see my education to its completion, I hope you are proud of the work I have done. You were a role model to me, and I plan to continue on my path in a way that will make you always proud. And to my family, thank you for supporting me through my studies. Though it took longer than expected, I relied on you all for the emotional support to get through it. Next, I am eternally grateful to Mariah for her support. Thank you for supporting my nights of work, enduring shopping trips cut short, and all the countless things you have done that have gone unnoticed. Finally, thank you to my advisor, Dr. Jane Zhang, for her feedback and assistance along the way.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Maintaining good posture can be an elusive goal for many office workers working behind a desk. Those with little knowledge on proper sitting posture expose themselves to a higher risk of different types of joint pain, forming habits that can be hard to break. Those trying to adopt new habits may easily fall back on old sitting positions with many deadlines and obligations of the work environment. Using joint depth-color data, we investigate components behind a computer vision system to automatically assess postures. In this chapter we define good posture, outline motivations, and describe basic project goals.

1.1 Posture Definition

The Occupational Health and Safety Administration(OSHA) is a branch of the U.S. Department of Labor dedicated to maintaining health and safety within workplaces. The department, enacted by Congress in 1970, is tasked with ensuring regulations are followed, assisting with outreach & training of employees and employers, in addition to promoting health & safety through educational resources [25]. We intend to conform to OSHA's criteria for a good working posture for office workers, highlighting specific sets of joints within the body.

In general, to maintain good posture while sitting, we must keep joints in neutral positions. Any positions requiring bending or continually contorting joints is considered bad posture. Suppose a student is typing an essay on an old laptop with a thick keyboard. If the student wishes to relax his or her arms on the desk while still making contact with the keys, he or she would need to bend at the wrist to reach the keys. As per OSHA's definition of workplace posture, this bending is not a neutral posture [28]. Now for more formal definitions.

According to OSHA, basic posturing can be outlined as follows:

Hands, wrists, and forearms are straight, in-line and roughly parallel to the floor.

Head is level, or bent slightly forward, forward facing, and balanced. Generally it is in-line with the torso.

Shoulders are relaxed and upper arms hang normally at the side of the body.

Elbows stay in close to the body and are bent between 90 and 120 degrees.

Feet are fully supported by the floor or a footrest may be used if the desk height is not adjustable.

As mentioned before, each portion of good posture requires maintaining a neutral position. To provide further example of what constitutes good and bad positions, some figures may be helpful.

1.1.1 Back Posture

(a) Declined (b) Reclined (c) Upright

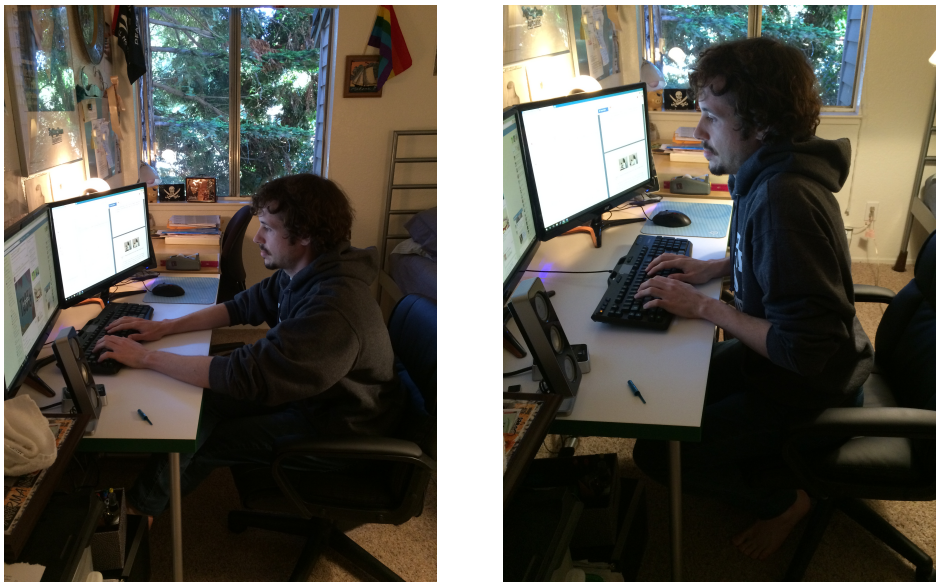
Figure 1.1: Reference sitting postures [28]

In Figure 1.1, we see three representations of good back posture. The classification of good posture is not simply based on a chair angle; good back posture is based on the amount of uniform support. In each image of Figure 1.1, the subject sits with back and legs entirely parallel with their respective positions on the chair. This support must be taken into consideration when evaluating good back posture.

1.1.2 Wrist Posture

Ideally, hands should be parallel to forearms to maintain a neutral wrist alignment. In the case of typing, this means that a worker should not be resting their wrists on a desk. A worker must also consider distance from the desk- by locating oneself too near

or too far, they may force wrist out of neutral alignment. In Figure 1.2, we see two examples of improper chair positioning. Considering distance, a worker should not need to lean against the top of a desk, as in Figure 1.2a, nor crouch over a keyboard, as in Figure 1.2b.



(a) Too Far (b) Too Close

Figure 1.2: Improper distancing from desk

A good distancing allows wrists to relax at a neutral point. Figure 1.3 characterizes a good distancing from the desk.

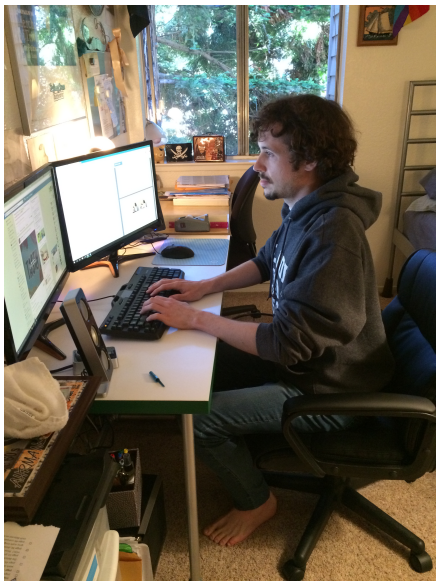
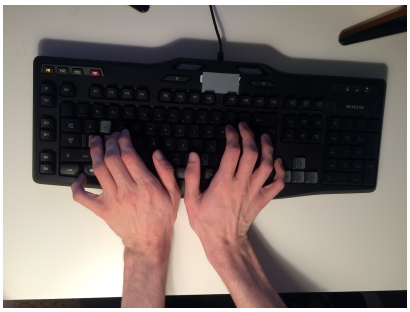


Figure 1.3: Good distancing from desk

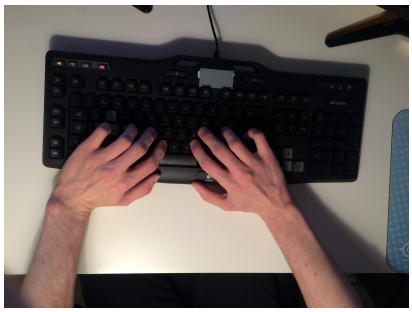
Figure 1.4 portrays four examples of bad wrist torsions. In 1.4a and 1.4b, we see an example of vertical wrist bending. These are called vertical wrist torsions. In 1.4c and 1.4d the bending are referred to as a horizontal wrist torsions. The distance from the desk affects both forms of torsions.

(a) Upward vertical torsion

(b) Downward vertical torsion



(c) Inward horizontal torsion



(d) Outward horizontal tortion

Figure 1.4: Improper wrist positions [28]

In Figure 1.5, we provide examples of neutral wrist positions from a horizontal and vertical reference. This position provides minimal torsion on wrists.

(a) Vertical

(b) Horizontal

Figure 1.5: Proper wrist positions

1.2 Motivation and Objectives

In the pre-computer age, jobs required a higher degree of physical activity: a builder placing supplies, a farmer ploughing a field. With the transition to white-collar jobs, workers must often adapt to a more sedentary work-style behind a desk. In fact, this lifestyle has become so commonplace that it is characterized in media. Office Space (1999) and The Office (2005-2013) are two works of comedy designed to highlight the workplace- both emphasizing the behavior of sitting behind a desk. Within the shows, each worker is confined to work at an assigned computer and an assigned desk for hours at a time. Given that a white collar worker spends approximately 82% of their time performing sedentary activities, this portrayal seems accurate [13].

In addition to this comically categorized sedentary work style, working behind a desk often requires continuous repetitive motions i.e. sitting at computer, moving a mouse, and typing on a keyboard. According to a study in the American Family Physician journal, repetitive motion is a contributing factor to the development of carpal tunnel [18].

Given that white collar workers are predisposed to MSDs, there needs to be options available to prevent it. If a worker is already affected by a MSD, there needs to be a way to prevent or reverse the damage. For carpal tunnel, preventative measures include lifestyle changes which may involve use of ergonomic keyboards, taking regular breaks from work, and avoiding repetitive motions [18]. Treatments post hoc include medication to relieve pain, use of wrist splints to bind the wrists neutrally, or injection therapy [18].

Now let us say a person developed carpal tunnel from work and had treatment. After the treatment finishes, what does this person do next? Likely they would

return to the same work. In order to prevent carpal tunnel from re-occurring, lifestyle changes would still be necessary. These lifestyle changes would need to be integrated into the workplace.

The goal of this work is to develop an automated posture assessment tool, allowing office workers to continue working worry-free. Such a tool would ideally satisfy the following criteria:

1. Easy to set up in a cubicle or work environment
2. Physically non-invasive so that the worker continue working free from distractions
3. Allow for some form of feedback on how to correct bad posture
4. Allow for incorporation of posture classification on multiple joints

Chapter 2

BACKGROUND

To provide context behind the work, this background section describes some requisites for understanding the work. First we define the software and hardware tools used in this research. Next, we cover some common image processing and machine learning techniques. Finally, we provide some details on how some of these techniques pertain to person detection.

2.1 Microsoft Kinect

The Microsoft Kinect v2 camera, seen in Figure 2.1, was released in 2014 offering several improvements over the v1 model. In this thesis we leverage the Kinect camera because it provides high quality captures, captures depth in addition to color, and the cost of 100$ is attractive for research.

We use the Kinect to collect a data-set of subjects, described later, based on color and depth information. Additionally, we leverage built-in joint detection where possible to bootstrap subject-joint information.

2.1.1 Hardware

In Table 2.1, we provide a list of Kinect v2 specifications.

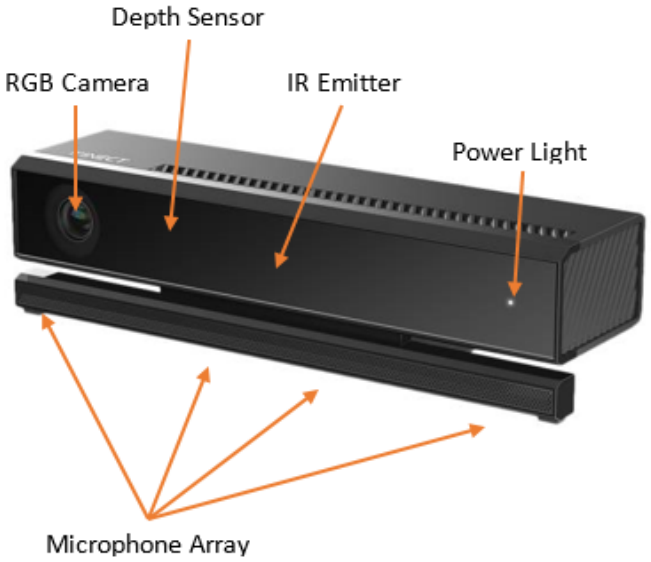


Figure 2.1: Microsoft Kinect v2 camera

Table 2.1: Kinect feature specifications [23]

Feature	Description
Body Tracking	Up to 6 persons
Joint Detection	Up to 25 Joints/Person
Depth Sensing	512 x 424 Resolution 30Hz
Active Infrared	512 x 424 Resolution 30Hz 3 IR Emitter
Color Camera	1920 x 1080 Resolution 30Hz
Depth Range	0.5 Meters to 4.5 Meters (Up to 8M on depth)
Field of View	70° Horizontal 60° Vertical
Microphone Array	Four microphone sensors linearly aligned

In this thesis, we leverage both the Color Camera and the Depth Sensing. The following subsections provide more detailed information on hardware features offered by the Kinect camera.

2.1.1.1 Color Camera

As mentioned in Table 2.1, the Kinect camera provides a color camera with a 1080p resolution at a frame rate of up to 30Hz. The 1080p resolution allows for high quality captures given the relatively inexpensive cost.

2.1.1.2 Depth Camera

To calculate depth information, the Kinect camera projects infrared (IR) light into a space through an array of three emitters located at the center of the Kinect. An IR receiving array, labeled as a depth sensor in Figure 2.1, then collects and examines the emitted data. Two criteria determine the result of the camera output:

1. The physical location of the IR reception in the array
2. The elapsed time since emitting an IR signal

It is possible to perform compute depth based on the two criteria noted because the speed in which light travels through air is relatively deterministic. The IR light, reflected off of objects within 0.5 meters and 8 meters of the camera, returns to the receiving array at different intervals in time. Depending on the physical location of the light response in the sensor array, and how long it takes to return, the Kinect calculates the distance that the reflected object must be. The Kinect provides simple means of interpreting the data in one of two ways: as a three dimensional coordinate map, or projected onto a two dimensional plane whose values are the from the camera. This projection can be calculated based on prior knowledge about the camera specifications. Figure 2.2 provides an example of the projected representation, where Figure 2.3 shows an example of the three dimensional coordinate map.

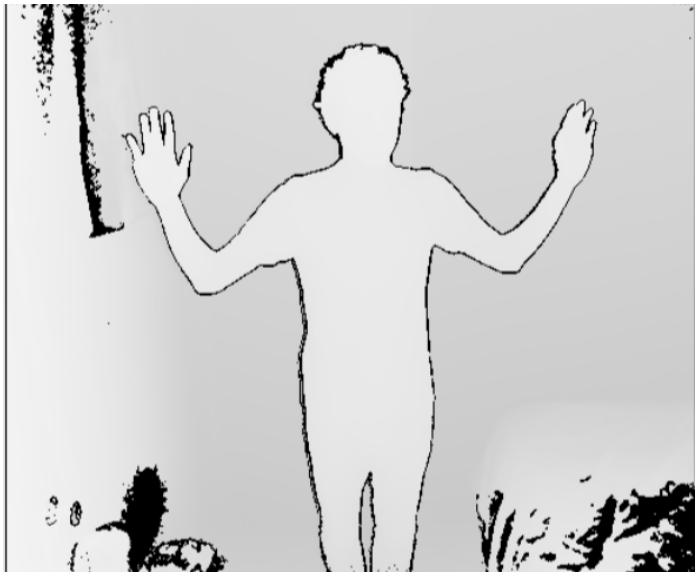


Figure 2.2: Kinect 2D depth image

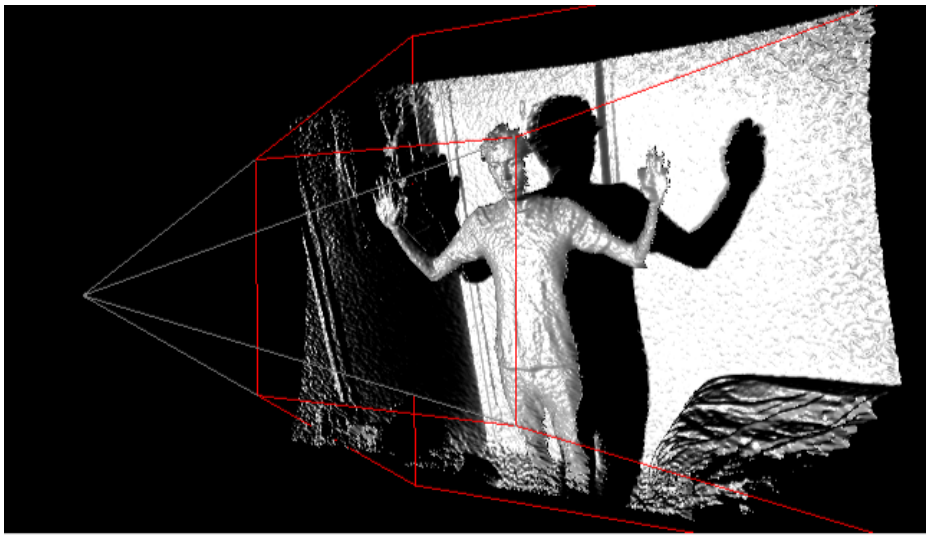


Figure 2.3: Kinect 3D depth image

2.1.2 Software

The Kinect camera toolkit provides several built-in software tools that perform pre-processing on raw input data from the IR, depth, and color information. The following section outlines software features pertaining to this project.

2.1.2.1 Coordinate Mapping

As seen in Figure 2.1, the color camera and IR sensor are not located in the same physical location on the Kinect- because of this, there are inherent discrepancies between the two images. In addition, each sensor contains a different quantity and configuration of sensors. These discrepancies prevent a one-to-one mapping of information between the two arrays.

Aware of this limitation, Microsoft released Kinect v2 SDK that includes a coordinate mapping tool. Based on a priori knowledge of the relative distance and camera properties, the sensor determines a mapping array from either color to depth, or depth to color. When using mapped information, the resolution is immediately degraded to the lesser resolution of the pair.

2.1.2.2 Joint Detection

Using Kinect joint detection, we can track a total of up to 25 joint positions [29]. The Kinect software derives this data specifically from static images, operating without using knowledge of previous frames. In Figure 2.4, we have a reference image of all joint locations.

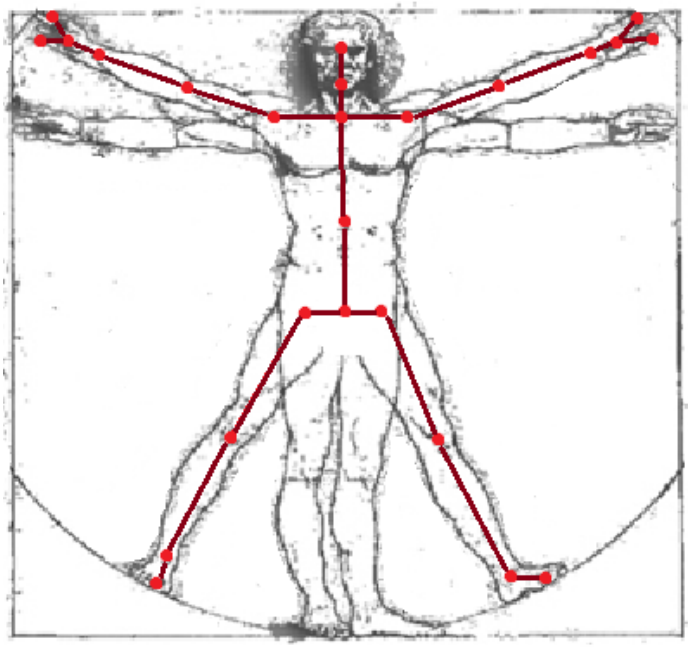


Figure 2.4: Kinect v2 joint locations

2.1.2.3 Body Extraction

At most, the Kinect camera can simultaneously detect six people in an image. Of the six people tracked, the camera collects joint data from at most two people based on proximity to the sensor and occlusions. To perform joint tracking, the internal software relies heavily on depth information. Ideally, a subject faces the camera directly and is separated from any other occlusions in the room. In Figure 2.5, we provide an example of body detection using the Kinect.

Figure 2.5: Example of body detection using the Kinect

2.1.2.4 Joint Estimation

Body detection acts as a preliminary step to computing joint locations. To train a classifier, Microsoft collected a database of over 500,000 frames in 100 sequences of different motion tasks [29]. They applied a random decision forest to determine 'regions' of likely joint locations, further refining these into single joint positions with additional processing. This detection is accurate, given that subjects are separated from occluding objects.

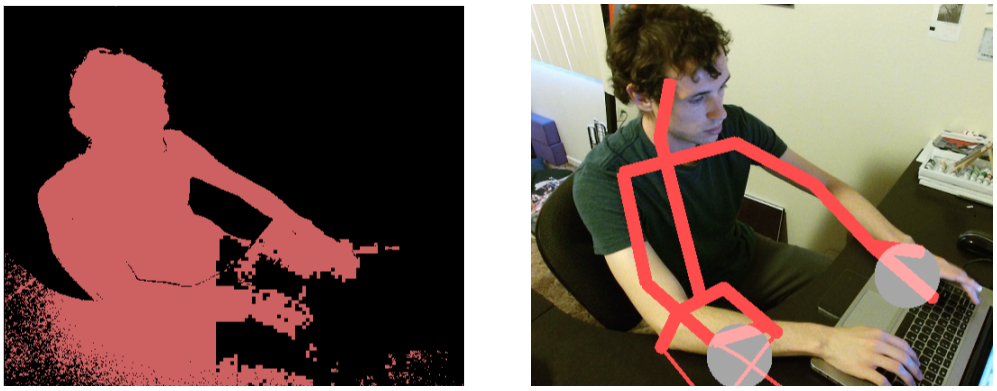
2.1.2.5 Occlusion Effects

The case of occlusion provides a shortcoming for the joint detection algorithm. In Figure 2.6, we capture a person sitting in front of a computer, partially occluded by a desk at an angle. In body extraction, the Kinect incorrectly interprets portions of the desk as being part of the subject. This causes an erroneous, probabilistic estimation

of arm and hand locations in the image, in that, the data is no longer useful.

