2014

3D Image Acquisition System for Facial Recognition

James E. Pearson
Eastern Washington University

3D Image Acquisition System for Facial Recognition

A Thesis

Presented To

Eastern Washington University

Cheney, Washington

In Partial Fulfillment of the Requirements

for the Degree

Master of Computer Science

By

James Pearson

Winter 2014

THESIS OF JAMES PEARSON APPROVED BY

Carol Taylor, PHD, GRADUATE STUDY COMMITTEE	DATE
Stu Steiner, ABD, GRADUATE STUDY COMMITTEE	DATE

MASTER'S THESIS

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Eastern Washington University, I agree that the JFK Library shall make copies freely available for inspection. I further agree that copying of this project in whole or in part is allowable only for scholarly purposes. It is understood, however, that any copying or publication of this thesis for commercial purposes, or for financial gain, shall not be allowed without my written permission.

Signature_____

Date_____

Abstract

Identification of an individual has been a long standing problem in human existence. It is clear from the rising rates of identity theft and misappropriation of services that current methods employed to identify individuals is ineffectual. The most common way people identify other individuals is via recognition of the subjects face. The speed and accuracy with which one person can identify another person is uncanny leading to the development automated facial recognition systems. The vast majority of these systems use two-dimensional (2D) images imagery to perform facial recognition with limited success. The use of 2D imagery has limitations which cannot be overcome therefore three-dimensional (3D) facial recognition systems have become the focus of study. The imaging systems used to capture 3D facial imagery is complex, expensive and requires a degree of cooperation from the subject. This paper proposes the use of older photogrammetry techniques to capture 3D facial imagery thereby eliminating the need for specialized 3D imaging equipment. In addition the camera system, once configured, does not require an operator nor does it require undo cooperation of the subject.

Acknowledgements

The completion of this work would not have been possible without the encouragement and help provided by Dr. Carol Taylor, Stu Steiner and Brian Kamp. Even though outside her area of expertise Dr. Taylor easily saw the impact this work will have on computerized security systems. The advice, assistance and direction Dr. Taylor offered was also invaluable for obtaining materials from outside institutions which were necessary for the project's completion. Dr. Taylor and Stu Steiner provided invaluable assistance during the final reviews of this paper improving both its content and quality. In addition Stu Steiner helped with the preparation of the presentation through review and providing audiences to critique the presentation. Brain Kamp assisted by permitting me access to the animation studio and the lighting systems available there.

Table of Contents

Table of Figures

List of Tables

List of Equations

1. Introduction

Proving an individual's identity has been an ongoing problem for centuries and continues to be a difficult task today. In modern western societies identity creation is typically started at birth through careful documentation, biometric data gathering and tagging with wristbands. The initial identification is added to over the years through additional documentation, imagery and further biometric data collection. Even with multiple means to identify an individual it can still be a challenging task to prove an individual is who they claim to be. Documentation can be destroyed, altered or falsely generated. Imagery can be manipulated and comparisons of such may be accurate, ineffectual or simply wrong. Biometric comparisons can and do suffer from the same problems as imagery and require an additional level of cooperation by the subject and some degree of training for the person obtaining the biometric. In addition the more definitive the biometric the longer it takes to process the data. Even with modern techniques, proving the identity of an individual is problematic.

Proving the identity of an individual today is most commonly photographic and descriptive. The individual's identity typically revolves around a multitude of tokens. In the United States the form of token most often used for identification is a driver's license which has both a picture of the individual and general physical description. Internationally the passport contains similar information. Another common token is the bank or credit card which may or may not have a picture of the authorized user and rarely has any other identifying information. Myriads of other tokens (ID cards, insurance cards, key cards, etc.) also exist with little if any descriptive information about the individual authorized to use it. The Internet and ecommerce use textual tokens, most commonly in the form of a username and password, to gain access to personal, financial, legal and medical information. The purchase of goods and services across the internet also uses the ubiquitous username and password system to gain access to personal and financial

information, purchase histories and more. Automated token based systems pervasive in today's society have no ability to determine if the individual using the token is in fact the authorized user. Automated token based systems only confirm the user of the token knows the secret to gain access regardless of the user's identity. The failure of token based systems confirming or proving the identity of individuals is repeatedly confirmed by rising rates in identity theft and fraudulent use of goods and services.

Automated facial recognition could replace the current token based systems with a new token – the human face. The concept behind automated facial recognition, a type of biometric, comes from the human ability to recognize individuals quickly simply by seeing the individual either in person or via an image of the face. The ease, rapidity and accuracy with which people can identify individuals is amazing however people at times misidentify individuals for many reasons including environment, health and even failures in cognition. Still, the human ability to naturally, quickly and most often accurately identify individuals has been the impetus to develop automated systems to identify individuals. Beginning in the early 1980's researchers began to develop computer algorithms to identify individuals through analysis of pictures. Through more advanced image processing techniques these early systems have been vastly improved to a degree where there are now commercially viable image based identification systems.