2.1.2.6 Joint Jitter

In addition to occlusion effecting the proper extraction of joints, we must consider inherent 'jitter' in Kinect joint locations. According to a white paper [3] published on Microsoft's web-page, the Kinect is plagued by two forms of noise that alter its effectiveness in tracking. First, as with any system, we observe white noise. We perceive this noise as small fluctuations surrounding the actual location of the joint. Secondarily there are large variations caused by inferring positions of joints. These noises heavily alter the ability to statically compute accurate joint information while considering a single band of data.



(a) Body Detection (b) Joint Detection on Color Image

Figure 2.6: Unsuccessful body detection and resulting joint detection overlay

One method to remove this jitter presented by Microsoft [3] is by employing an Exponential Weighted Moving Average, EWMA filter. To describe this filter simply, the EWMA filter works by first taking a data point in time and performing an averaging behavior based on previous historical data. The algorithm gives this data less weight the further back it is from the current point in time. We describe this behavior in 2.1 [14]

$$EWMA_t = \lambda Y_t + (1 - \lambda) EWMA_{t-1} \text{ for t=1,2,...,n} \tag{2.1}$$

Where in this equation:

- $EWMA_0$ is the mean of historical data
- Y_t is the observation at time t
- n is the number of observations monitored
- $0 < \lambda < 1$ is a constant that determines the depth of memory of the $EWMA$

2.2 Image Processing

In order to perform computations on images in this work, we must define some terminology. This next section serves as a background in the field of image processing and explicitly defines some terminology used within the paper.

2.2.1 Color Spaces

A color image is represented by a two-dimensional array of picture elements (pixels). Each pixel represents detail on the sort of illumination we expect for that specific area that needs to be rendered. For example, if the color must be displayed as green, the computer needs some way to provide data to the monitor that can identified this pixel as green. The pixel in this case would store some numeric representation of this color data. Later, we will provide some detail on how to use this data within the realm of skin detection.

Figure 2.7: Image used for reference in color spaces

2.2.1.1 RGB Space

To interpret any kind of color, we can leverage several pre-defined color spaces. Typically we represent color within monitors by the Red-Green-Blue (RGB) color space. Each pixel on a monitor consists of a three filters for each band of RGB. By varying the intensity of these basic colors, the monitor produces a wide range of different observable colors. The Kinect v2 camera provides an RGB representation of the input- where each RGB image pane consists of an 8-bit value pertaining to the color intensity.

As an alternative to RGB, YCbCr and HSV spaces can be derived from RGB, which we define later. By using different color spaces, varying features can be highlighted within an image. In Figure 2.7, we have a reference image and in Figure 2.8, we have the reference image represented in an RGB color space.

(a) Red band (b) Green band (c) Blue band

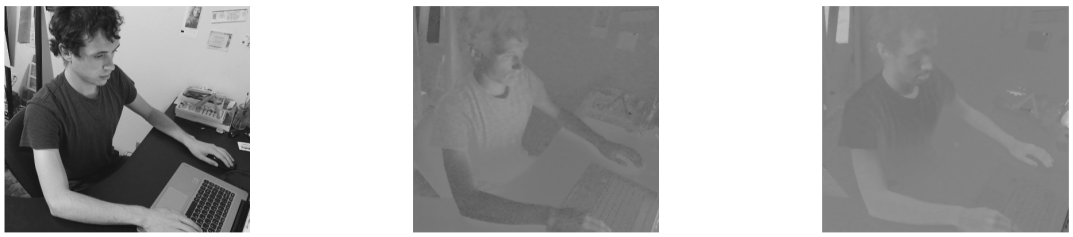
Figure 2.8: RGB bands separated into 8-bit components

2.2.1.2 YCbCr Space

Three bands define the YCbCr space:

1. Y, the luminence, or brightness in an image
2. Cb, blue difference, the difference between the blue component and a reference
3. Cr, red difference, the difference between the red component and a reference

In Figure 2.9, we split YCbCr into its three components. Luminance, represented in Figure 2.9a, is essentially the grayscale representation of the image, indicating how much light should be present in each pixel.



(a) Luminance band (b) Blue difference band (c) Red difference band

Figure 2.9: YCbCr bands separated and expanded into 8-bit components

To compute YCbCr we use the equation denoted in Equation 2.2. First we must scale the input RGB bands into a range of [0 1]. The resulting values for luminance range between [16 235], and blue difference, red difference range between [16 240].

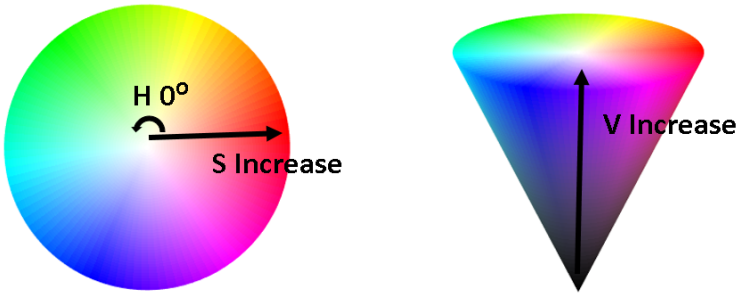
$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \tag{2.2}$$

2.2.1.3 HSV Space

The HSV or Hue-Saturation-Value is similar to YCbCr in that it divides the color into luminance and chrominance. We define the components of HSV as follows:

1. Hue, the type of color, (e.g red,green,blue)
2. Saturation, representing the intensity a color, or how much of the color can be seen in the pixel
3. Value, the luminance of a pixel

Figure 2.10 portrays a visualization of the HSV color space. In 2.10a we have a representation of the color wheel. At the zero degree point, hue is represented as red. This color changes as we increase the angle. The saturation, increasing to one, determines the intensity of a color- the highest intensity representing a pure color, such as a pure blue. With no saturation, the pixel can be represented as a shade of gray. Finally, in 2.10b we have the value, or gray scale intensity of the pixel. With zero value, we have a black color that is invariant to hue or saturation.



(a) Top view (b) Side view

Figure 2.10: HSV conical color space representation

We compute HSV from RGB using a set of equations outlined from 2.3 to 2.6. For this conversion, we again shift RGB into a range of [0 1]. The HSV conversion is nonlinear thus more complex than the conversion for YCbCr.

For hue:

$$H = \begin{cases} \theta, \text{if } B \leq G. \\ \\ 360 - \theta, \text{if } B > G. \end{cases} \tag{2.3}$$

and to compute θ:

$$\theta = \cos^{-1} \frac{\frac{1}{2}[\,(R-G) + (R-B)\,]}{[\,(R-G)^2 + (R-B)\,(G-B)\,]^{\frac{1}{2}}} \tag{2.4}$$

For saturation:

$$S = 1 - \frac{3}{(R+G+B)}[\,min(R,G,B)\,] \tag{2.5}$$

For value:

$$V = \frac{1}{3}(R+G+B) \tag{2.6}$$

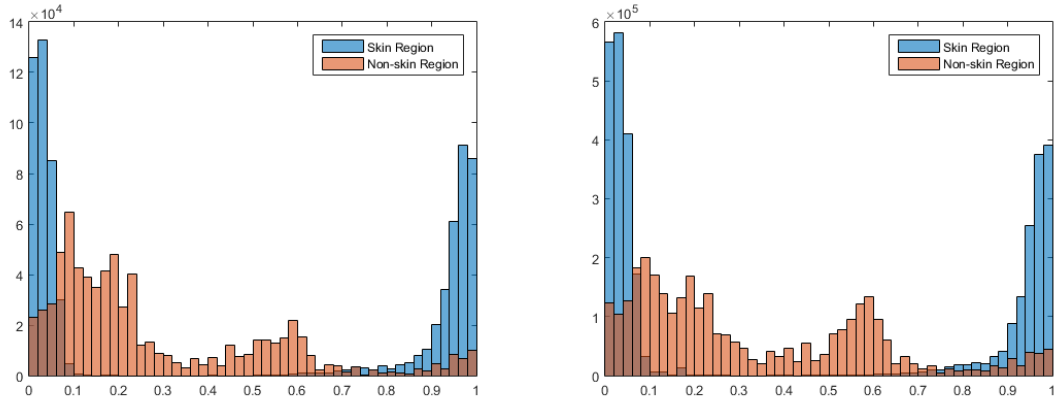
In Figure 2.11, we provide the resulting bands from performing this conversion.

(a) Hue Band (b) Saturation Band (c) Value Band

Figure 2.11: HSV bands separated and expanded into 8-bit components

2.2.2 Skin Detection

To improve effectiveness of the hue space in skin detection we right shift the hue space by approximately 72 degrees, or .2 when mapped into a range of [0 1]. To understand why this is useful, we collect images of faces in different environments from an online source. From these images, we generate masks by hand to highlight skin regions versus non-skin regions. We observe at 50 images of separate people in varying light conditions in Figure 2.12.



(a) Histogram of hue in varying light (b) Histogram of hue with different subjects

Figure 2.12: Histograms of skin versus non-skin color distribution in the Hue space with varying light conditions, and by different subjects

In Figure 2.12, there are two histograms of skin/non-skin pixels in the hue band.

To allow for an equal representation between skin and non-skin pixels, we normalize sampling based on the quantity of each classification. As seen with both images, there is a the skin hue lie in separate regions than the non-skin hue regions. By shifting the hue space by 72 degrees, approximately .2, we can join these two 'separated' regions. This allows for a distinct linear separation between skin and non-skin regions.

2.2.3 Image Gradients for Depth

Within the depth domain, we represent points as millimeter distances from the camera. From this representation, we can derive a two-dimensional depth map of the sensor space. An effect of this mapping is that the same object provides different depth values depending on its distance from the sensor. Having raw depth information allows us to assign both a distance and a scale to objects within view. To draw additional information from this depth mapping, we can observe the rate in which depth information changes with respect to other pixels within the image. With this information, known as an image gradient, we can derive basic attributes about objects in the image. Large changes in depth can indicate corners of objects, where parabolic changes may imply round objects, and near-zero changes imply flat objects. We use Figure 2.13 as a reference to provide an example of the gradient response. This image contains 2D depth information backing the location of each pixel.

Figure 2.13: Kinect-generated reference image of a flat table

To solve for an image gradient, we first determine directional components g_x and g_y. These values represent a one-directional gradient. To compute the two gradient components, we convolve image with two gradient masks, where the second matrix rotates the first by 90 degrees. In Figure 2.14, we provide some examples of pairs of gradient matrices.

-1	0
0	1

0	-1
1	0

(a) Roberts gradient pair

-1	2	-1
0	0	0
1	2	1

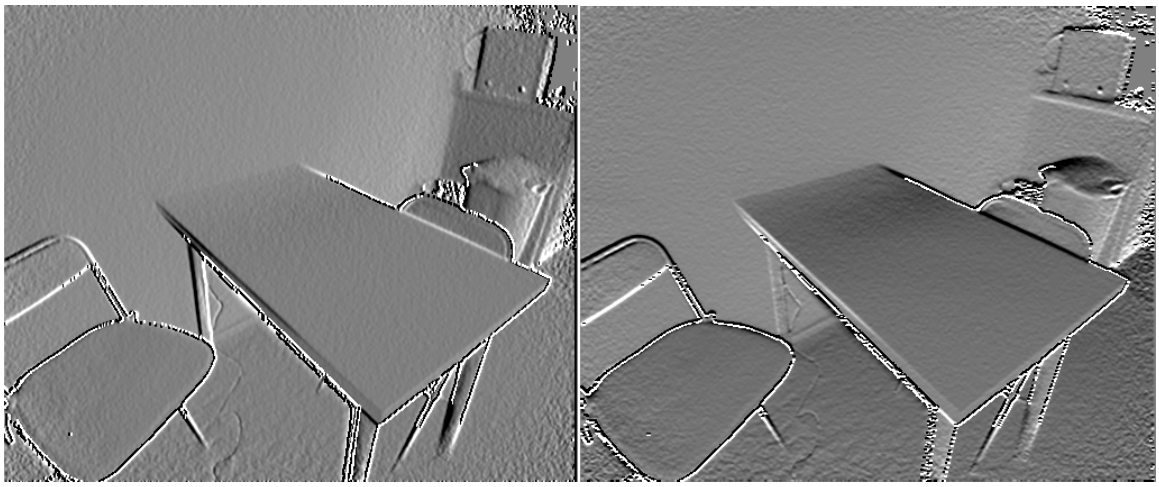
-1	0	1
-2	0	2
-1	0	1

(b) Sobel gradient pair

Figure 2.14: Two pairs of masks used in computing gradient components[12]

We apply Sobel masks to the image in Figure 2.13 and produce directional com-

ponents, in Figure 2.15. White and black pixels represent the highest depth variation in both the positive and negative directions, where gray represents almost no change.



(a) g_x (b) g_y

Figure 2.15: Directional components of reference image with applied Sobel filter

2.7 contains a representation of the two-dimensional gradient ∇f of an image.

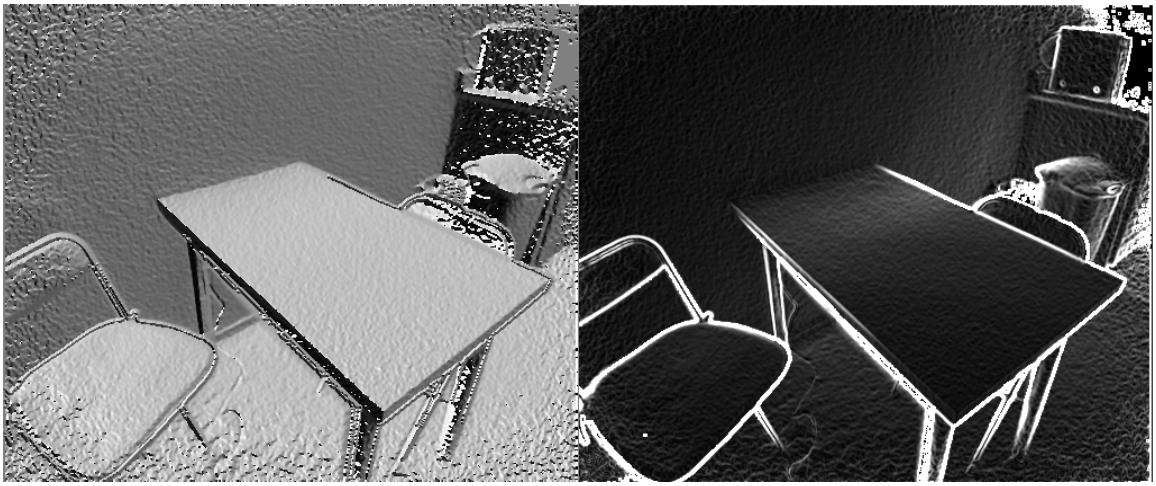
$$\nabla f = \begin{pmatrix} g_x \\ g_y \end{pmatrix} \tag{2.7}$$

One method of viewing these gradients would be to represent each point as a matrix consisting of complex number, representing the magnitude and directional change of combined g_x, g_y. Another method is to extract the magnitudes and angles separately by performing common conversions.

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \tag{2.8}$$

$$\angle \nabla f = tan^{-1}(\frac{g_y}{g_x}) \tag{2.9}$$

We can apply this formula to our example, using the gradient components mentioned in Figure 2.16. These images are generated using Matlab's imshow, where values are normalized into a pixel range of 0 to 255, where 0 represents the lowest value in the spectrum, and 255 represents the highest. With respect to direction, the lowest value approaching -180 degrees would be of value 0 and the highest approaching 180 degrees would be represented as 255. In Figure 2.16a, the raw magnitude value is unfiltered- due to noise, the magnitudes calculated are changing pixel-to-pixel. Given a filter on the original image, this value could appear much smoother. In Figure 2.16b, we observe a similar effect. Both the magnitude and direction appear to provide useful edge detection on flat surfaces, such as the table and chairs represented.



(a) Normalized $M(x, y)$ (b) Normalized $\angle \nabla f$

Figure 2.16: Gradient components of reference image with applied Sobel filter

2.2.4 Morphological Operations

As mentioned in 2.2.1, images on a computer are represented via 3 bands of numeric data arrays. In the case of standard RGB we observe these values ranging between 0-255 for an 8-bit pixel. Though traditional monitors are only capable of

displaying pixels with a resolution of 8-bit bands, it is possible for them to be represented by any quantity of bits. If we generate image representations using a single bit of data per pixel, this is known as a binary image. Many morphological operations, as described in this area, apply to any range of data information. For the sake of simplicity, we will observe these operations for a binary image. In addition we will reduce the complexity to two dimensional operations.

2.2.4.1 Structuring Elements

A structural elements, SE, consists of a set of binary values that identify characteristics of an image. SEs are typically odd, and we consider the middle element as the center of the structure. We represent an SE as a two dimensional image of [0 1].

Using an SE, we define regions within an image to be either a member, or not a member of the SE set. To determine membership, we verify a pixel region contains all values which are defined in the SE set. To perform this comparison, we place the center of the structural element at the center of each point in the image. In the case of membership in a binary image per pixel, membership is determined a region containing a value at all indices specified by the structural element.

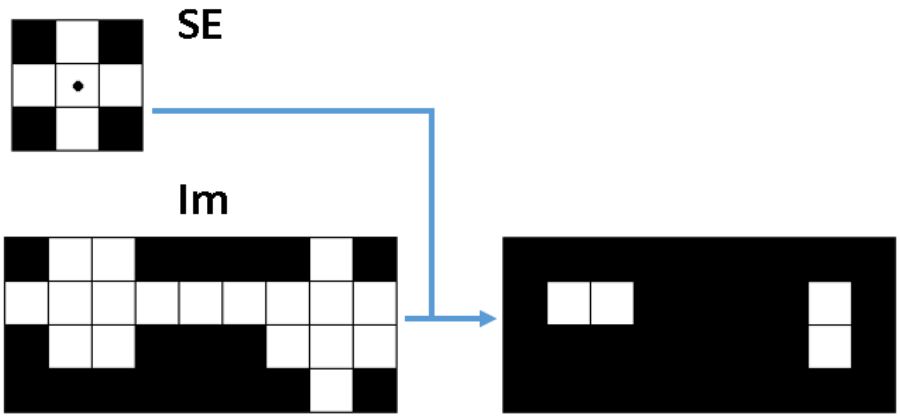


Figure 2.17: Morphological erosion operation applied using a structural element on an image

In Figure 2.17, we show an applied 3x3 SE to an image, Im. For example, if we would like to determine which pixels within the image are a member of the structural element SE. As noted before, we must move the center of the SE, denoted by a black dot- to each element within Im. Members are those which contain the SE. Our resulting image consists of 4 pixel regions.

2.2.4.2 Erosion

To begin with a simple operation, we examine erosion. Figure 2.17 is the affect of an SE applied to an image using an erosion operation. Using set notation, we compute $Im \ominus SE$, the erosion of Im with SE within a two dimensional Z^2 domain of z points using 2.10 [12].

$$Im \ominus SE = \{z|(\hat{SE})_z \subseteq Im\} \tag{2.10}$$

In 2.10 we see the result of the erosion of image Im by structural element SE is a set of indices- where by translating the structural element to each pixel, the occurrence of all elements of the SE are contained within Im. Performing erosion can effectively separate two regions that are connected by a relatively thinner line. In 2.17 we originally have a single cluster of points. After erosion is performed, we now have two clusters.

2.2.4.3 Dilation

As erosion is capable of separating clusters of points, dilation can join them. We describe dilation, denoted by $\oplus$, in equation 2.11 [12].

$$Im \oplus SE = \{z|(\hat{SE})_z \cap Im \neq \emptyset\} \tag{2.11}$$

Once again we translate the SE to every point in the image. In this case, if there is any intersection between the translated structural element and the image, then we determine point z is a member of the dilation set of the image. Figure 2.18 provides an example of this operation.

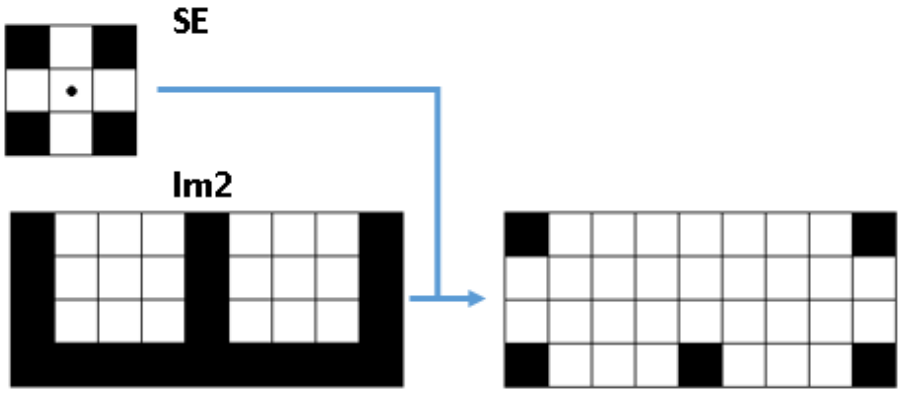


Figure 2.18: Morphological dilation operation applied using a structural element on an image.

2.2.4.4 Opening and Closing

By combining dilation and erosion in different orders, we retain certain properties of the original clusters of a binary image. For both 2.11 and 2.10, the resulting positive regions are either much smaller, or much larger, respectively, than the original image. To retain sizes, but still achieve combined clusters, as in Figure 2.18, we apply the alternating dilation, erosion operation, denoted in 2.12.

$$Im \circ SE = (\, Im \ominus SE) \oplus SE \tag{2.12}$$

$$Im \bullet SE = (\, Im \oplus SE) \ominus SE \tag{2.13}$$

The opening of an image can be defined in 2.12 and its analogue, closing in 2.13 [12]. The effect of performing opening/closing operations can be seen in 2.19

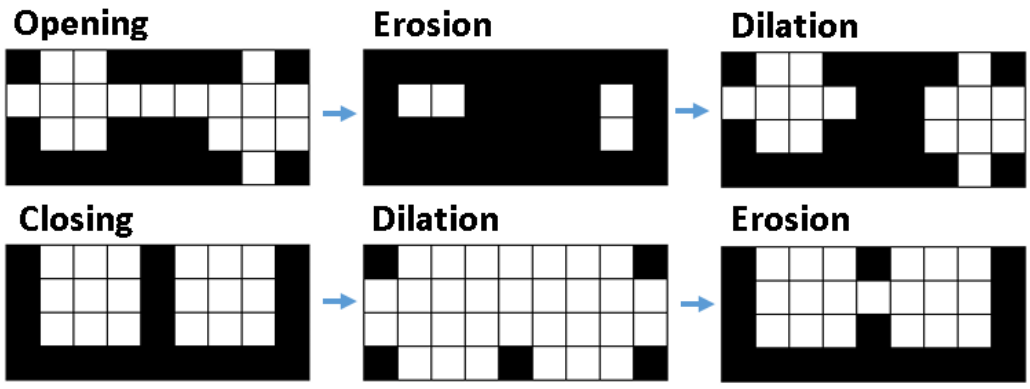


Figure 2.19: Effects of performing opening and closing operation on SE used in Figure 2.17

2.2.4.5 Connected Component Extraction

Connected component analysis provides another method of viewing clusters within an image region. Though connected components can be viewed in various degrees, we will focus on 4-direction connectedness. Any point z within a set is a connected component with another point, if they are located within at most 1 pixel in either horizontal or vertical direction. The same SE in 2.17 can be used in checking for connectivity. The process for extracting connected components is iterative and is outlined via Equation 2.14.

$$X_k = (\, X_{k-1} \oplus SE) \cap Im \quad k = 1, 2, 3... \tag{2.14}$$

Where X_k are k connected components within an image [12].

Once all points are contained within k connected components, the iterative operation ends. With an image separated into components, next shape features can be extracted to classify the nature of the components. Such meta data includes area, perimeter, major axis length, minor axis length, and orientation. For example, areas of different regions could be compared for determining the importance of components.

2.3 Machine Learning

One important factor for us to determine classifications through machine learning is by reducing dimensionalities of problems. To determine good back posture, for example, we may want to examine how straight the back is by calculating the angle between back joints. We call this angle a feature of the Kinect joint space. This reduction serves two benefits- first now a classifier has a singular data point instead of three, second we can highlight specific criteria on the collected data in order to prevent bias. Given our back posture example, for our classifier we are less concerned about the specific depth space our joints are in, and more concerned on their locations with respect to each-other.

The typical process for developing a classifier involves an initial training, or learning phase. In this phase, we provide a mapping of features which have an applied label describing their result, e.g. good posture or bad posture. During this learning phase, the classifier attempts to discern a division between labelled feature points. Training data refers to data we input into a learning phase to develop a classifier, where the testing data is labeled data reserved for observing the effectiveness of a classifier. We can use one of several criteria in evaluating how well the classifier performs. We will focus on three such criteria:

1. Sensitivity
2. Specificity
3. Cohen's Kappa Score

Given we consider identified good posture as a true positive result, and identified bad posture as a true negative result, we can define these metrics.

2.3.1 Sensitivity

This score is the ability for the classifier to correctly identify good posture. To compute this, we apply Equation 2.15, where a perfect score would be 1, when all good posture data is correctly identified.

$$Sensitivity = \frac{TP}{TP + FN} = \frac{\text{True good posture}}{\text{Total number of good posture results}} \tag{2.15}$$

2.3.2 Specificity

Specificity is the ability for the classifier to correctly identify bad posture, the opposite of Sensitivity. To compute this, we apply a similar equation:

$$Specificity = \frac{TN}{TN + FP} = \frac{\text{True bad posture}}{\text{Total number of bad posture results}} \tag{2.16}$$

2.3.3 Cohen's Kappa Score

This score provides a metric for overall reliability of results. This reliability "... represents the extent to which the data collected in a study are correct representations of the variables measured". This score, ranges between -1 and 1, where a score of about 0.41 can be considered acceptable Kappa scores according to Cohen [21]. The Kappa score additionally compensates for chances of fluctuations due to randomness. It is computed as:

$$\kappa = \frac{Pr(a) + Pr(e)}{1 - Pr(e)} \tag{2.17}$$

Where Pr(a) is the actual observed agreement, computed as:

$$Pr(a) = \frac{TP + TN}{Total} = \frac{\text{Correctly identified posture count}}{\text{Total posture data count}} \tag{2.18}$$

And where Pr(e) is the probability of random agreement, computed as:

$$Pr(e) = \frac{m_a + m_b}{Total} \tag{2.19}$$

$$m_a = \frac{(TP + FN) * (TP + FP)}{Total} \tag{2.20}$$

$$m_b = \frac{(TN + FN) * (TN + FP)}{Total} \tag{2.21}$$

2.4 Cascade Object Detection

A cascade detector is a trained detector that identifies complex patterns within an image. Cascade detectors are commonly used successfully in face detection [34, 33, 7, 19].

The first step in cascade detection is to build a trained classifier. To do this, we feed positive and negative samples into an iterative algorithm. For Matlab, we define positive samples as hand-selected sub-images of an image. Matlab derives negative samples directly from other regions of the image.

The cascade detector is designed as, a cascade of stages that become increasingly discrete in scrutinizing sections of an image. The detector scrutinizes sections of an image based on a feature type. To train the classifier, iteratively we go through a process of deriving a classifier that reaches some user-defined threshold of acceptance and then we discard the samples misclassified as negatives.

Once trained, the cascade detector is re-used across different images. Based on the quantity of input data and cascade structuring, the success rate of detectors varies. Bias based on common qualities from the training data provides another limitation on cascade detection. For example, we have two built-in face detection cascade classifiers provided by Matlab:

1. Front face

2. Profile

These detectors were originally trained based on completely different input data. Why did the team not simply stick these two training image collections together? To quote the idiom "Jack of all trades, master of none" having trained data on faces of specific positions increases the ability for a detector to successfully identify that position. If we want to use both detectors, it is likely more effective to simply train two cascade detectors and run an image through both.

2.4.1 Cascade Detector Feature Types

In this work we consider three feature types:

1. Haar-like

2. Local Binary Patterns(LBP)

3. Histogram of Oriented Gradients (HOG)

Each feature type breaks down the sub-images into different component forms.

2.4.1.1 Haar-like Features

The name for Haar-like features was derived from Haar wavelets- that are essentially square function that oscillate once around an axis at a particular time interval. To derive a Haar-like feature from, for example, a black and white image, we observe and record the change in intensity moving across the image in a given direction. Moving from a black region to a white region outlines a separate change in intensity from observing a purely white or purely black region. In the regions where we have

intensity changes, we can interpret these features as a square function, similarly to Haar wavelets. This behavior is where the name for haar-like features are derived.

2.4.1.2 Local Binary Patterns Features

Local binary features can be very effective at pulling out patterns, or textures within an image. To generate LBP features, a sub-image is created where we determine some form of intensity scale, luminosity, for example. For each pixel in the image, we must check each direct neighbor in the same order per pixel. Next, we assign a binary system to the difference between the two luminosity values. If the neighbor is a higher number, then assign a value 0 or 1. For the opposite, assign 1 or 0. Next, we combine this into a N-digit binary number. we Finally, create a histogram of the entire sub-image based on the N-digit binary number created for each pixel. This is a very simple LBP feature.

2.4.1.3 Histogram of Oriented Gradient Features

The first step in creating a HOG feature space, is to compute the gradient. This can be done by applying a gradient filter kernel such as the Roberts or Sobel gradient seen earlier, to an image. This is performed across each pixel, where pixels are grouped into subimages. Based on the angle computed by the filters, these gradients are weighted and combined into histograms. A form of the magnitude of the gradient effects the 'weight' of a pixel vote. These histograms would next be normalized.

Chapter 3

RELATED WORKS

The Microsoft Kinect camera has been a popular tool for many posture recognition studies since its release in 2010. In its intended environment, the Kinect allows users to incorporate body movements as an additional input into the virtual world. As such, the Kinect comes with standard tools that allow for rough approximations of body locations and hand orientations. Firstly, a draw for research into use of Kinect cameras comes from its ability to identify and separate persons from a background environment. In a gaming environment, developers use this functionality to track players and discern from normal household objects, such as a couch or tables. In upcoming research we provide some examples of the Kinect's functionality in research. A secondary draw to using the Kinect is the price point; With a price of the Kinect v2 camera at $149.99, researchers are able to afford the feature-rich device at a fraction of equivalent earlier devices. The following chapter outlines some posture-related projects that leverage the Kinect.

3.1 Standing Workplace Assessment using OWAS

In a paper published in 2013, "Using Kinect^{TM}sensor in observational methods for assessing posture at work" [9], the team develops an assessment system which categorizes posture risks levels. Jose Antonio Diego-Mas and Jorge Alcaide-Marzel of Universitat Politécnica de Valéncia, Camino de Vera, Spain, develop this system to follow ergonomic rules specified by the OWAS method. Using this method, they classify body posture under distinct four-digit codes based on back, shoulders, legs, and load (See Figure 3.1 for reference codes). Next, these code were categorized into a series of four classes, 1-4, a higher class signifying a higher risk. A Microsoft Kinect v1 sensor captures a subject position over even intervals. The frequency of occurrences of risk levels over time intervals factor into posture assessment.

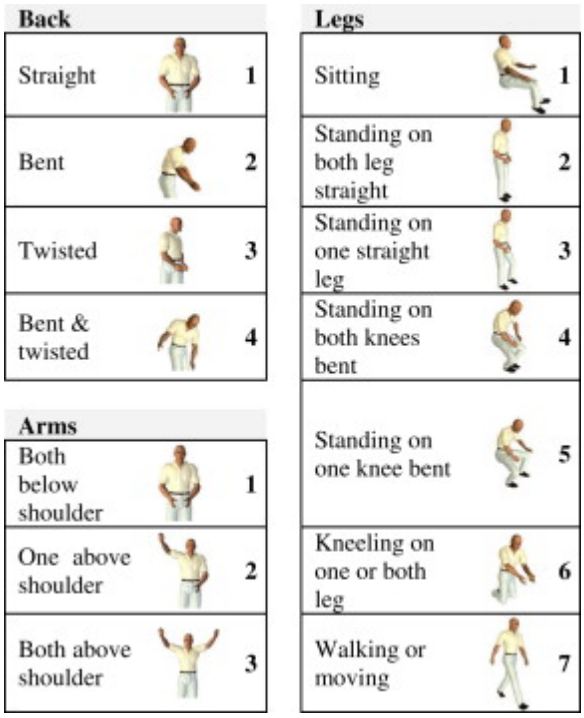


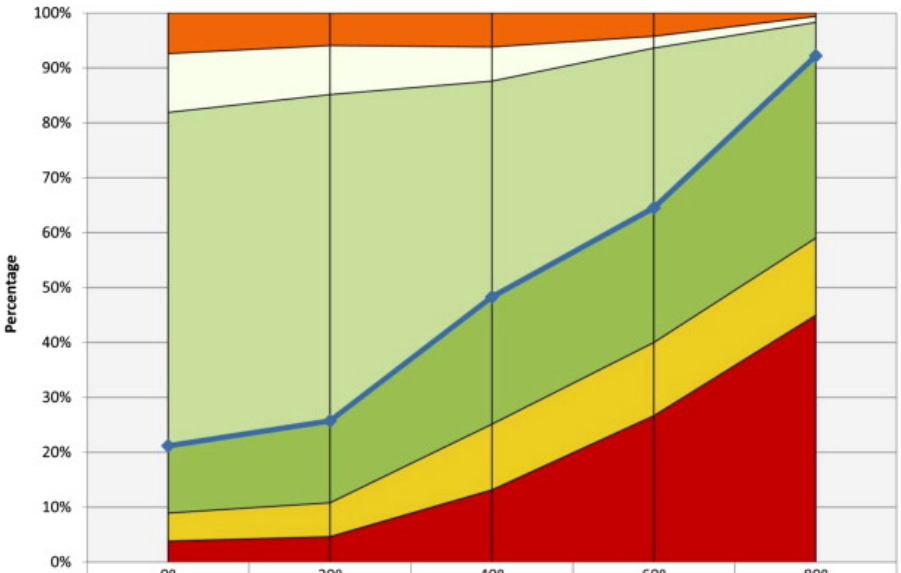
Figure 3.1: OWAS risk codes for back, shoulders, and legs [9]

They capture both both RGB and Depth data in order to locate subjects in an

image. They incorporate standard Kinect joint tracking techniques to determine the subject body locations. Subjects are separated from nearby objects to allow an explicit threshold of the subject from background and foreground. From extracted three dimensional joint information, the team draws three joint location planes:

1. The Trunk Plane
2. The Sagittarial Plane
3. The Frontal Plane

They determine joint angles by comparing planes and projecting joint pairs against each other. The paper additionally examines the effectiveness of the Kinect camera in different positions- from distances of zero degrees (facing the front of the subject) to 80 degrees. They compare Kinect classifications against the consensus classification of two OWAS experts. Figure 3.2 shows these compiled results.



	0º	20º	40º	60º	80º
%No Inferred/Wrong Coded/Wrong Categ.	7.40%	5.94%	6.20%	4.26%	0.62%
%No Inferred/Wrong Coded/Right Categ.	10.72%	8.91%	6.20%	2.13%	1.09%
%No inferred/Right Coded/Right Categ.	60.71%	59.42%	39.30%	29.10%	6.09%
% Inferred/Right Coded/Right Categ.	12.24%	14.92%	23.18%	24.52%	33.19%
% Inferred/Wrong Coded/Right Categ.	5.10%	6.17%	11.96%	13.33%	14.05%
%Inferred/Wrong Coded/Wrong Categ.	3.83%	4.63%	13.15%	26.67%	44.96%
% Total Inferred Postures	21.17%	25.72%	48.29%	64.52%	92.20%

Figure 3.2: Estimated success table of observations over different angles

[9]

From the estimated success table we observe two effects of changing the angle. The green labels represent instances where the Kinect classification results align with the expert opinion. The red show areas where the results disagree. Also to be noted are the two different shades of green. The lighter green label represents data points where the Kinect could not definitively locate a joint; the joint location is inferred based on other joint locations. This occurs at higher rates when aligned at 0 degrees, ie directly in front of the subject. As the angle increases, the likelihood of an inferred image increases from 21.17% to 92.20%. Additionally, the quantity of incorrectly classifications increases from 27.05% to 60.72% at 80 degrees.

From the data presented in this paper we conclude a large decrease in ability to classify postures based on larger deviations from a forward viewing angle. We must

consider viewing angle in a system designed for assessing posture. This paper derives data from a subject free from environmental occlusions. We must also consider some leniency to viewing angle in this paper to allow a higher quantity of joint information to be collected.

3.2 Back Posture Biofeedback

In a 2013 research project, Students within the Department of Computer Science in Saskatoon, Saskatchewan Canada develop a posture biofeedback tool using a Microsoft Kinect camera. With a focus on a non-invasive system, the students conclude that using a camera would provide a suitable means of classifying posture. The team incorporates a third party API, PrimeSense OpenNI, to collect raw data from the camera.

To operate this proposed tool, the team calibrates the proposed system against the subject. To limit the degree of data, they track only five upper body joints. From these joints, they compute the following angles:

1. Neck to Head
2. Torso to Neck
3. Left to Right Shoulder
4. Lateral tilt of the spine

The system provides biofeedback via a system rigged on a cart, containing a 24" monitor with a mirrored view of the user orientation. They overlay color on the mirrored view specifying classification results. A green overlay represents classified good posture, where orange/red overlay represents that posture corrections are required.

The requirement for calibration limits in ease of use of this device. To operate, either a trained operator supervises, or a the team instructs the user how to calibrate it. Both techniques introduce user error. In addition, this solution also requires an occlusion free zone. For workplace posture requires the user be sitting a distance away from the desk- a self defeating statement for classifying workplace posture.

For the purpose of this work, this research concludes that it is possible to design a system capable of interpreting posture using a Kinect camera and joint information. We cannot draw much qualitative information directly from the paper- it merely shows some precedence to the research outlined in this thesis.

3.3 Driving Posture Recognition

In a paper [35], Yan et al. of University of Liverpool provide investigative results for the use of neural networking in learning and predicting driving postures.

They develop and present a method to reduce traffic accidents caused by driver fatigue and inattention. The method intends to accurately detect when a driver is distracted, based on learned driving postures. The team develop a proposed scheme detecting 'bad' driving postures, presenting a system boasting a best-case accuracy of 99.78%

To verify their approach, Yan et al. tested against a Southeast University video clip driving posture set. This dataset includes clips of normal driving, as well as eating, smoking, and cellular phone use.

They select a convolutional neural network(CNN) to pull features directly from the training set, focusing on characterizing hand positions- where locations are made difficult through varying lighting. They originally considered skin segmentation for their system, but due to the difficulty of lighting, CNN was chosen. To train this

CNN they generate data set using the same car in each image, with the camera in the same position- mounted on the passenger side door. They define four distinct positions.

1. Normal
2. Using gear shift
3. Eating
4. Taking a phone call

The architecture for the derived unsupervised CNN consisted of 9 data size transformative stages, including inter-weaved convolution and maxpooling stages, followed by successive full connection stages. Each convolution stage, using a kernel function, formed feature vectors, which would be subsequently used in the following pooling stage. The pooling stages were used to soften the data, eliminating some sensitivity of the feature detection. Max pooling in the article is simply non-linear max filtering on the specified data region.

In addition to the transformative stages, for each inter-weaved convolution stage there was also a local normalisation and non-linear activation stage. The non-linear activation stage occurs after the convolution stage. Instead of a standard sigmoidal activation, an alternative function, ReLU was performed. ReLU activation replaces any negative convolution value with 0. A local response normalisation was used for the normalisation stage.

The outline proposed on this paper presented a 99.78% accuracy on average in determining which posture a driver was in. This compares to the baseline percentage of 90.63%. In further research of this project, a convolutional neural network for wrist posture identification may be useful. In conjuction with a reduction on the data space for wrist locations, a CNN design may be able to perform sufficient classification.

Chapter 4

RATED POSTURE DATASET

To fulfill objectives specified in Section 1.2, we generated a custom data set by enlisting participants in a human study. After building a set of data, we employed crowd-sourcing to generate labelled images. This provides us with both:

1. Quantitative posture data
2. Qualitative reference images for 'good' and 'bad' posture

In this section we discuss the process of generating this data, and present some limitations.

4.1 Generating Data

First we generate data by enlisting participants in a human study. To gather information on human subjects, we adhere to the human subjects protocol for Cal Poly [6]. We created a protocol for evaluating subjects and provided a consent form to participate. These forms are included in appendices for reference.