Significant advances have been made in the field of facial recognition. Facial recognition using photographic imagery (2D) is not 100% accurate and may be at its limits of capability. Many studies have progressed to using three dimensional (3D) facial images obtained from various sources as a means to overcome the limitations of 2D systems while others have combined 2D systems with 3D systems. The initial research has been promising as 3D facial recognition systems have proven to be more accurate and robust than existing 2D systems. However imaging systems capable of capturing 3D imagery are expensive and require the

cooperation of the subject during image capture. A simple and inexpensive system to capture 3D images for use with facial recognition systems must be created.

Today the techniques, hardware and software exist to create an inexpensive 3D image capture system. To create an inexpensive 3D imaging system three things are needed: inexpensive and computer compatible cameras, a method to convert 2D images into 3D images and software to manipulate the images. Inexpensive cameras compatible with computer systems exist which create electronic images suitable for computer analysis and manipulation. Photogrammetry has long been used to capture pairs of images that when viewed with special lenses simulate a 3D image. Software tools have been developed to edit and manipulate images in just about any way imaginable including converting 2D images into 3D images.

The primary focus of this master project is to explore the use of inexpensive cameras and photogrammetry techniques to create 3D images for use in facial recognition systems. Specifically, the research establishes a process showing how to use image processing software to calculate individual camera properties, use those properties to compute the correspondence between a pair of cameras and then use the correspondence information to create a 3D image suitable for use in facial recognition programs. Although other research has been conducted in this area no other research has attempted the process using inexpensive and highly available off the shelf equipment as a basis of the 3D image capture system. Furthermore, and unlike currently available 3D image capture systems, the image capture system should not require an operator or more than casual cooperation of the subject and therefore extends the image capture capabilities of 3D facial recognition systems in general.

2. Background

Background information on identity creation and validation, the ability of people to identify specific individuals, biometrics and anthropometry are presented here to enhance understanding of the fundamental concepts and terminology presented in this paper.

2.1. Identity Validation

Using documentation to validate the identity of a claimant requires the information contained on the documents is verifiable, i.e. direct comparison of the height, weight, age, picture or other identifying information to the claimant must be performed. However, correct identification using documentation is still fallible for reasons listed above and by human error in the identification process or even guile by the claimant. Using imagery for identification is also problematic because imagery can be manipulated and comparisons of imagery, either by individuals or automated means, may be accurate, ineffectual or simply wrong. Likewise biometric identification requires comparisons similar to documentation and imagery but also requires skilled operators and technicians to obtain and process the biometric data and also requires an additional level of cooperation from the claimant seeking identification. For biometrics, the more definitive the biometric, the longer it takes to process the data to either confirm or deny the validity of an individual. All typical methods to validate the identification of a claimant, documentation, imagery and biometric, have their flaws and problems.

Exacerbating the validation of identity for claimants is the fact virtually none of the electronic systems in general use today are capable of using documentation, imagery or biometrics to validate the identity of a claimant. Rather, most use some form of a token; the driver's license, identification, bank or insurance cards are examples of tokens but other forms exist. The use of a token alone is not enough to identify an individual because any claimant, authorized or unauthorized, possessing the token may gain access to the systems to which the

token is connected. Whether or not a token is used, the vast majority of electronic systems use a challenge and response or matched pair system to validate the identity of a claimant. Challenge response systems involve the claimant being presented with a question to which a response is given. A match between the response provided and the response expected grants access to the claimant. Matched pair systems are almost universally username and password systems where the claimant provides both username and password and the system confirms the username exists and matches the password provided with the one stored for the username. The principle problem with challenge response or matched pair identity validation systems is any claimant knowing the correct response or matched pair is permitted access.

2.2. Human Failure to Recognize Faces

Human ability to recognize familiar faces is uncanny in its speed and accuracy. It is also highly fallible when the faces presented are unfamiliar, distorted or otherwise not clearly viewable by the observer. Wilkinson reports ten percent of the December 2004 tsunami and fifty percent of the Bali October 2002 bombing were misidentified citing decomposition and physical alterations of the victim faces [1]. In judicial matters eyewitness testimony identifying individuals is often wrong. Kleinberg states: "…eyewitness identification evidence is among the least reliable forms of evidence…" [2]. Research by Davis reports identification of individuals with whom the subject is familiar is highly reliable but other identifications are not [3]. Kleinberg states research conducted by Kemp, Towell and Pike showed cashiers had a difficult time matching photographs on credit cards with the customers [2]. Gupta states that for 84% of all DNA exonerations reported the wrongful convictions were due to false recognition by witnesses [4]. The psychology of why people often misidentify an individual are many and include things like distinctive morphology of the face, whether the face is attractive, similar in race, time to observe, clarity due to lighting and many other factors. The reliability of people

correctly identifying individuals, especially unfamiliar individuals, is low having a dramatic and costly impact on all aspects of a society. People's poor rate of identifying unfamiliar individuals is a major impetus for developing automated facial recognition systems which promise to better reliability.