4.1.1 Overview

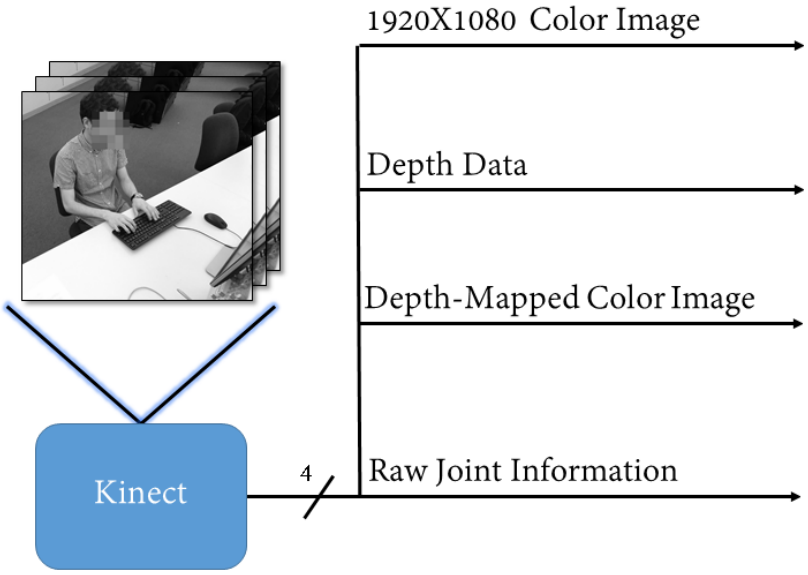


Figure 4.1: Initial workflow from Microsoft Kinect capture

In Figure 4.1, we describe the details captured from the Kinect captures on subjects. For each recorded subject, we capture static captures once every 5-10 seconds containing depth, mapped, and color images. We collect joint information for all joints as a continuous stream of data, with a hardware-limted resolution of up to thirty captures/second.

We place the camera between three and four feet distance from the subject, at approximately at a sixty degree angle for the subject, seen in Figure 4.1. The Kinect captures subjects for up to 5 minutes performing tasks on a computer.

4.1.2 Subjects

We select subjects aged between eighteen and fifty to participate- each aware of the purpose of the study. Any subjects with a pre-existing MSD cannot participate

as per protocol. To examine these subjects, we prompt them to perform tasks on a computer under observation of a Kinect Camera: first perform a typing test, then browse the Internet. During the tests, we request the subjects to perform adjustments to their position. By requesting these adjustments, we can generate additional data points that are objectively good or bad posture.

4.1.3 Workstation

We use a Lenovo IdeaPad U410 Laptop in processing data gathered by the Kinect. Specifications of the laptop are provided in Table 4.1.

Table 4.1: Computer hardware specifications

Item	Specifications
Processor	Intel Core i5-3337U CPU
Memory	6144MB
GPU	NVIDIA GeForce 610M
Operating System	Windows 8.1 and Ubuntu 14.04 Virtual OS

4.1.4 Candidate Reduction

In order to reduce large variations due to inferred joints, for our labelled images we perform a reduction based on quantity of inferred joints. Ignoring the 9 lower body joints, we require that at least 10 upper body joints be present. This incorporates lenience into the 6 joints tracking both the left and right arm.

4.2 Rating Positions

In order to determine posture classifications we employ crowd-sourcing. We build a system that uses a webserver, queries users to rate postures on a scale, and records the result of a set of classified training data. Figure 4.2 shows this process. In the following sections, we describe each steps in detail.

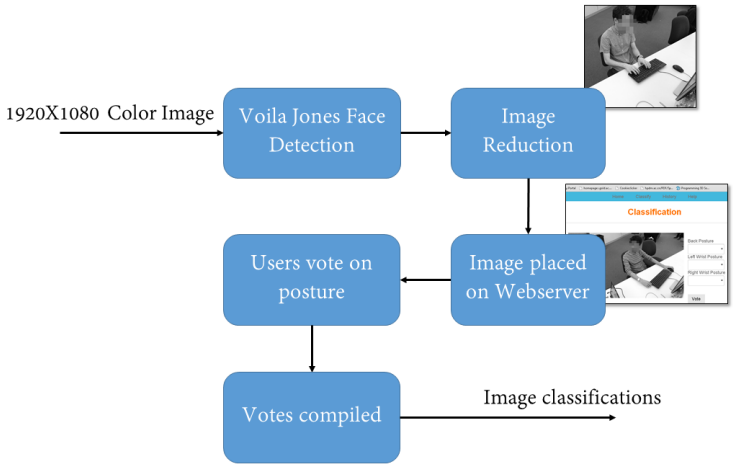


Figure 4.2: Steps to generate classifications for image frames

4.2.1 Face Detection and Blurring

As per the human subject consent form, we must retain anonymity of the research subjects. To allow both anonymity and a means of classification, we leverage Matlab to detect faces and blur these faces. Using this detection scheme we capture approximately 90% of faces in images. Note that this is not 100% successful because Matlab does not configure a face detection trained at the angle that we film subject. We blur any face missed by the built-in detection manually.

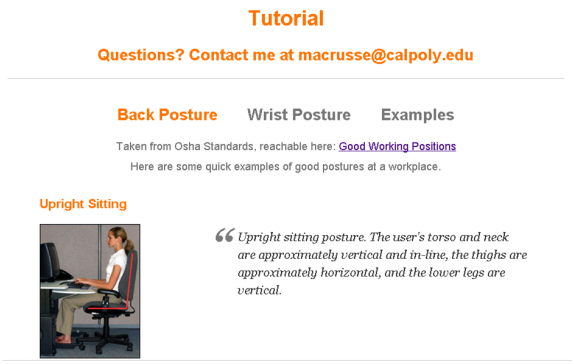
4.2.2 Reduction

As we are placing these images on the web, we must compensate for loading time of the images. We reduce the 1920x1080 color images to one-half scale. Additionally, we convert images into gray-scale. This reduces the data footprint per image by a factor of 12.

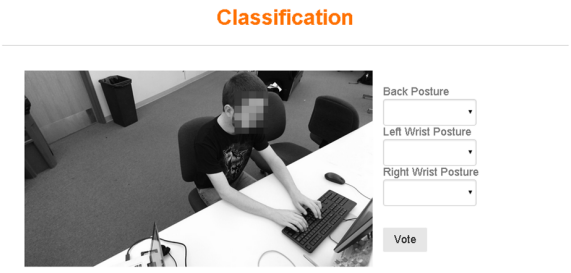
4.2.3 User Voting

We created a website for posture classification that facilitates crowd-sourced user voting. To store votes, people, and images, we use a SQLite database. We employ Django, a Python web framework, and host the web server on Pythonanywhere.com. In Figure 4.3, we outline the three pages presented to the user on navigating to this website. To allow users to become more skilled in the technique of posture analysis, we provide users with a tutorial page as seen in Figure 4.3a. We reproduce the OSHA specifications for good posture in a new configuration so that we can establish a baseline knowledge for evaluating posture.

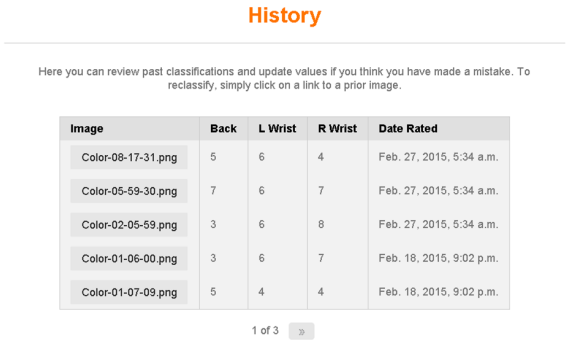
The voting stage, shown in Figure 4.3b, allows users to assign one of 11 ratings to the left wrist, right wrist, and back. We instruct users to vote on a scale between terrible posture, 1, and perfect posture, 10. For cases in which a user is unable to determine a rating, they can select 'unknown'. Users can review and edit prior votes using a History Page as seen in Figure 4.3c. To allow users to resume a prior session on the same computer or another, we provide login functionality and locally cache credentials. If a voter prefers anonymity they can select an option to not store any credentials. Some voters were asked to simply determine back ratings based on not having enough time to perform a full classification of all three postures.



(a) Tutorial Page



(b) Classification Page



(c) History Page

Figure 4.3: Pages within a web page generated to build classifications

In order to provide an even spread of ratings, we randomize the order of images that we present. Once user rating is completed, we collect the results.

4.3 Results

For our study, we collected 1,186 depth images, 1,136 color images, and 1,122 depth-mapped color images from 17 subjects. From these images, we extracted 600 candidate images to be used in training, approximately 35 images per subject.

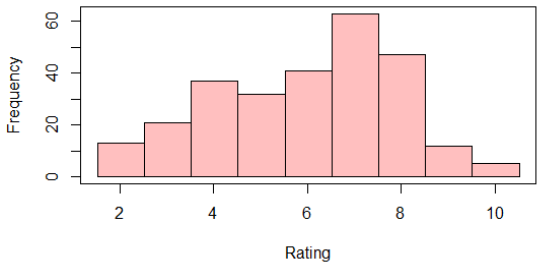
In joint captures, we collected 48,569 frames of seated posture data across 3,349 unique hh-mm-ss timestamps. This equates to approximately 14.5 frames/second of captured data.

Of these 600 images we ultimately classify 271 unique images. Between left wrist, right wrist, and back we collected 1,158 total classifications. Table 4.2 provides a breakdown of these results.

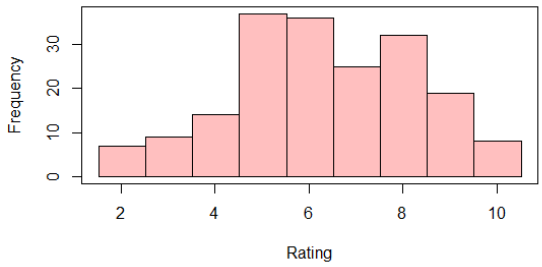
Table 4.2: Summary on vote collection for ratings

	Images	Ratings	Med. Score	Mean Score
Back	271	545	6	5.813
Left Wrist	187	300	6	6.197
Right Wrist	195	313	6	5.992

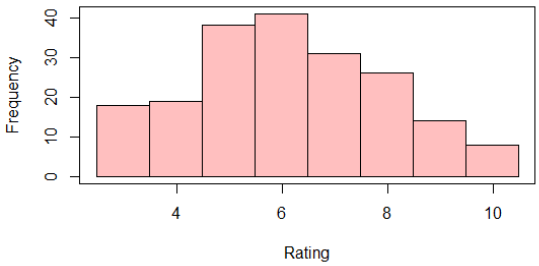
For all rating types, we observe the same median score of 6 and a mean score of roughly the same value. To provide another dimension to these results, we generated histograms of each consolidated rating. Before consolidation, each image contains a variable-length list of user ratings. We perform a mean against this list, with the results displayed in Figure 4.4.



(a) Back



(b) Left wrist



(c) Right wrist

Figure 4.4: Distribution of ratings per joint

In Figure 4.4, both back and left wrist each have two explicit local maximum separated with a local minimum. For back, the center drops at local minimum around a rating of 5. For left wrist, we see a drop at around 7. With this distribution, we can conceivably define good and bad posture based on the local minimum. For the right wrist distribution, separating the distribution requires additional considerations- there is no clear minimum between the spread of higher and lower reviews.

4.4 Limitations

From reviewing the capture process for gathering subject data and classifications, we determined several limitations on using this data-set. In this section we highlight

some of these limitations.

Workstation Our machine was not able to calculate joint data at the maximum limit of 30 frames/second. Our resulting data-set provides information at a rate of approximately 14.5 frames/second. With an improved workstation, the joint data-set could be doubled over the same time-span. Given the amount of processing occurring on the workstation at any given time, the number of frames/second varies. To visualize this, we calculated the frames/second count across all captures, seen in Figure 4.5.

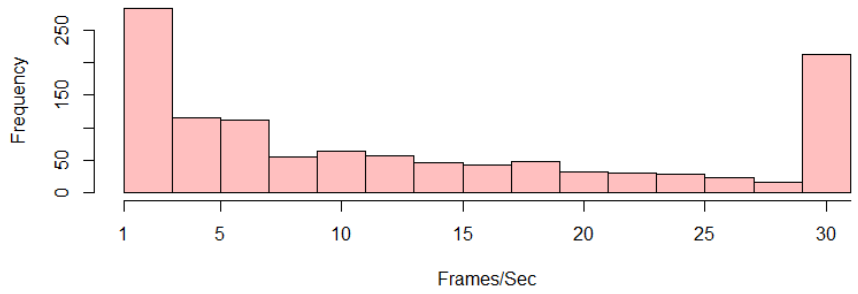


Figure 4.5: Frequency of frames/second recorded over all captures

Given the variable frame-rate, there is a possibility that the jitter removal methodology suggested in [3] may be less effective. In using this data-set we should perform additional jitter removal strategies that are less dependent on a constant number of frames recorded.

Bias towards positive review As per the human subjects form attached in the appendices, we prompted the user to make adjustments to their posture in order to include bad postures. As a limitation of the committee form, we only asked that subjects adjust one facet of their posture. On review, a better scenario would generate varied postural situations based on combinations of each of these facets. A median

and mean value of 6 suggests that within the classification subset our images tended to favor good wrist posture. In addition to this uneven distribution, the data-set does not contain any markers on posture based on prompted adjustments. In order to build a wider classification distribution this could be used in generating a selective group of classification candidates.

Limitations of rating storage In order to provide anonymity to voting and allow for rapid recording of data, we coalesced ratings based on the specific rating image, not based on the user whom provided the reviews. Having this information in the data-set would allow for users to apply normalization per voter before coalescing the data.

Limitations of crowd-sourced collection After compiling our rating data, we discovered that we collected on average two ratings per image. In our methodology, we described that the candidate images were provided to the users in a random order. To improve these results, we could have reduced the initial quantity of candidate images, or increased the number of user ratings. In result, this small list of crowd-sourced labels limits data processing capabilities from more intelligently determining classifications from ratings.

Chapter 5

BACK POSTURE CLASSIFICATION

In this section, we focus on using machine learning fundamentals to create a classifier for back posture. We filter joint data in order to reduce Kinect 'jitter'. We select several machine learning algorithms for classification, including SVM, Nearest Neighbor, Neural Network, and Random Forest. Finally, we contrast results and discuss limitations based on the setup.

5.1 Overview

To classify back posture, we collect back classification and joint positioning data from our data-set. In Figure 5.1, we outline the steps in our test system. In this system, we can use the resulting classifier in a similar system for determining back posture classifications.

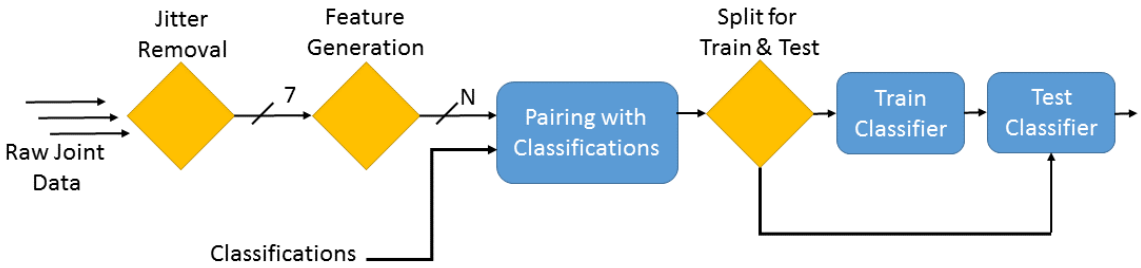


Figure 5.1: Overview of back classification test design

In each stage of Figure 5.1, we apply multiple techniques and compare their effectiveness. Specifically, during the jitter removal and training classifier stages, we employ several distinct methods. In the following sections, we describe methodology presented for each stage.

Python is employed in all stages prior to split for training and testing, including

some third party libraries for data processing. From the splitting of training and testing data stage, we use the statistical language R, including several packages provided within its package distribution utility, CRAN [27].

5.2 Joint Data Representation

From the data-set we collect joint location data. For back classification, we represent these joint captures per time-stamp, allowing multiple points to map to each joint value on a subject. Table 5.1 provides an example of the data format.

Table 5.1: Excerpt of data collected from capture

Data Point	X0	Y0	Z0	X1	Y1	Z1
04-31-08	167.739975	297.324554	1.271576	166.35199	224.966736	...
04-31-08	166.6492	296.660858	1.27524	165.994583	224.896545	...
04-31-08	166.667648	296.700684	1.27547	165.877975	225.013306	...
04-31-08	166.673431	296.776733	1.275406	165.781982	225.103134	...
04-31-08	166.527039	296.801544	1.275773	165.640686	225.158478	...
...	...	...	...	...	...	...

As mentioned in the limitations, this data is only 'approximate' due to 'jitter' recorded from the Kinect. Due to the heavy nature of the processing and limited resources, we have omitted some capture data that is missing a mapping of any data points within the feature space. To counteract this limitation, we employ several methods to determine actual joint locations given a subject time-stamp pair, outlined in the following section.

5.3 Jitter Removal

To process the raw joint information, first we apply filtering techniques to remove 'jitter' and reduce joint information over time into a single data point. We outline the methodology for the following four techniques:

1. Average all values of a single time-stamp
2. Single raw data point at a single time-stamp
3. Exponential Weighted Moving Average (EWMA) given a window
4. Median Filter

Supplementing the two filtering strategies described in 2.1.2.6, we include two 'rating count'-invariant strategies: Average all values at a single time-stamp and single raw data at a time-stamp. As discussed in the data-set limitations, Section 4.4, our classification data contains varying quantities of frames/second. To further visualize the variance discussed, we computed the number of frames/second limiting our data to only frames included within the classification data set for back posture, shown in Figure 5.2.

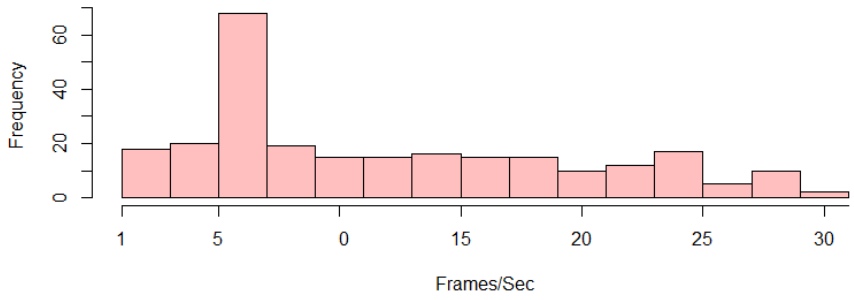


Figure 5.2: Frequency of frames/second recorded over back classification captures

To describe the effect of a varying frame/second count, for both Median and EWMA, discussed following, we must configure some time window for filtering data. Given that we usually configure a time window given some range of time that is static, it is difficult to determine how many second we are spanning in a non-uniform frames/second situation. To supplement these window-based filtering strategies, we also include two time-invariant implementations, Average all values at location and raw data at value.

5.3.1 Average All Values at a Single Time-Stamp

For this case if we are looking at Table 5.1, assuming the information is all available in Table 5.1, we average all of these values. In terms of varying frames/second we only observe data within a specific time frame.

5.3.2 Single Raw Data at a Single Time-Stamp

To benchmark results we also include the raw data results at the final data point recorded within a time-stamp. This acts as a control for gauging the results of our filtration techniques given a varying quantity of collected frames/second.

5.3.3 Exponential Weighted Moving Average

We employ the EWMA described in Section 2.1.2.6 using an implementation provided by the Python library pandas [22]. To build an EWMA, we configure a parameter 'span'. This parameter relates to λ described in Section 2.1.2.6 via the equation 5.1 from [26]

$$\lambda = \frac{2}{span + 1} \tag{5.1}$$

Using the span syntax, this allows us to make a '20 data point' span by configuring the span to be 20. For this project, we test span values of 5, 10, 20, and 50- equating to an average span of approximately 0.3, 1, 2, and 3.5 respectively given the average frames/second of the data set.

5.3.4 Median Filter

To apply a median filter, given a span sized N of previous values, the value at a specific point in time T is determined by the median of N values previously, including the current value. We test using a span of 5, 10, 20, and 50.

5.4 Collating Ratings

To process classifications we leverage the process used in generating our histogram distribution from the data-set from Figure 4.4. We ignore any unknown values, and take the mean of each classification image vote. Based on the two local maximum containing a local minimum between, and the premise provided to subjects for voting, we classify any posture above 5 as good posture and any value below as bad posture. We train classifiers based on approximately 103 bad posture and 168 good posture classifications.