2.3. Biometrics

Biometrics from the Ancient Greek bios, "one's life," and metron, "a measure," is the science of identifying individuals from common yet distinct physical attributes [5] [6]. Many forms of biometrics exist some of which are the fingerprint, iris, palm print, hand geometry, vascular patterns, gait, facial recognition and DNA [7]. The main goal of biometrics is the identification of specific individuals. The first uses of biometrics were private concerns regarding the identification of individuals in personal or business transactions. In the mid to late 1800's scientific investigation of biometrics ensued resulting in the scientific fields of dactlyoscopy (fingerprints) and anthropometry. Forensic identification became the primary impetus for biometric study of both avenues of measurement from about 1880 onward. Though most early forms of biometrics have fallen out of favor the two still in widespread use today are photographs and fingerprints. In the late 1900's DNA identification became the most discriminatory biometric measurements and is a dominant form of identification for legal systems and is used in conjunction with photographs and fingerprints.

2.4. Anthropometry

Anthropometry is the science that defines physical measures of a person's size, form, and functional capacities [8]. A French police officer, Alphonse Bertillon, is arguably the inventor of modern anthropometry. His system, Bertillonage, revolutionized the identification of individuals throughout the world [2]. The methods and reasons Bertillonage became the most popular means to identify individuals is clearly stated by Moenssens:

> "Bertillon's system of anthropometrical measurements was based on three ideas: the fixed condition of the bone system from the age of twenty until death; the extreme diversity of dimensions present in the skeleton of one individual compared to those in another; the ease and relative precision with which certain dimensions of the bone structure of a living person can be measured using simply constructed calipers." [9]

Since Bertillion's time anthropometry has fallen out of favor as a means of identification having been replaced with more reliable methods such as dactlyoscopy and more recently DNA analysis. Although dactlyoscopy and DNA analysis are both significantly better than previous facial methods of anthropometric analysis neither is foolproof. Both dactlyoscopy and DNA analysis require professional analysis wherein resulting analysis and identification of the individual is related as a probability of an exact match based on the professional's opinion. Neither dactlyoscopy nor DNA analysis is able to positively and without error identify a specific individual.

Because of the ever increasing need to correctly and positively identify individuals coupled with the increasing rates of identity theft and the ease with which token based identification systems are compromised, facial anthropometry is again a focus of research. The main failing of Bertillonage was not the system Bertillion created but the inaccuracies in the measurements taken by different individuals and changes to the Bertillonage system introduced by those same individuals. Today, with the advent of automated processing, the errors introduced by the users of Bertillonage can be avoided; accurate and repeatable measurements can be made. In addition, using automated systems to capture images, facial anthropometry is non-intrusive requiring little or no effort by the subject unlike the high level of cooperation needed from subjects to acquire fingerprints or DNA samples [2].

3. Reliability of Biometrics

Many researchers have studied the reliability of biometric measurements. Some studies determine the use of biometric measurements are not suitable for identification while others indicate that when proper and careful measurements are taken biometric measurements are a definitive means by which individuals may be identified. What follows are examinations of both types of studies with some analysis of problems found in the studies and their contributions.

3.1. Studies Showing Unreliability of Biometric Measurements

In a study by Kleinberg it was found that the use of anthropometric measurement alone is not sufficient for identifying specific individuals [10]. The study used video footage from surveillance cameras and still photographs taken with a high resolution camera. Facial measurements were made using the images from each camera system. The resulting measurements from the video footage were compared to the measurements from high resolution images. A minimal set of points was selected for measurement: ectocanthion (bilaterally), stomian and nasion. Between these points distances and angles were measured then proportions calculated from the measurements. Use of proportions compensates for slight differences in size between images. The results of the study concluded anthropometric measurements were inconclusive and unable to identify an individual. During the experiment two photographs were found to have three identical proportions and three identical angles. Later morphological results indicated the two pictures had different eye and eyebrow shapes in addition to having different mouth and nose sizes.

The study is flawed in two ways. First far too few facial points were selected and second the program used to select points required operators to place points on the photographs and video images. With such a limited number of points from which to calculate angles and proportions, the likelihood of having enough information to differentiate two individuals is unlikely.