5.5 Feature Generation

5.5.1 Back Features

To reduce bias on data, we generate specific features that relate to back posture alignment. Alternatively, we could have directly input joint data into training classifiers. To reduce variance based on spatial location, we opt to extract angles, and distance ratios from joint positions.

Though we have access to data for every joint collected for study in this paper, we focus on five main joints pertaining to back posture:

1. Neck
2. Spine shoulder
3. Shoulder right
4. Spine mid
5. Shoulder left

Figure 5.3 visually describes these locations.

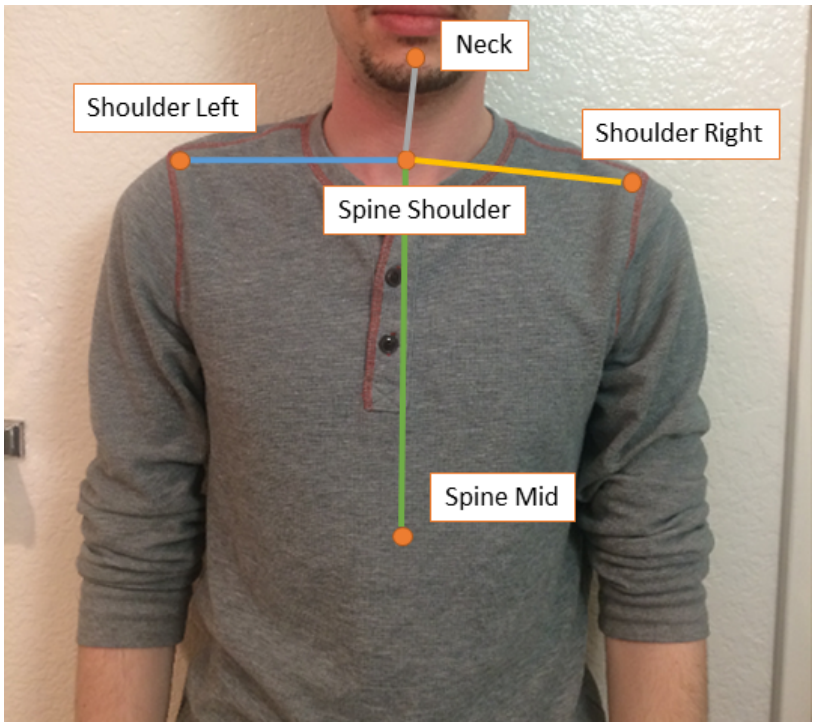
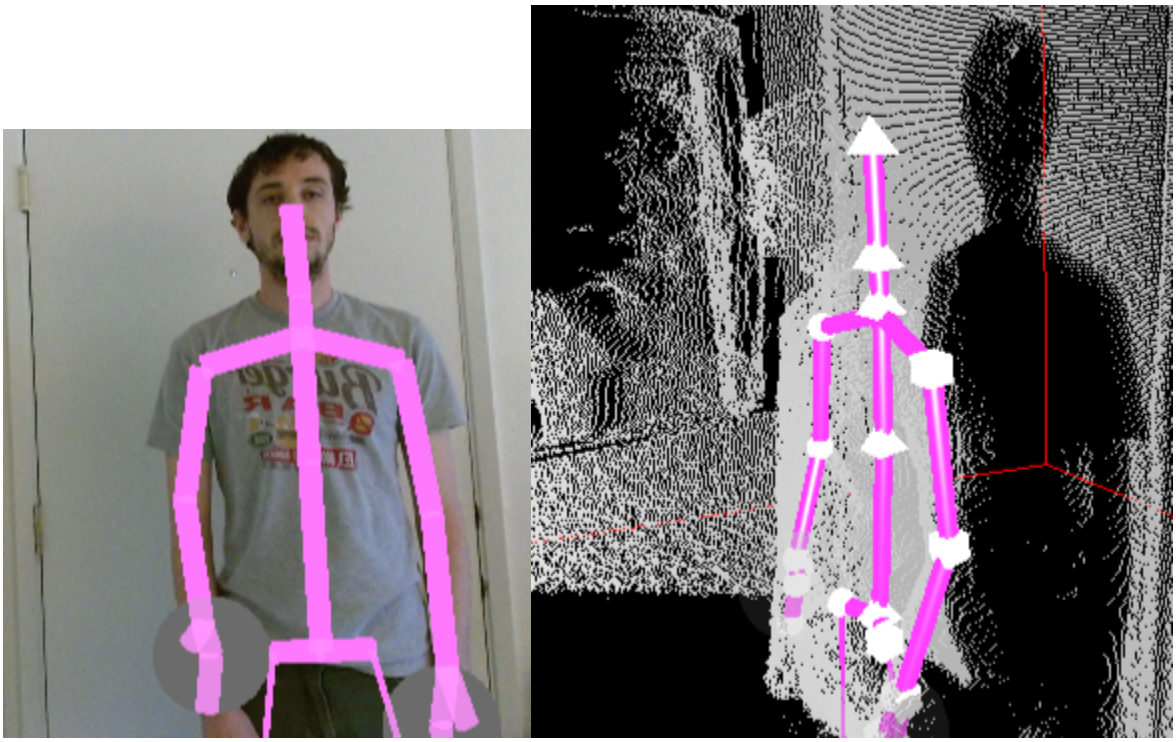


Figure 5.3: Subset of available joints from Kinect joint detection

In the following sections we describe simple mathematics employed to define these features and explore some examples of collected joint postural data.

5.5.1.1 Joint Posture Samples

In Figure 5.4 and Figure 5.5, we provide examples of joint captures for a subject exhibiting good and bad posture, respectively. In both images, we observe the two dimensional front-facing representation, and a three dimensional angled view. For the first subject, the user stands upright, with a straight neck alignment. This position is reflected in the back posture data, represented by the purple overlays. In the three dimensional image, the white dots represent tracked joints. As can be seen, from a side view the joints are aligned relatively flat with respect to the z-dimension.

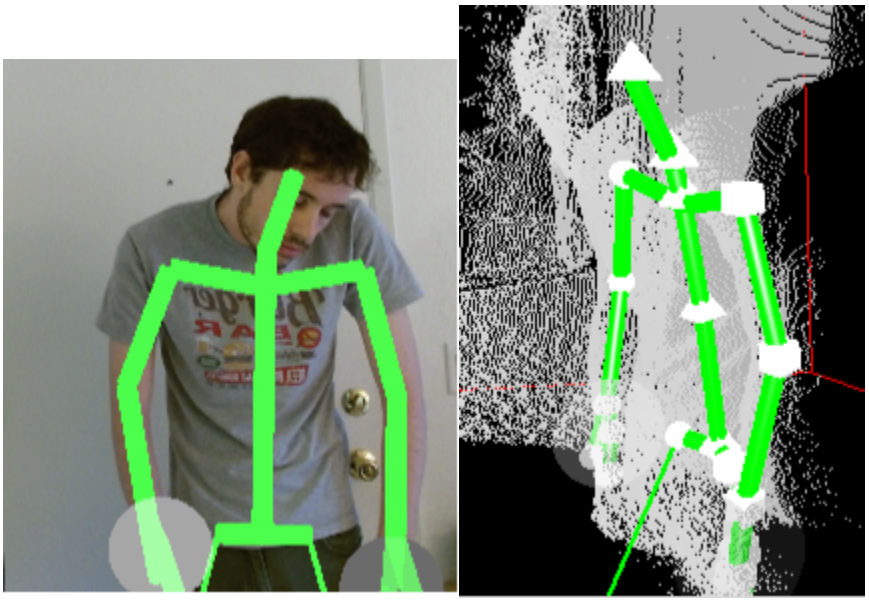


(a) 2-Dimensional view (b) 3-Dimensional view

Figure 5.4: Sample captures on subject with good posture

In Figure 5.5 we observe a subject exhibiting obvious bad posture as per our rules of posture classification, having neutral posture positions. From a z-dimensional space, the subject leans forward, showing a sloping joint location from back to shoulders to neck. Notably, the position of the shoulders with respect to the intersection of

the spine and shoulders is a different angle from Figure reffig:SampleJointCapturesGood. The neck joints also appear more compressed than the good capture. We use these changes in angles to differentiate classifications of good and bad posture.



(a) 2-Dimensional view (b) 3-Dimensional view

Figure 5.5: Sample captures on subject with bad posture

5.5.1.2 Angles

Within our feature space, we create three angles based on the joint positions in Figure 5.3, and name these features as following:

1. Angle:ShoulderLeft-SpineShoulder-Neck
2. Angle:ShoulderRight-SpineShoulder-Neck
3. Angle:Neck-SpineShoulder-SpineMid

We compute these angles using the Law of Cosines.

$$\cos(v) = \frac{A^2 + B^2 + C^2}{2AB} \tag{5.2}$$

Where A, for example in Angle:ShoulderLeft-SpineShoulder-Neck, is the distance between ShoulderLeft to SpineShoulder, B is the distance from ShoulderRight to SpineShoulder, and C is the distance from ShoulderLeft to ShoulderRight. All distance calculations use the euclidean distance. Given that a body in neutral position tends to indicate good posture, we would expect large deviations on these angles from neutral positions to indicate bad posture.

5.5.1.3 Ratios

We select two ratios based on joint positions to characterize compressive movement which might hinder neutral positioning- named as following:

1. Ratio:NeckSpineMid-NeckSpineShoulder
2. Ratio:ShoulderLeftShoulderRight-ShoulderLeftSpineShoulder

In order to compute ratios, we define distance between two joints using the Euclidean distance formula, provided in Equation 5.3:

$$Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{5.3}$$

And we define the ratio between the two joints:

$$Ratio = \frac{A}{B} \tag{5.4}$$

Where A, for example in Ratio:NeckSpineMid-NeckSpineShoulder, is the distance between Neck and SpineMid, and B is the distance between Neck and SpineShoulder. By computing the ratio, we allow for scale-invariance that would not be possible using only distance.

5.5.1.4 Summary of Joints Features

In total our feature space consists of five features: three angles and two ratios. We first will perform classifier training based on all five features, and later will apply feature selection to select an optimal subset before training.

5.5.2 Split and Scaling Data

To train our classifier, we use an 80-20 split of training-testing data. To do this we apply a built-in tool for the R package CARET, createDataPartition [16]. This tool allows us to randomly sample based on classification, giving an even relative distribution of each classification in both the training and testing set. For example, lets say we have a dataset with 200 positive classifications and 100 negative. If we would like to perform an 80-20 split on this data-set, via non-relative distribution it is possible that all 20% of testing data contains negative results. By splitting the distribution evenly, we allow 80 negative and 160 positive to consist of the training data, and 20 negative and 40 positive to consist of the testing data. By partitioning this way we ensure that all classification are represented while testing.

Before feeding this data into the classifier, we also scale all feature information into a range of [0, 1] using CARET preProcess method. This transformation ensures that distance-sensitive classifiers, such as Nearest-Neighbor machine learning tools, are not biased toward feature whose range of values is much larger than others.

5.5.3 Training Classifier

In building our classifiers, we heavily leverage the R CARET package, particularly the 'train' method [11]. This package leverages existing R classification implementations, allowing configuration for training total of 233 classification/regression models.

To use this method, we provide a model, for example, "rf", random forest, and we also provide tuning options and optimization controls. These tuning options are very broad, allowing the classification tool to enumerate possible configurations and determine the best result by comparing against metrics. These metrics are configured using the optimization controls.

In addition, we define by what method we would like to use for evaluating results, described via the 'optimization' controls. Some built-in tools offered are Bootstrap, repeated k-Fold Cross-Validation, and k-Fold Cross-Validation. To compare different methodologies, in our system we enumerate across these three optimization controls. In building our classifier training models, we rely on examples provided through the package documentation [16].

To provide a range of models, we select four distinct machine learning tools for classification on our back posture.

1. Support Vector Machine [15]
2. Neural Network [31]
3. K-Nearest Neighbor [31]
4. Random Forest [20]

Each model leverages tuning functionality provided by CARET. We apply all possible data input types, including all possible optimization control types to each of the selected machine learning models.

5.5.3.1 Support Vector Machine

To describe the SVM simply, the goal is to draw a dividing line between classifications given a feature space. We call the dividing line a hyper plane, where the nearest

training data to this line are known as support vectors. The minimum distance between any support vector and its dividing line is known as the margin. In training the SVM, the goal is to maximize this dividing line, known as the margin. The training algorithm leverages Lagrangian optimization in order to determine the result having the largest margin. In Figure 5.6, we provide a sample of an SVM result. In this result, we have a curved line drawing a boundary between two classification across two features, denoted by the x and y axis. In our implementation the result will be derived across five features.

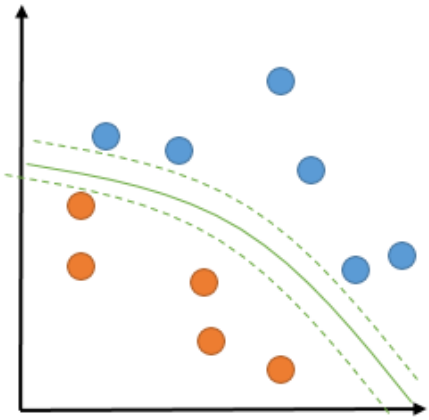


Figure 5.6: Sample SVM result

5.5.3.2 Neural Network

In the neural network machine learning design, our goal is to achieve classification results through simulated artificial neural networks. Within the design, each node is represented as a configured 'neuron'. These connected components, also known as perceptrons, are configured in a format similar to Figure 5.7. The input to these perceptrons are any N number of 'features' that have had weights applied to them. Once these weights are summed, they are passed into a stepwise function which can determine the neuron output.

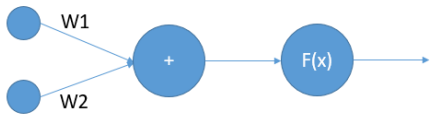


Figure 5.7: Sample perceptron

In the CARET implementation we leveraged, nnet, the design is of a simply feed-forward neural network. Within a neural network design, there are typically internal 'hidden' layers that perform additional transformations the data before deriving a result. In this implementation, there is a single layer, with a configurable count of number of neurons within this layer. In 5.8, we provide an example of a configured neural network. Orange nodes represent hidden neurons within the network, operating from left to right. The leftmost nodes are the input features, where the rightmost node represents the final output of classification.

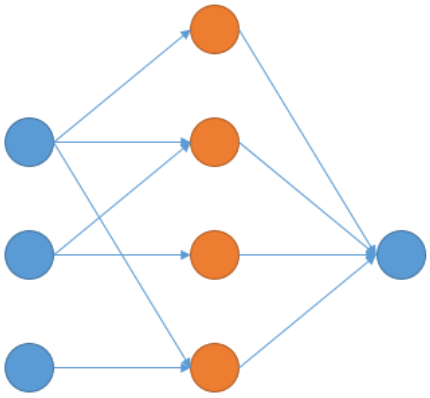


Figure 5.8: Sample neural network

5.5.3.3 K-Nearest Neighbor

For K-nearest neighbor, the goal is to use training data directly in determining what classification should be given to input data. This classifier operates purely on distance, selecting mode of the closest K neighbors as the assigned classification.

This algorithm requires scaled data, in order to properly determine distances cross feature space. In Figure 5.9, we provide a quick example of a green node that we are classifying across a feature space of two features. To note, given a K of 2 we would classify the item as blue, where with a K of 5 the classification would be orange.

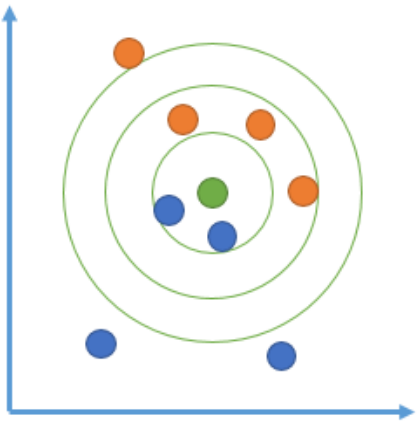


Figure 5.9: Sample K-nearest neighbor

5.5.3.4 Random Forest

In a random forest, the goal is to combine decision trees through a mode operation in order to classify results. To describe the behavior of decision trees, we can refer to the simple example in Figure 5.10. In Figure 5.10, we have two features, X1 and X2, and two classifications, orange and blue. Given we have a value of 1 for X1, and .5 for X2, at the initial root node we would branch to the right. In the next comparison, we observe that our X2 value of .5 is false for the conditional check of X2 $<$ .1. As such we branch to the left. The final classification is the orange node.

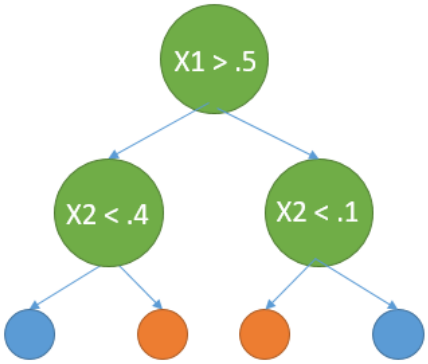


Figure 5.10: Sample decision tree

The random forest classifier is known as an ensemble method, requiring the result of multiple, individual decision tree classifications.

5.5.4 Evaluation

To test these configurations, we perform a set of predictions across the reserved training data. We compute a confusion matrix based on each prediction result, and compute three evaluation metrics: sensitivity, specificity, and kappa score. In our results we compile a list of the top 10 results for each metric. To combine these results, we assign a relative ranking based on the highest score in each metric: the highest score assigned, 1, second highest 2, and etc. Our final ranking metric assigns a number to each classify context based on the sum of the three metric ranks- where the highest ranks are those with the lowest score. We assume that each metric holds the same weight in deciding the highest-rank, so these metrics do not have scales.

5.6 Results

The following table provide the top 10 results given our specified evaluation criteria.

To define row results, we create a unique label based on contexts. Each context is labeled in the format **Control-Classifier-DataInputType**. These contexts are defined following.

5.6.1 Control

For control we refer to the training control used for tuning the classifier and in determining the best tuned parameters. The available controls are:

1. RCV, 10-Fold repeated cross-validation
2. CV, 10-Fold cross-validation
3. Boot, Bootstrap method

5.6.2 Classifier

For classifier we refer to the available machine learning classifiers. The available classifiers are:

1. KNN, K Nearest Neighbors
2. NN, Neural Network
3. SVM, Support Vector Machine
4. RF, Random Forest

5.6.3 DataInputType

For this we refer to the available filtering methods used in removing jitter from joint information. The available methods are:

1. EWMA5/10/20/50, EWMA filter based on span

2. Raw, Raw filter based on single data point

3. Average, Average filter based on all within timestamp

4. Med5/10/20/50, Median filter based on span

We also have added a runtime column, RT, describing the performance of the classifier against testing data averaged over the total size of the testing data. First we compare results on sensitivity- in Table 5.4, we show the top ten contextual results.

Table 5.2: Top 10 results sorted by Sensitivity

	TP	FP	FN	TN	Sens.	Spec.	Kappa	RT(ms)
Boot-SVM-Raw	32	1	13	7	0.97	0.35	0.36	0.35
RCV-SVM-EWMA50	32	1	14	6	0.97	0.30	0.31	0.29
CV-SVM-EWMA50	32	1	14	6	0.97	0.30	0.31	0.27
Boot-SVM-EWMA50	32	1	15	5	0.97	0.25	0.25	0.29
CV-SVM-Raw	32	1	15	5	0.97	0.25	0.25	0.30
CV-SVM-Average	31	2	13	7	0.94	0.35	0.32	0.27
Boot-SVM-EWMA10	31	2	13	7	0.94	0.35	0.32	0.29
CV-SVM-EWMA10	31	2	13	7	0.94	0.35	0.32	0.27
RCV-SVM-Raw	31	2	13	7	0.94	0.35	0.32	0.31
RCV-SVM-EWMA10	31	2	14	6	0.94	0.30	0.27	0.29

See page 69 for description of context labels

In case of sensitivity, the highest performing classifier across the board is SVM. At a sensitivity of 0.97, the Bootstrap method, and using a plus minus range of 50 results in 32 true-positive results with only 1 false positive. This model used an

automatically configured radial basis kernel function with a sigma of 0.2 and a C of 1.25 By control, we see that CV, RCV, and Boot all appear in the list for top ten in sensitivity, the most being CV. For filtering strategies, the highest quantity of top-10 results appears to be using the EWMA filtration.

For specificity, in Table 5.3, the highest performing classifier is a Neural Network, given Bootstrap and a Median Filter size 10. This model was finally configured with a size of 8 and decay of 0.004. At a highest threshold we observe a sensitivity of 0.70. Neural Network, K Nearest Neighbors, and Random Forest all comprise the list of the top 10 results, with a mixture of controls. Plus minus range of 5 constitutes for the highest number of data input types. Compared with the results for specificity, these results seem rather low. To understand the misclassified false-positives, we provide two examples of the misclassified bad postures in Figure 5.11.

Table 5.3: Top 10 results sorted by Specificity

	TP	FP	FN	TN	Sens.	Spec.	Kappa	RT(ms)
Boot-NN-Med10	24	9	6	14	0.73	0.70	0.42	0.06
RCV-KNN-EWMA50	29	4	8	12	0.88	0.60	0.50	0.06
CV-KNN-EWMA50	29	4	8	12	0.88	0.60	0.50	0.06
RCV-KNN-Med50	28	5	8	12	0.85	0.60	0.46	0.06
Boot-KNN-Med50	28	5	8	12	0.85	0.60	0.46	0.07
RCV-KNN-Med20	27	6	8	12	0.82	0.60	0.43	0.07
CV-RF-Med20	27	6	8	12	0.82	0.60	0.43	0.14
CV-NN-Med5	27	6	8	12	0.82	0.60	0.43	0.07
Boot-KNN-Med5	27	6	8	12	0.82	0.60	0.43	0.07
RCV-RF-Med20	26	7	8	12	0.79	0.60	0.39	0.14

See page 69 for description of context labels

In Figure 5.11a, crowd-source voting determine that the back posture deserves the lowest rating, 1. To understand this misclassification, we need to consider the limitations on this dataset. It is likely that within the 271 classification images, that this body position is unique given the angle the subject is bending over. Provided the data-set covered a larger range of motion of the subject, this posture may have been classified correctly with the given feature space. In Figure 5.11b, the classified rating was placed at 5. Given that this position rates on the borderline between good and bad posture, it seems feasible that the posture could be misinterpreted by a classifier. Due to the limitation of the dataset in quantity of crowd-source votes, it is also possible that the current 'bad' rating be changed to 'good'.

(a) FP of Rating 1 (b) FP of Rating 5

Figure 5.11: Samples of false positives for Boot-NN-Med10

Figure 5.4 combines results for the Top 10 classifiers based on Kappa score.

Table 5.4: Top 10 results sorted by Kappa

	TP	FP	FN	TN	Sens.	Spec.	Kappa	RT(ms)
RCV-KNN-EWMA50	29	4	8	12	0.88	0.60	0.50	0.06
CV-KNN-EWMA50	29	4	8	12	0.88	0.60	0.50	0.06
RCV-KNN-Med50	28	5	8	12	0.85	0.60	0.46	0.06
Boot-KNN-Med50	28	5	8	12	0.85	0.60	0.46	0.07
RCV-SVM-Med50	29	4	9	11	0.88	0.55	0.45	0.32
CV-SVM-Med50	29	4	9	11	0.88	0.55	0.45	0.29
RCV-KNN-Med20	27	6	8	12	0.82	0.60	0.43	0.07
CV-RF-Med20	27	6	8	12	0.82	0.60	0.43	0.14
CV-NN-Med5	27	6	8	12	0.82	0.60	0.43	0.07
Boot-KNN-Med5	27	6	8	12	0.82	0.60	0.43	0.07

See page 69 for description of context labels

The highest kappa score ranks at 0.50, with a score suggesting moderate agreement between the predicted and observed values. We observe this same kappa score for K nearest neighbors on a EWMA of 50 between all two of the three possible control types. For this model, the CARET package selected a k value of 5. The top four results all are k nearest neighbors- suggesting that given this metric they outperform the standard configurations of the other classifiers. A median filter data input type is present in 8 of the top 10 results.

To generate final rankings, we compute a score based on the sum of the rankings of each metric described previously. Figure 5.5 displays our final results for classification.

Table 5.5: Top 10 results sorted by Ranking

	TP	FP	FN	TN	Sens.	Spec.	Kappa	RT(ms)
RCV-KNN-EWMA50	29	4	8	12	0.88	0.60	0.50	0.06
CV-KNN-EWMA50	29	4	8	12	0.88	0.60	0.50	0.06
RCV-SVM-Med50	29	4	9	11	0.88	0.55	0.45	0.07
CV-SVM-Med50	29	4	9	11	0.88	0.55	0.45	0.29
RCV-KNN-Med50	28	5	8	12	0.85	0.60	0.46	0.07
Boot-KNN-Med50	28	5	8	12	0.85	0.60	0.46	0.07
Boot-SVM-Med50	29	4	10	10	0.88	0.50	0.40	0.37
RCV-KNN-Med20	27	6	8	12	0.82	0.60	0.43	0.07
CV-RF-Med20	27	6	8	12	0.82	0.60	0.43	0.14
CV-NN-Med5	27	6	8	12	0.82	0.60	0.43	0.07

See page 69 for description of context labels

Our highest ranking classifiers are the two highest performing kappa metric evaluations: k nearest neighbor with a EWMA of 50. Both median and EWMA data input type capture all of the top 10 results, suggesting that raw and average provide less useful information in capturing classification results given our scoring criteria. The final tuned parameter for our K-nearest-neighbor value was K=5.

Another important note is the absence of our random forest classification from within the highest of any of the top 10 results. This does not necessarily mean the random forest classifier itself is a poor tool for classification, rather that finer tuning is required in order to get better results given a random forest design. Based on the final classifier we select, K nearest-neighbors at a EWMA of 50, we can correctly detect 88% of true positive results in an image. For evaluating posture on subjects, if a student is evaluated as having a bad back posture, it is likely that this evaluation

is true. The highest specificity we reached was at a 60% rate. For each student we detect that truly has bad back posture, there is slightly above a 50% chance that this will be considered good posture. In the next section we discuss some limitations to these results and remark on some considerations that can further improve specificity.

5.6.4 Scoring based on Discriminative Feature Selection

Based on instructions from [5], we can leverage CARET in determining feature rankings and accuracy given N number of features. To determine rankings based on feature, we use built-in Recursive Feature Elimination. This benefits machine learning algorithm by reducing the feature space dimensionality, allowing for quicker training time and simplified models. To determine accuracy given the RFE algorithm, we provide all classification data given all feature spaces. In the returned model, we observe:

1. A list of features sorted by RFE determined importance
2. A plot of accuracy over an increasing number of used variables

After running this algorithm across our data set, RFE provides the following importance ranking, highest to lowest:

1. Ratio:NeckSpineMid-NeckSpineShoulder
2. Ratio:ShoulderLeftShoulderRight-ShoulderLeftSpineShoulder
3. Angle:ShoulderRight-SpineShoulder-Neck
4. Angle:ShoulderLeft-SpineShoulder-Neck
5. Angle:Neck-Spineshoulder-SpineMid

Given this ordering, the RFE model uses cross-validation to select the number of variables based on highest accuracy, shown in Figure 5.12.

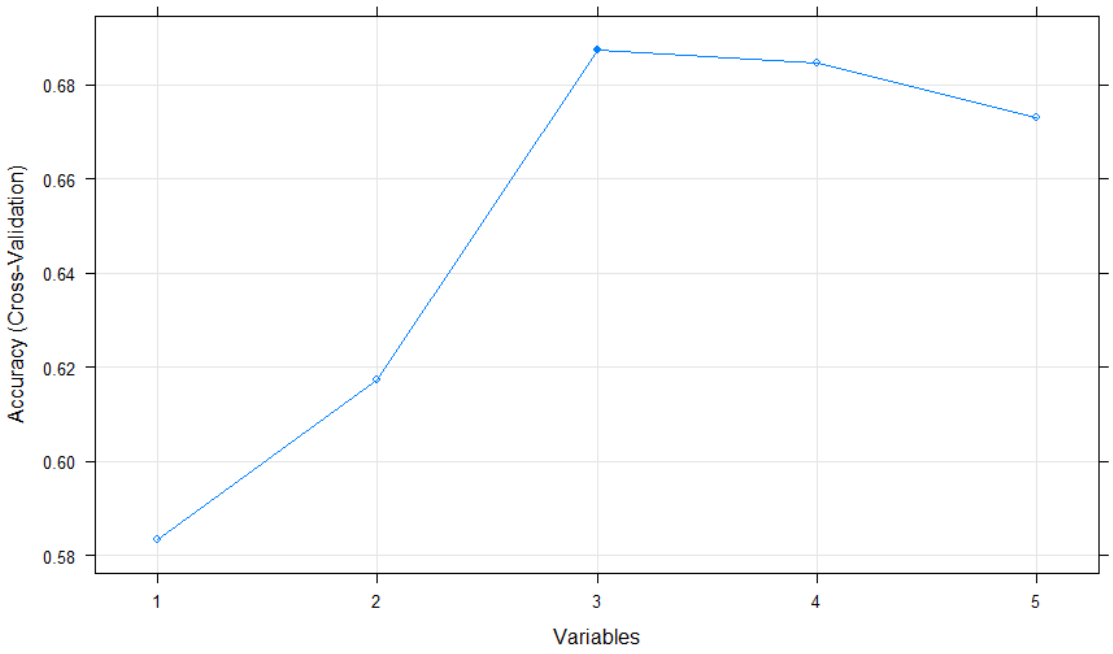


Figure 5.12: Accuracy determined by RFE given quantity of a subset of sorted features

In Figure 5.12, the RFE algorithm determines that using three features provides the highest accuracy - Ratio:NeckSpineMid-NeckSpineShoulder, Ratio:ShoulderLeft ShoulderRight-ShoulderLeftSpineShoulder, and Angle:ShoulderRight-SpineShoulder-Neck. To compare this result with our previous feature space, we re-ran our scoring system given this subset of features. We include results for the top 5 scoring classifiers in Table 5.6

Table 5.6: Top 5 scores after using selected 3 features

	TP	FP	FN	TN	Sens.	Spec.	Kappa	RT(ms)
Boot-RF-Raw	30	3	9	11	0.91	0.55	0.49	0.00014
CV-RF-Raw	29	4	8	12	0.88	0.60	0.50	0.00015
CV-NN-Med10	28	5	8	12	0.85	0.60	0.46	0.00006
Boot-KNN-Med10	28	5	8	12	0.85	0.60	0.46	0.00007
CV-RF-Med50	28	5	8	12	0.85	0.60	0.46	0.00012

See page 69 for description of context labels

Comparing these results with our scored results in Figure 5.5, our top classifier for Boot-RF-Raw has nothing in common with the top selected classifier of RCV-KNN-EWMA50. In investigating why we observe Random Forest within the top 5 given that its performance in our prior ranking was low, we revisit the recursive feature elimination set-up provided by [5]. In computing feature elimination parameters, the instructions suggest using a Random Forest function configuration internally. Thus we provide additional tuning specifically for Random Forest. As mentioned earlier, RF may require finer tuning to present better results. In performing feature elimination we create a new tuning functionality. The resulting RF tuned a parameter 'mtry' to 0.35, defined as, "Number of variables randomly sampled as candidates at each split" [20]. Given this tuning we achieve results with a sensitivity of 0.91, higher than the previous top scoring sensitivity at 0.88. In these results we show that given additional tuning, it is possible to further improve classifier results. In addition to this, run-time increases dramatically with the reduce feature space across all three of Random Forest, Neural Network, and KNN by reducing the feature space. This highlights the usefulness of performing feature elimination, in increasing run-time performance by magnitudes of 10

5.7 Limitations

Though we were able to achieve a high specificity, our resulting classifiers had a low sensitivity and by association a lower kappa score. Some identified limitations for these machine learning classifiers are:

5.7.1 Resulting limitations from data

Because we used the generated data-set referenced in the prior chapter, any limitations to the generate joints and classifications will directly apply to our back classification procedures. In containing more varied negative data and labeled prompted data, the machine learning could produce more reliable results. Given a larger data set, we could apply broader practices in determining how to divide 10-scale ratings into two class posture classifications.

5.7.2 Limitations to default tuning

These classifiers were trained using default tuning practices provided for out-of-the-box implementations of machine learning schemes. It is possible these results can be further tuned on research into each specific technique

5.7.3 Limitations from breadth of techniques

The CARET machine learning tool offers 182 separate models specifically for classification and 233 in total [16]. Of these models we selected 4, based on a spread of techniques. If we were to enumerate this set, it is likely that there is at least one technique better than the four we surveyed.

Chapter 6

TOWARDS WRIST LOCATION

To fulfill our defined objectives we ideally would like to allow for multi-postural classification. In this section we focus on wrist posture. As mentioned in the Kinect background, Section 2.1.2.5, we cannot use joint detection on wrists due to the camera position. In order to classify, first we must location wrists in an image. For this work we focus only on locating wrists. In future work we describe methods to iterate on the wrist locations.

6.1 Overview

Figure 6.1 portrays the steps followed in testing a hand-wrist locating scheme. To develop this system we perform a series of iterative improvements, where the goal is to maximize wrist detections and minimize false positive results. Within our iterative improvement algorthim we employ cascade object detection, provided as a tool with Matlab's Computer Vision System Toolbox.

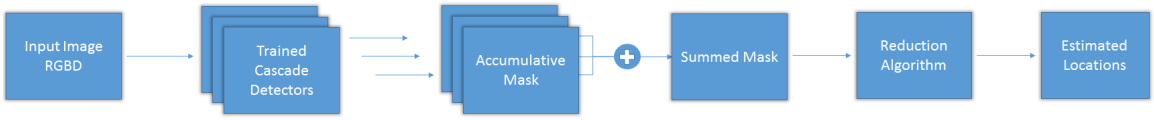


Figure 6.1: Overall flowchart of testing setup

We describe each stage in Figure 6.1 in further detail as we develop our iterative improvement. During the process we generate results on the dataset. Within the results portion of this work we compare the result of each additional iterative result.

6.1.1 Dataset Generation

For training, we select approximately 75%, from our data set via random sampling. After removing noisy images, those including multiple people, or subjects outside of the viewing window, we further limit the data-set to 705 images. In these images we identify 1352 regions containing hands manually. For 647 of these images we have both wrists visible, where in 58 we observe one obscured wrist based on the sitting position. For cascade detection these 1352 hands describe positive regions in wrist detection. Negative samples are defined automatically as negative regions of the testing image.

For testing, we randomly sample the remaining 25% of images. We randomly select 116 images, approximately 10% of the total quantity of data. We locate 227 hands regions from the 116 images.

6.1.2 Successful Wrist Location Definition

For testing result locations, we define a successful true positive wrist detection when a predicted wrist position is within the region defined by a located hand in the testing set. A false positive wrist location lies outside of the testing region. Figure 6.2 describes example of positive training data across subjects.

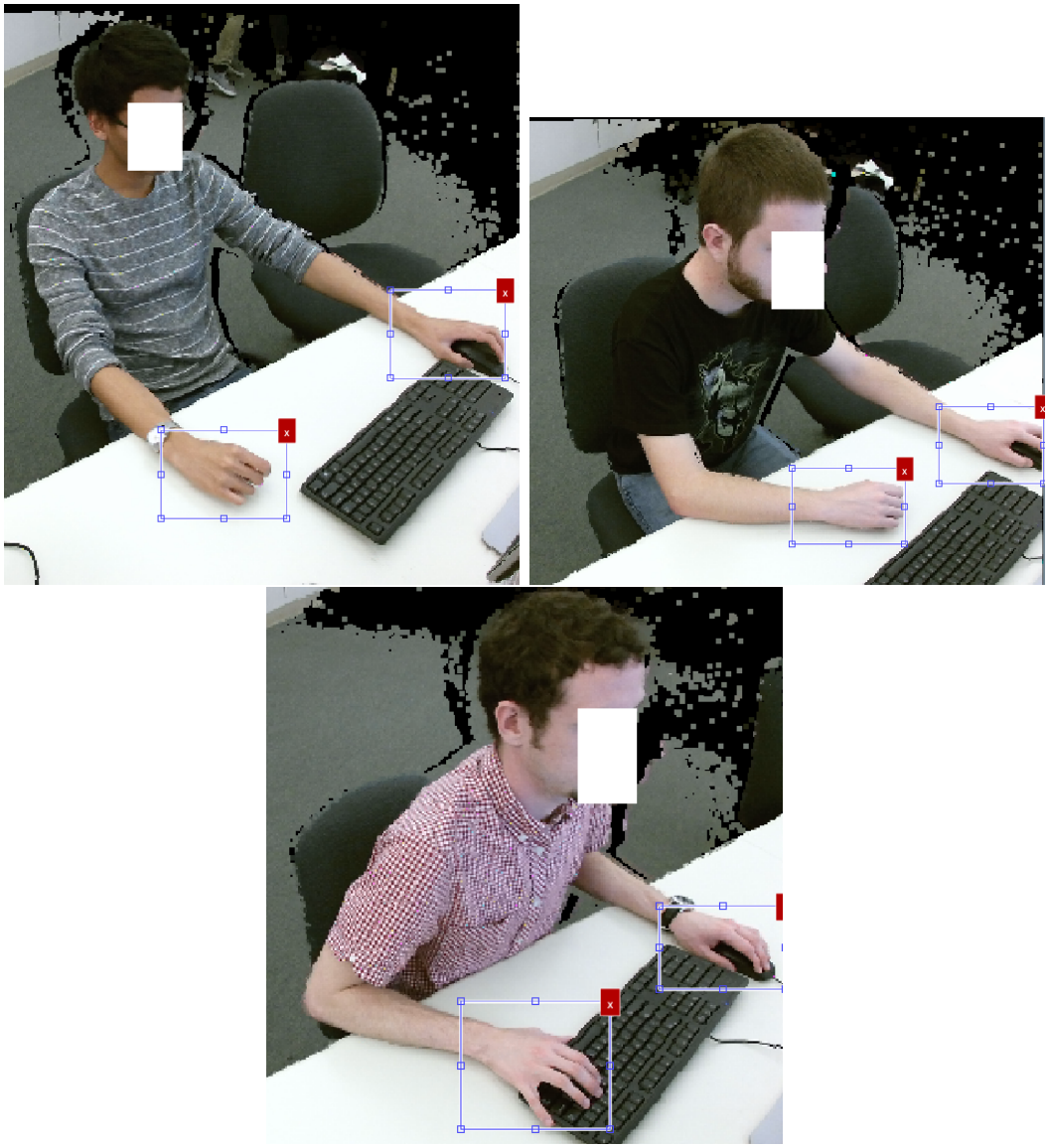


Figure 6.2: Examples of positive training data across three subjects

6.2 Luminance Detector

For deriving a detection scheme, we first look at the Luminance band. For a shorthand naming, we refer to Luminance as Lum within figures. To generate the band, we use the data conversion referenced in YCbCr color space in Section 2.2.1.2. We apply the final chosen detection method to alternate bands of image data following

this preliminary test.

6.2.1 Simple Cascade Run

During a simple cascade detection run, the Matlab cascade detection tool provides a square sub-region, known as a bounding-box, as the location of a detected object. To fit this result within the confines of our definition of successful wrist location, we compute the centroid of the detected region as the predicted location of our computed wrist. In Table 6.1, we provide preliminary results on computing a cascade detector using HOG feature types.

Table 6.1: Results of initial cascade detector on 116 test images

	Lum Results
False Positive	416
True Positive	214
False Negative	13
Time/Image(s)	0.13

Even in a basic configuration, the cascade detector predicts the majority of wrist location correctly, at a true positive rate, TPR, of 94.27%. In incidence to this high success detection, we also encounter very high numbers of false positives, at approximately four per image. Though we already know at most two hands will be in any test sample, in multiple circumstances the detector there are far more present. In Figure 6.3, we provide two images of subjects which the cascade detection discovers more hands than present. To be more precise, we observe nine hands in the first image, denoted by green dots. Though we predic two true-positive points within the hand bounding-boxes, we also see seven false positives. On the second, we detect four hands- two being false positives.

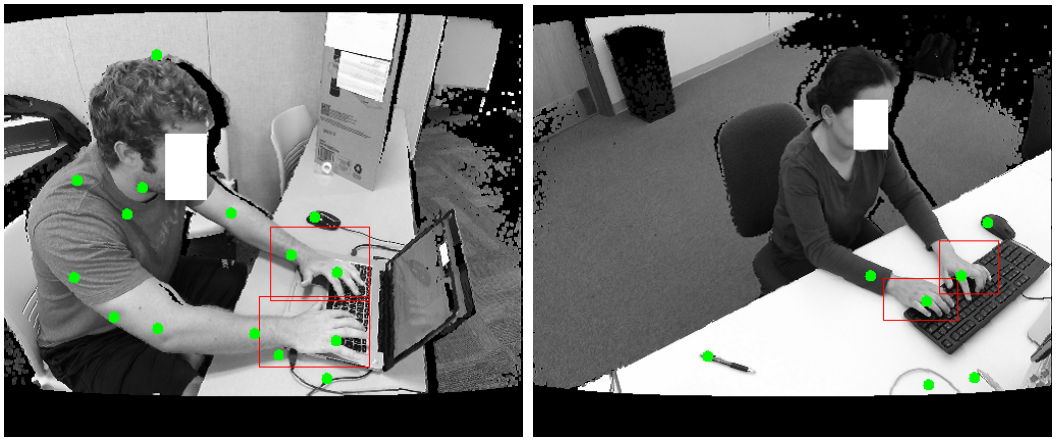


Figure 6.3: Two results of cascade detection using simple detection

Though we can consider a TPR of 94.27% a good result of the detector, the high FP count is not. These statistics are used as a baseline for comparing future results in our iterative algorithm. To build a better detector, we must maintain a high TPR but reduce the number of FP results.

6.2.2 Comparing Different Feature Spaces

The Matlab cascade detection tool offers three choices for feature space: LBP, HOG, and Haar. On experimenting with each feature space, we determined that computing Haar features on the test machine, described in Section 4.1.3, was infeasible. Creating Haar-like wavelets requires more computationally complex calculations than HOG and LBP. To provide some examples in training time given the workstation, for HOG the training time finishes within a scale of minutes. For LBP training time completes on a scale of hours. For Haar-like wavelet the training time operates on the order of days. After two days of running the Haar detector without even the first stage of iteration complete, we ceased training the detector. We consider only a comparison between the HOG and LBP feature spaces.

6.2.3 Simple Results of HOG Feature Space

We configure a HOG detector scheme, described in Section 2.4.1, using a ratio of 10x negative sampling to positive regions. Using this ration for every positive sample image, given that unselected regions are negative select 10 random regions for usage as negative samples. Given we have 705, this results in 7050 negative training samples. In Figure 6.4, we collect the results of the trained detector. This training process takes approximately 5 minutes.

Figure 6.4: Sample results for HOG feature type detection

As shown in Figure 6.4, for each image, the detector predicts both wrists successfully. In addition to this, we have many false positive results. Based on the sample images, we observe clustering around skin region, those with a sharp contrast in color, and around objects- namely mice.

6.2.4 Simple Results of LBP Feature Space

Next, we configure a LBP detector scheme with a 10x negative sampling rate. Using the same configuration, we trained the detector in approximately 8 hours. We provide sample predictions in Figure 6.5.

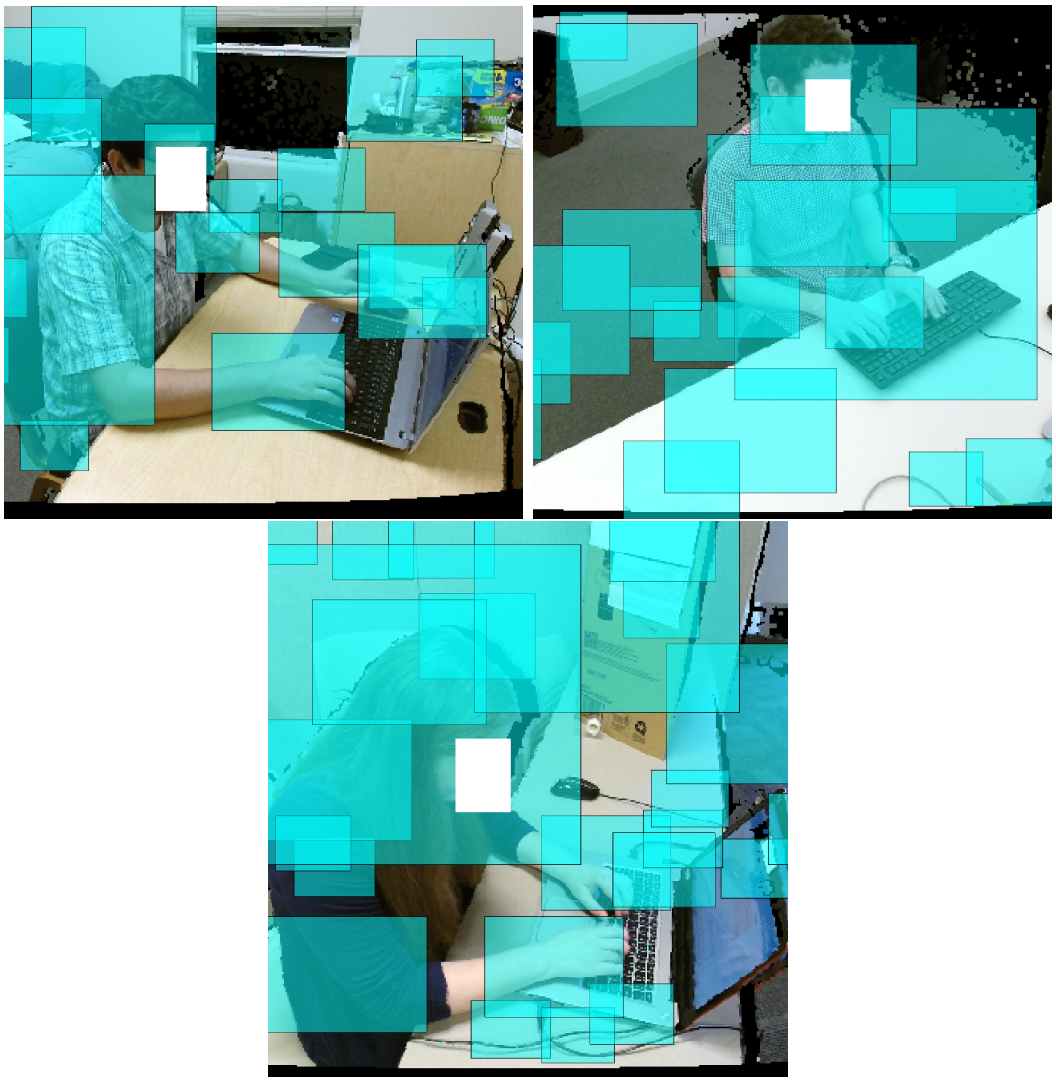


Figure 6.5: Sample results for LBP feature type detection

For the LBP feature space, it seems like there is very little visual 'rhyme' to the detection results. Some 'positive' regions contain little information present. Given the samples in Figure 6.5, we observe more randomness in classifying posture, including additional false positives within the carpets.

6.2.5 Conclusions on Cascade Detector Feature Type

Given the training time of LBP versus HOG, we would expect more immediate benefits. However, comparing the results of the training data, detection results seem more spurious for the LBP space.

Additionally, considering training time, the HOG space offers a huge advantage in timing. Given that LBP requires 8 hours, and HOG requires 5 minutes, HOG offers a clear advantage in rapid development.

We select the HOG feature space to continue our iterative improvements. The HOG space offers a higher potential in discrimination on detection and also requires a far lower time footprint.

6.3 FPR Reduction Cascade Algorithm

Matlab's simple cascade detection tool provides a 'merge-threshold' option, allowing the user to configure at what granularity to expect that two regions are part of the same classification. In the raw representation, Matlab Cascade detection represents classified locations at the pixel location. They coalesce these ratings based on the merge-threshold value. Given our knowledge on the data-set, we opt to define our own merge scheme. By turning off the merge threshold feature we can treat cascade results as representative clusters of hand-regions. We then custom processing techniques to transform these positive clusters into detections.

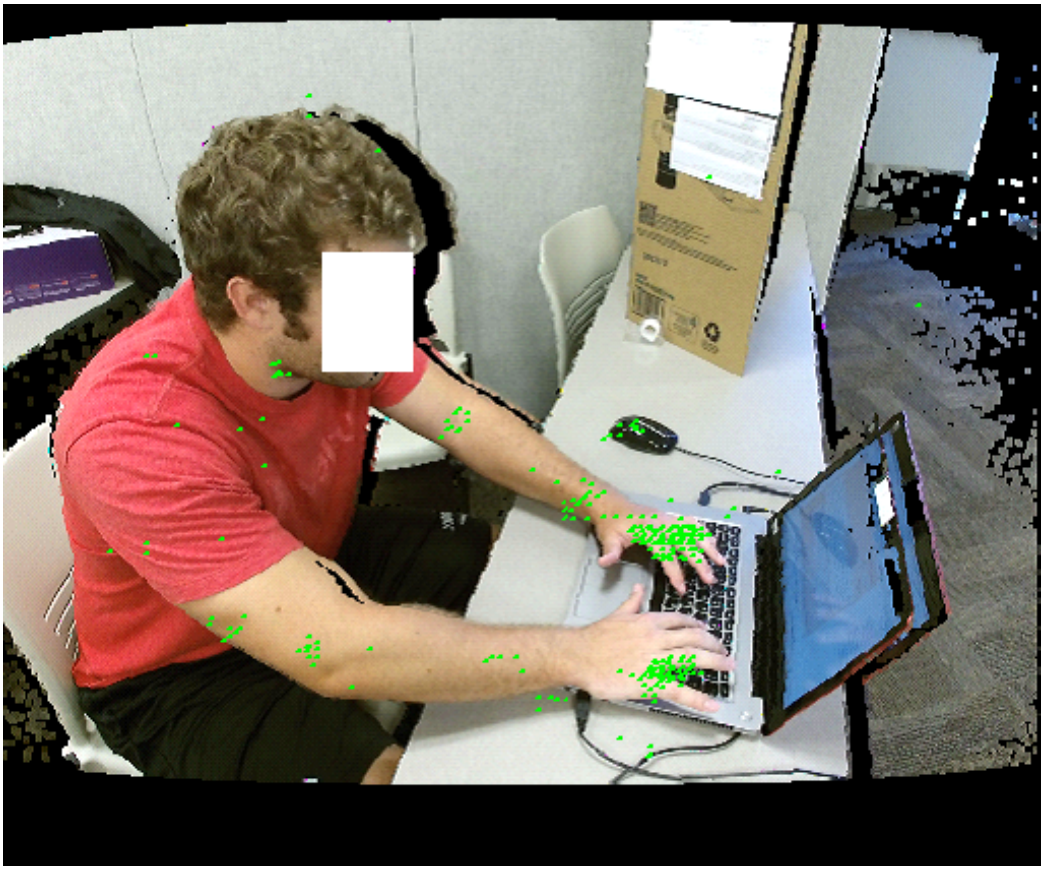
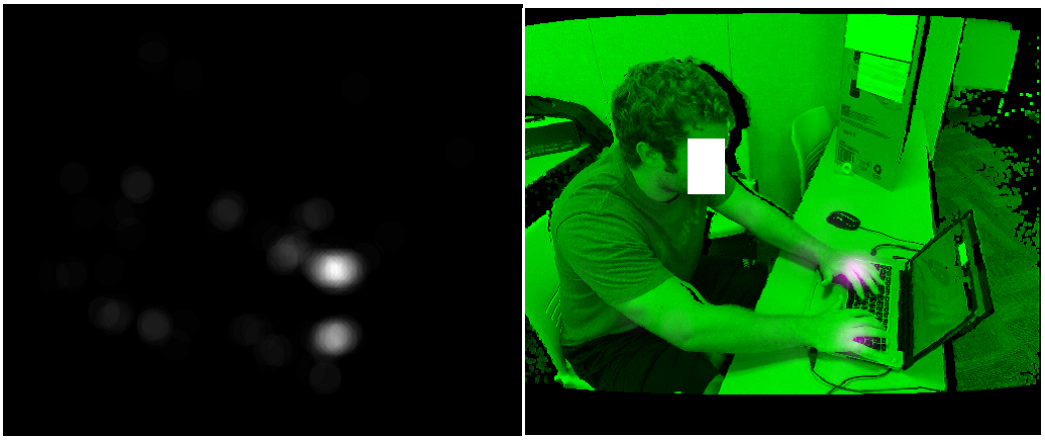


Figure 6.6: Clusters of detected regions overlaid on original image

In Figure 6.6, we provide an example of the cascade detector with overlaid clusters. Initially the cascade detector returns small bounded box areas, which are converted to points based on the centroid. By observing Figure 6.6, we see that pixels the cascade detector identifies as detected locations are cluster closely surrounding hands. In contrast, on regions with false positive detections we find smaller clustering. First we want to merge these clusters and retain information on number of data points surrounding. To do so, we convert each point into a mask by applying dilation by a structuring element with a radius of 16 pixels. We selected 16 based on the approximate size of a hand within the image. Next, we sum these masks into an accumulative mask, where higher numbers represent higher quantities of overlapping points. Figure 6.7 provides an example of this accumulation.



(a) Accumulative mask results (b) Overlay of mask on original image

Figure 6.7: Example of accumulative mask results

Next, we determine locations of the highest local regions, or peaks of the mask. To do so, we use Matlab function imregionalmax which computes the centroid location of local peaks. Each peak that imregionalmax predicts will be considered a hand location. In 6.2 we provide the compiled results.

Table 6.2: Results of no-merge accumulative on 116 test images

	All Peaks
False Positive	702
True Positive	226
False Negative	1
Time/Image(s)	0.44

Using the no merge-threshold technique, we improve the TPR from 94.27% to 99.59%, but also increase the FP count from 416 to 702. Despite the maintaining a high TPR, we ideally want to reduce FP count.

Next in our iterative improvement, we perform erosion and threshold operations on the clusters defined by the accumulative mask results. We ignore any accumulated

values less than fifty percent of the largest peak then apply morphological erosion to 'flatten' the remaining maximum values into a plateau. Finally, we leverage imregionalmax to determine locations of the new peaks, sorted based on their value. To further restrict the number of peaks observed, we select the largest N number of peaks within the image. We compile the results of choosing from Top 5 to Top 1 peaks into Table 6.3.

Table 6.3: Results of choosing top quantity of peaks from accumulative mask as coordinates

	Top 5	**Top 4**	**Top 3**	**Top 2**	**Top 1**
False Positive	301	200	103	26	2
True Positive	222	219	215	206	114
False Negative	5	8	12	21	113
Time/Image(s)	0.42	0.49	0.44	0.52	0.40

At a best result for reduction in FP count, we select the top 1 value in an image. Based on our knowledge of the data-set, this almost always will leave one hand out of prediction, resulting in a large jump in quantity of false negatives. Using the top five peaks within an accumulative image, we observe a higher TP rating, but also a higher FP count. To better visualize the effect of selecting N number of peaks, we compiled these results into Figure 6.8.

Figure 6.8: False-positive, false-negative counts across varying subsets of max peak values

In the top 5 detector, at a rate of approximately 5 identifications per image, the FP count is already lower than our original 'merge-threshold' method. Additionally, the TPR in the top 5 improves from the original Luminance of 94.27 % to 97.80 %. Across the spectrum of the top 5 values, as number of max peaks chosen decreases, the FP count decreases. Based on a more discriminate peak selection, the TPR reduces. These values experience a cross-over at a count of top two peaks selected. Based on the results observed, we choose to continue iterating using a top 2 peak detection scheme. This also intuitively makes sense, given that we expect at most two hands within any given image.

6.4 TPR Improvement via Camera Location Algorithm

Resulting from the camera setup in our data-set, for all data collected from subjects, one arm is always closer and therefore larger within the image. Due to this, we observe more peak regions being discovered on the nearest arm. Whenever both peaks are detected on a single arm, the FP count and FN count both increase, giving us worsened results. Based on our knowledge of the camera location, we can apply

geometry to reduce the number of times that this situation occurs. We focus on two properties of hand coordinate locations within the image:

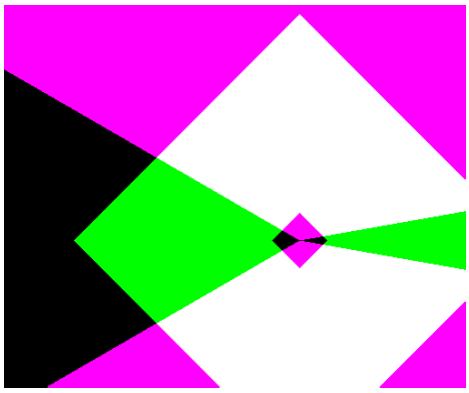
1. coordinate distance
2. relative coordinate angle

In this algorithm, we assume the largest peak in an image is a hand. To determine the next, we speculate candidate peaks based both on point distance and angle relative to the first peak.

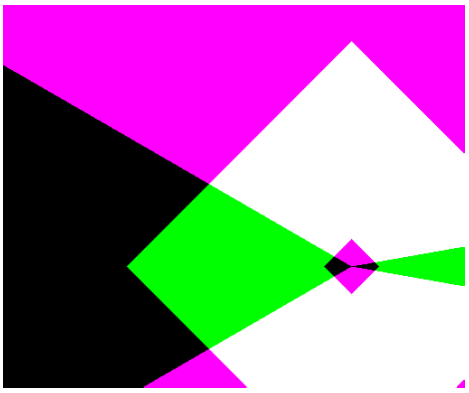
To determine an angle threshold, we compute a four-quadrant degree angle from the first peak. We compute the coordinate angle, θ for a proposed coordinate, P, and the maximum coordinate, M using a built-in Matlab four-quadrant inverse tangent formula, $d_x = M_x - P_x$, $d_y = M_y - P_y$. We determine an acceptable coordinate angle of $-10 < \theta < 150, 10 < \theta < 150$ by observing physical properties of the data-set.

For determining the point distance, we consider that identified hand coordinates should be relatively separated in the image. Based on the distance from the Kinect camera, we choose minimum city-block distance of 10, defining city-block as the distance being the sum of the vertical and horizontal distances between two points, A and B. For the maximum, we expect both hands are within the distance of the desk. As such, we select a maximum distance of 250 pixels.

To use this geometrically limiting methodology, first we determine the highest peak value. Next, we filter out only peaks falling in this geometric range. Finally, we select the highest remaining peak. We provide some sample images of the geometric filter in Figure 6.9.

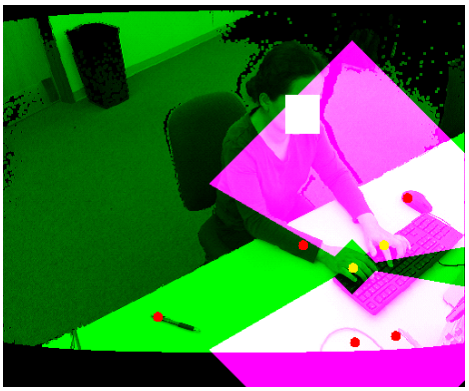


(a) Acceptable coordinate point locations, denoted by white region



(b) Acceptable coordinate point locations, denoted by white region

(c) Overlay of intersecting region, Selected points yellow



(d) Overlay of intersecting region, Selected points yellow

Figure 6.9: TPR reduction algorithm using new location criteria

In Table 6.4, we provide final results with the applied algorithm.

Table 6.4: Results of TPR improving algorithm across detectors

	Initial Lum.	**Top 2**	**Final Lum.**
False Positive	416	26	16
True Positive	214	206	216
False Negative	13	21	11
Time/Image(s)	0.13	0.51	0.45

Adding geometric data, both the TPR and FP count for locating hand regions improved. By applying both the top two peaks detector and the final detector, we decrease the approximate FP count decreased from 3.58 per image to 0.22 and 0.14 respectively. In the final luminance detector we additionally see an improved TPR from 94.27% to 95.15%. Though the Top 2 Detector still boasts a TPR of 90.07%.

One limitation for the final luminance detector is that we now rely on the camera being set in a similar location to the testing data. To defend this approach, we refer to limitations inherent to the cascade object detector, namely distance and angle. As the cascade detector already contains inherent bias towards the angle that its training data is configured, we can also apply within that same limitation for iterative improvement on the algorthim.

6.5 Building a Depth Cascade Detector

Next, we explore using depth information to compute wrist location predictions.

To derive a 'best' depth detector we first observe three depth configurations. From these three configurations, we select the best result for further investigation. For the best result, we first focus on maintaining TPR. Secondarily we focus on the FP count as a criteria for detection quality. We finally focus on relative compute time between each configuration.

6.5.1 Limited Depth, Depth Gradient Magnitude, and Depth Gradient Direction

In Section 2.2.3 we describe our definition for image gradients. We apply gradient principles to depth image using magnitude, M and angle, ∇f. Our first detection method uses raw depth information from the camera, and is limited based on a maxi-

mum distance from the camera. Later, we describe this in further detail. For gradient magnitude and direction, we directly apply principles discussed in Section 2.2.3.

To provide some additional context of what applied gradient structure looks like in terms of depth information, we examine gradients for a subject in Figure 6.10. To allow for better scaling, we ignore depth values greater than 1500- i.e. we expect the subject to be within a range of 1.5 meters or approximately 5 feet. In Figure 6.10a and b, we present raw depth space. The 'bumps' present in the image are an effect of camera noise presented in background.

(a) Depth Magnitude (b) Depth Angle Pair

Figure 6.10: Images of visualized depth characteristics

To process the raw information, we limit the range to within five feet. We selected this range based on knowledge of camera configuration. This additionally helps remove sparse noise from behind the observer. We collected results of the three depth detectors into Table 6.5

Table 6.5: Primary results of depth detection bands

	Limited Depth	Grad Magnitude	Grad Direction
False Positive	18	20	29
True Positive	214	209	203
False Negative	13	18	24
Time/Image(s)	0.36	0.41	0.51

Of the three methods of deriving depth information, the limited raw depth provides the lowest FP/image count at 0.16, and TPR at 94.27%. This may be explained by the cascade object detector's use of oriented gradients. For the HOG setup on gradient information, a gradient is already performed, meaning that we are already performing a gradient operation by using HOG. Using magnitude, the direction components information of hands is been lost. Limited gradients will be selected for further examination.

6.5.2 Experimenting with Filtering Strategies

As for a quick test, before being quantized into an 8-bit image, first we process limited depth information using three filters of block size, two linear and one non-linear:

1. Linear average filter
2. Linear gaussian filter, sigma of 2
3. Non-linear median filter

We select a block size of five to ensure edges are not lost in filtering. For each filter, we train a cascade detector using all 1122 detected hands. In Table 6.6, we compile these results.

Table 6.6: Experiment results performing filtering on depth information

	Limited Depth	Gaussian	Median	Mean
False Positive	18	20	22	20
True Positive	214	212	209	211
False Negative	13	15	18	16
Time/Image(s)	0.36	0.43	0.36	0.36

For each configuration, the resulting TPR and FP/Image results are degrade. Despite the presence of noise in the depth images, the unprocessed data has objectively better results in discovering wrists. For this reason, within the cascade detection, we continue to examine unprocessed limited depth.

6.6 Comparing Detectors

In our final detectors we included the shifted hue space, which in Section 2.2.2 shows a high separability between skin regions and the background. The results for shifted hue, limited depth, and luminance detectors are compiled into Table 6.7.

Table 6.7: Experimental results of detectors trained across three bands

	Limited Depth	Hue Shifted	Lum
False Positive	18	26	16
True Positive	214	206	216
False Negative	13	21	11
Time/Image(s)	0.36	0.49	0.36

In hue shifted we see results objectively worse that in limited depth and luminance. This seems counter-intuitive given that hue shifted should have a higher degree of separability than luminance. Regardless, the hue shifted still maintains a high TPR

at over 90%. For this reason we continue to examine improving a detector based on all three detectors.

6.7 Combining Detectors

To improve singular results, we combine masks produced within the accumulative mask stage. We expect that if each band contains different strengths in discerning wrist locations in an image, then the combination of these band results should be better than the individual results.

To provide an example of how these results are combined together, we provide two reference images in Figure 6.11.

0	0	0	0	0	0	4	4
0	0	0	0	0	0	4	4
0	0	0	0	0	0	0	0
0	0	0	0	4	4	0	0
0	0	0	0	6	6	0	0
0	0	0	0	6	6	0	0
0	0	0	0	6	6	0	0
0	0	0	0	4	4	0	0
0	0	0	0	0	0	0	0

(a) Mask discerning blue elements

3	0	0	0	3	0	0	0
5	6	5	5	0	0	0	0
6	8	6	0	0	0	0	0
5	6	0	0	4	4	0	0
4	0	0	0	6	6	0	0
2	0	0	0	6	6	0	0
0	0	0	0	6	6	0	0
0	0	0	0	4	4	0	0
0	0	0	0	0	0	0	0

(b) Mask discerning square elements

Figure 6.11: Sample representation of two bands of accumulative masks

In our hypothetical example, these two reference images are used in detecting blue rectangles within an image. The detector, in Figure 6.11a, discerns blue elements with a high accuracy, where in Figure 6.11b, it discerns rectangular shapes. We roughly determine number based on number of adjacent tiles, including itself. In our implementation we simply sum the values within the accumulated mask. We provide a sample outcome of this accumulative mask sum in Figure 6.12.

3	0	0	0	3	0	4	4
5	6	5	5	0	0	4	4
6	8	6	0	0	0	0	0
5	6	0	0	8	8	0	0
4	0	0	0	12	12	0	0
2	0	0	0	12	12	0	0
0	0	0	0	12	12	0	0
0	0	0	0	8	8	0	0
0	0	0	0	0	0	0	0

Figure 6.12: Sample result of summation of two accumulative masks

From Figure 6.11 our final result provides improvement in the detection of blue rectangles. Provided we look for peak values within the image, the blue rectangle contains the highest possible peak. We apply this same behavior to our three wrist locating bands.

Table 6.8 provides the results when applying a summed accumulative mask on hue shifted, depth limited, and luminance.

Table 6.8: Experimental results of combined detectors

	Hue-Lum	**Hue-Depth**	**Lum-Depth**	**Hue-Lum-Depth**
False Positive	17	10	9	6
True Positive	215	222	223	226
False Negative	12	5	4	1
Time/Image(s)	.55	.54	.60	.80

Using combined hue-luminance-depth information we achieved a TPR of 99.56%. In this implementation we only detected 1 false negative within 226 true positives. The total time to classify a record image clocks in at 800ms, provided we run a single-threaded system.

6.8 Results

In our initial wrist location configuration, we present a method for determining wrist locations at a TPR of 94.27% at approximately130 ms per detection. Using this simple run result, we observe 416 false positives.

In our applied proposed system, we achieve a TPR of 99.56% at a count of 1 false-negative. While improving the location rate, we diminish false locating by a factor of nearly 100. An 800ms run-time is one cost of this format, though with a multi-threaded design we can improve this compute time to a potential of 490ms, the run-time of the longest cascade detector.

In depth-only tracking we present equal results of the simple configuration, at 94.27%. The run-time extends to approximately three times the length of the simple configuration, but to the benefit of 18 false-positives. In using depth-only information, we generate classifications invariant to obstructions on wrists, including tattoos and gloves. Given a depth-only classification we improve the scope for satisfying the objective to apply this detection in a work environment.

Based on these results, we derived some limitations in its use that we identified through the process.

6.9 Limitations

As the cascade detection is inherently biased towards the specific physical location of the image, so is our wrist locating system. Though this detracts from our objective on deploying this in a work environment, it is possible to use multiple cascade detectors across varying angles to add some robustness to the detection.

Chapter 7

CONCLUSION

In this thesis, we sought to work towards a means of preventing office workers from suffering from posture-induced MSDs. We set out to design components behind a system to detect bad posture, fulfilling the following objectives:

1. Easy to set up in a cubicle or work environment
2. Physically non-invasive so that the worker continue working free from distractions
3. Allow for some form of feedback on how to correct bad posture
4. Allow for incorporation of posture classification on multiple joints

To fulfill these objectives, we generated a data-set containing 1,186 depth images, 1136 color images, and 1,122 depth-mapped color images from 17 subjects totalling 48,569 frames of capture joint-tracking data. We then produced a rated data set consisting of 11,58 ratings across back, left wrist, and right wrist.

Using the classification and joint-tracking data we trained several models on back posture, including KNN, Neural Network, Random Forest, and SVM, each with four separate data input types and three controls. In total, we trained 242 distinct configurations of machine learning classifiers. Between these processes we achieved at the highest a 97% sensitivity, a 70% specificity, and a 0.5 kappa score. Our highest scoring classifiers were a bootstrap control SVM with a raw data input type, and a bootstrap neural network of a median filter of span 10.

Working towards wrist posture location, we leveraged the color, depth, and depth-mapped images from our data set. Using our accumulative mask approach across hue,

luminance, and depth, we presented a combined wrist detector result allowing for a 99.56% true positive location rate, with 6 false positives across 116 images. Using the depth-only approach, allowing for a wider range of environments, we designed a wrist detector with a 94.27% true positive rate with 18 false positives.

In conclusion, we studied two components behind a system that can automatically detect bad posture and methods for wrist detection given an occluded environment. Such a system, if built, has the potential to reduce repetitive strain injuries and keep office workers happy and productive.

Chapter 8

FUTURE WORK

In this section we outline some areas of work following up the research presented here. To highlight different facets of the future work, we divide it into multiple topics, we first describe direct enhancements to the work presented within our contributions, then go into general future work on the topic of posture biofeedback.

8.1 Dataset

To further improve wrist detection within the data-set, we can additionally capture images from multiple angles against subjects. Given a specific angle that the subject is with respect to the camera, we can potentially choose the appropriate cascade detection method.

To improve the effectiveness of this posture tool in a work environment, we can record two subjects simultaneously in an image. As described, the Kinect tracks up to two separate joint structures, allowing for separate workers to be evaluated at the same time.

On improving the overall effectiveness of back classification, more crowd-sourced classification can be performed on the pre-existing data set. In order to provide a more even spread of classification results, we can employ an expert trained in the field to primarily classify posture results. We can additionally leverage the 'expert' data to judge the classification ability of a crowd-sourced voter.

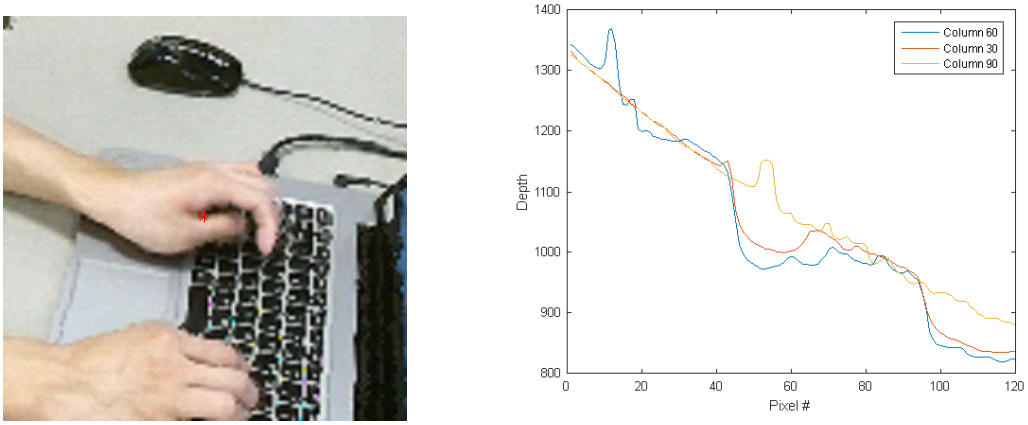
8.2 Back Classification

One potential step to improve back classification, would be to test alternative methods provided by the caret package, approximately 182 variants. To improve the results of our specific classifiers, we can perform additional tuning on the machine learning parameters.

To combine these classifiers, in [10] Deane-Mayer et al. released a package build on top of caret, caretEnsemble. The package allows for multiple fittings of different training models in both linear and non-linear combinations. It is possible that using the same classifiers we present in a combination may provide better results than the separate implementations.

8.3 Wrist Classification

Within this thesis we presented a method to perform wrist locating. Based on the data we present, it should be possible to use these points and grow a region containing hand information. Both color and depth information can be leveraged: using color information we can extract wrists and arms. Using depth information we can perform a threshold to separate left and right arms. We can also use depth information in separating arms from the desk. In Figure 8.1, we present some preliminary results of performing a vertical depth slice of hand regions in a test region.



(a) Image of centered hand

(b) Depth of vertical wrist slicing

Figure 8.1: Sample for growing wrist region

We performed three vertical slices of three different horizontal locations of a hand centered within a test region. In the center we observe non-linear changes, indicating that there could be an object in that location.

Alternatively, as a next step in wrist classification we could directly extract the detected regions as hand locations. To generate this we simply grow the region by x number of pixels, where x is determined based on the distance of the hand from the detector. From this extracted region we create a feature space based on cascade detection features, for example, haar-like features.

8.4 Objectives

As noted in our objectives, two direct continuations of this work would be to actually perform wrist classification on data, and to allow for multi-subject evaluation.

To further our fourth objective and the topic of this Thesis, allowing for biofeedback in the Kinect, the current back classifier can be directly plugged into some biofeedback tool. As we have a high certainty that we expect a low rate of false-

negatives, any data that is returned from the classifier as predicted bad posture is likely, actually bad back posture. Knowing this, having a feedback tool that reacts to bad posture detection is ideal. In a simple implementation we present a red tint directly to the subject's monitor whenever the classifier predicts bad posture. In seeing the red light, the subject knows that they must correct their sitting position. Presenting a red lights allows for our objective in providing non-invasive posture evaluation tools.

BIBLIOGRAPHY

[1] A. O. Association. Computer vision syndrome. Online, 2014. http://www.aoa.org/patients-and-public/caring-for-your-vision/protecting-your-vision/computer-vision-syndrome?sso=y#1.

[2] A. P. Association. Multitasking: Switching costs. Online, mar 2006. http://www.apa.org/research/action/multitask.aspx.

[3] M. Azimi. Skeletal joint smoothing white paper. Online, 2005. https://msdn.microsoft.com/en-us/library/jj131429.aspx.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[5] J. Brownlee. Feature selection with the caret r package, Sep 2016.

[6] Human subjects – guidelines for research protocol. Online, jan 2017. http://research.calpoly.edu/HS-guidelines.

[7] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun. Joint cascade face detection and alignment. In *European Conference on Computer Vision*, pages 109–122. Springer, 2014.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[9] J. A. Diego-Mas and J. Alcaide-Marzal. Using kinect^{TM} sensor in observational methods for assessing postures at work. *Applied Ergonomics*, 45(4):976 – 985, 2014.

[10] Z. A. D.-M. et al. Caretensemble: Ensembles of caret models. Online, feb 2016. https://cran.r-project.org/web/packages/caretEnsemble/index.html.

[11] M. K. C. from Jed Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, and T. Hunt. *caret: Classification and Regression Training*, 2016. R package version 6.0-73.

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

[13] T. J. Hopkin and S. Sarkar. Sedentary behavior of white collar office workers - review. Online, apr 2016. https://www.ecronicon.com/ecnu/pdf/ECNU-03-0000102.pdf.

[14] N. is an agency of the U.S. Department of Commerce. Ewma control charts. Online, oct 2013. http://www.itl.nist.gov/div898/handbook/pmc/section3/pmc324.htm.

[15] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004.

[16] M. Kuhn. The caret package. Online, nov 2016. https://topepo.github.io/caret/index.html.

[17] A. G. Lalkhen and A. McCluskey. Clinical tests: sensitivity and specificity. *Continuing Education in Anaesthesia Critical Care & Pain*, 8(6):221, 2008.

[18] C. W. LeBlanc K. Carpal tunnel syndrome. *American Family Physician*, 83(8):952–958, apr 2011.

[19] S. Z. Li and Z. Zhang. Floatboost learning and statistical face detection. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 26(9):1112–1123, 2004.

[20] A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[21] M. L. McHugh. Interrater reliability: the kappa statistic. *Biochem Med (Zagreb)*, 22(3):276–282, Oct 2012. 23092060[pmid].

[22] W. McKinney. pandas: a foundational python library for data analysis and statistics. *PyHPC 2011: Python for High Performance and Scientific Computing*, 2011. http://www.scribd.com/doc/71048089/pandas-a-Foundational-Python-Library-for-Data-Analysis-and-Statistics.

[23] Microsoft. Kinect for windows features. Online, 2015. http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx.

[24] Microsoft. Kinect for windows sdk. Online, feb 2015. https://msdn.microsoft.com/en-us/library/dn799271.aspx.

[25] A. A. OSHA. What is osha? Online, 2015. http://www.allaboutosha.com/what-is-osha.

[26] Pandas documentation. Online, oct 2015. http://pandas.pydata.org/pandas-docs/version/0.17.0/index.html#.

[27] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[28] O. Safety and H. Organization. Good working positions. Online, may 2010. https://www.osha.gov/SLTC/etools/computerworkstations/positions.html.

[29] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[30] B. Taylor, M. Birk, R. Mandryk, and Z. Ivkovic. Posture training with real-time visual feedback. In *CHI 2013 Interactivity*, pages 3135–3138, Paris, France, 2013.

[31] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.

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

[33] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[34] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg. Fast asymmetric learning for cascade face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):369–382, 2008.

[35] C. Yan, F. Coenen, and B. Zhang. Driving posture recognition by convolutional neural networks. *IET Computer Vision*, 2015.

APPENDICES

Appendix A

RESEARCH PROTOCOL

Research Protocol

Title of the research: Workplace Posture Assessment and Biofeedback with Kinect

Primary Investigator: Matt Crussell, graduate student, Department of Electrical Engineering

Faculty Advisor: Dr. Jane Zhang, Department of Electrical Engineering

Statement of purpose, benefits, and hypothesis:

The purpose of this research is to work towards a biofeedback tool in assessing a person's posture while sitting at a desk. Musculoskeletal Disorders (MSDs) such as Carpal Tunnel can be influenced by poor habits in front of a computer. Biofeedback for those who begin regressing to these bad habits could be beneficial at reducing the occurrence of MSDs.

An important step in this biofeedback process is to train a classification system on what is 'good' and 'bad' posture. Gathering training data will allow us to discriminate between different sitting postures. We hypothesize that a classification system will be effective in providing feedback for sitting positions.

Methods:

Subjects: The subjects will predominantly be Cal Poly students between the ages of 18-50. Those who already suffer from an MSD will be excluded from this study.

Experimenter(s): Tests will be administered by the primary investigator, Matt Crussell.

Materials and Procedures: Each volunteer will be selected to perform various tasks on a Computer as per location. In the case a volunteer is uncomfortable with being recorded in a public setting, the volunteer will be scheduled for a time to be recorded privately. Each volunteer will be asked to view content and respond to it (e.g. read an article and respond to it via word document). The documents for the test will not be saved.

We intend to collect approximately 100 data points of both 'good' and 'bad' posture for classification. This will equate to at least 30 subjects. In the case a subject primarily exhibits a singular type of posture, the protocol may be adjusted. To exhibit a different posture, we will select one of the following set of adjustments

1. Change the reading material to a document citing good ergonomics
2. Adjust the seat height either up or down
3. Adjust the keyboard, mouse positions from the subject's preferred location

Analysis:
To determine what a 'good' and 'bad' posture is, we will employ a crowd source solution. To do this, a set of images with blurred faces will be collected for each subject.
These images will be placed on a website where trained viewers can select and classify postures based on a given criteria. Viewers will be trained to discriminate between

postures and then tested with some basic images with known posture classifications. Those viewers who successfully pass the test will be provided with additional images and asked to perform posture classification.

The classification consensus set will next be condensed into a single posture determination based on mean and standard deviation of posture data.

Appendix B

INFORMED CONSENT FORM

Informed Consent Form

INFORMED CONSENT TO PARTICIPATE IN A RESEARCH PROJECT, "Workplace Posture Assessment and Biofeedback with Kinect"

A research project on computer vision is being conducted by graduate student Matt Crussell and Dr. Jane Zhang in the Department of Electrical Engineering at Cal Poly, San Luis Obispo. The purpose of the study is to develop a computer program that can classify and provide feedback on a person's posture when sitting in front of a computer.

You are being asked to take part in this study by performing 3-5 simple tasks on a computer in front of a Kinect v2 sensor. As you work, your image will be recorded. Your participation will take approximately 5-10 minutes. Please be aware that you are not required to participate in this research and you may discontinue your participation at any time without penalty.

The possible risks associated with participation in this study are minimal risks associated with performing various activities on a computer. There is a minor risk of pain, discomfort, injury and also a minor risk of embarrassment from being recorded in front of a camera. If you should experience any of the aforementioned ailments, please be aware that you may contact Cal Poly Health and Counseling Services at (805) 756-1211, located in Building 27, for assistance.

Your confidentiality will be protected by storing all data on a password protected computer. Prior to image analysis, your facial features will be blurred to prevent identification. Your name will not be used in any reports of this research without your permission. Any and all video or images released will have faces blocked out. Potential benefits associated with the study include contribution to the development of a posture biofeedback system.

If you have questions regarding this study or would like to be informed of the results when the study is completed, please feel free to contact Matt Crussell at (925) 360-6861 or Dr. Jane Zhang at (805) 756-7528 . If you have concerns regarding the manner in which the study is conducted, you may contact Dr. Steve Davis, Chair of the Cal Poly Human Subjects Committee, at (805) 756-2754, sdavis@calpoly.edu, or Dr. Dean Wendt, Dean of Research, at (805) 756-1508, dwendt@calpoly.edu.

If you agree to voluntarily participate in this research project as described, please indicate your agreement by signing below. Please keep one copy of this form for your reference, and thank you for your participation in this research.

_____ _____

Signature of Volunteer Date

_____ _____

Signature of Researcher Date