Additionally the selection of stomian is a poor choice because it varies in position depending on if the jaw is clenched or relaxed and with facial expression. Operator placement of points introduces inter and intra operator variation of point placement. This variation skews the resulting measurements. Invalidation of results occurs when taking anthropometric measurements when inter and intra operator error of this type is allowed.

Another study by Kramer researched the width and height ratio of faces to determine its validity for determination of sex. Kramer used 2D photographs, 3D scans and direct anthropometric measurements with the latter two compared against the 2D photograph [11]. Results showed the width and height ratio not to be sexually dimorphic in nature and not reliable for sex determination. In fact the difference in ratios between male and females was statistically insignificant. Like Kleinberg, this study is also flawed by the choice of landmarks. The vertical distance measurement the points selected were the highest point of the upper lip and the highest point of the eyelids (averaged bilaterally) both of which are highly variable to facial expression. Horizontally left and right zygions were used and are also influenced by facial expression though much less so than the vertical measurement chosen. This study again illustrates the failure of anthropometry's usefulness in identification when a limited number of measurements are used and when points are influenced by facial expression.

Another study by Davis used more facial points in the similarity analysis of faces. In this study 38 facial landmarks and 14 profile landmarks were used to calculate Euclidean distances and angular measurements; no proportions were used [12]. The results of the study were similar to those of Kleinberg in that there is a great amount of similarity between anthropometric facial measurements between different subjects. In conclusion, Davis stated "…need for caution, when attempting to 'prove' an identity match beyond reasonable doubt."

The Davis study also has flaws. Facial landmarks selected for use were arbitrary. Arbitrary selection of landmarks includes measurements with high and low statistical deviations. The inclusion of measurements with low statistical deviation does little to differentiate individuals while also skewing matching results by returning false positive matches. Eliminating landmarks with low statistical significance and the corresponding measurements will produce more reliable results while using the same matching algorithms.

3.2. Studies Showing Reliability of Biometric Measurements

The reliability of direct anthropometric measurement is possible with training and practice. The World Health Organization reported on the reliability of anthropometric measurements taken in the study of child growth [3]. The key to reliable measurements is practice and training of individuals taking biometric measurements. During this study measurements initially taken were inaccurate and unreliable. Investigation showed persons taking measurements lacked training and experience. After these individuals received training and had the repeated opportunity to apply what they learned, the anthropometric measurements became both accurate and reliable allowing the growth study to be completed. The World Health Organization shows it is possible for individuals to obtain accurate and reliable direct anthropometric measurements with training and practice. Likewise, computer algorithms accurately and repeatedly determine the location of facial points.

Fourie performed a study using cadaver heads to evaluate the accuracy of anthropometric measurements [13]. The study was an evaluation and direct comparison of three image acquisition methods: laser surface scanning, cone beam computed tomography and 3D stereo-photogrammetry. To ensure measurements from each of the camera systems used the same physical location spheres were mounted to all facial landmarks used for anthropometric measurements. Physical linear measurements were also taken for each computer-measured

distance for comparison and accuracy evaluation. Fourie found all three 3D methods of anthropometric measurement to be highly accurate and reliable with an overall absolute error of less than one millimeter. He concluded: “By analyzing human remains via 3D models, forensic anthropologists can construct biological profiles using precise and accurate metrical data to determine key aspects of identity.”

This type of anthropometric analysis was used in the work of Zhuang to build facial profiles. In the study, the categorization of anthropometric measurements by gender, age and ethnicity is used for the evaluation of face mask fitment [14]. The conclusions drawn were that a selection of face masks should be made available to workers based on gender as the primary sizing criteria followed by ethnicity. What makes this work significant is the use of anthropometric measurements to differentiate between genders and ethnicity. Using anthropometric measurements and correlations could be made between the measurements and gender in addition to ethnicity.

4. Types of Facial Biometric Problems

Many people confuse or use interchangeably the terms detection, identification and recognition when discussing different aspects of facial biometrics. While similar, each poses a unique problem for automated computer processing systems.

4.1. Face Detection

The lowest level is the detection of faces in imagery or videos [7]. A simple task for humans of virtually all ages is to find a face in an image. However, a computer must use pattern-matching algorithms to detect faces. Typically a machine learning algorithm is used to train the system to locate and detect the patterns in the image which are then used to determine if a face or multiple faces exist. There are a myriad of packages available, some at cost and others free,

which can perform face detection. There are also several open source detection packages available.

4.2. Face Identification

Sometimes identification is called verification and the terms can be used interchangeably in this context. Identification is a one to one matching problem. A system is presented with an unknown claimant and known identity and asked to identify the claimant as being the identity [4]. The system may or may not use face detection to locate the subject in the provided image prior to matching depending on the algorithms used for face matching. There may exist development packages or programs only performing identification. However, this author did not find any during the search.

4.3. Face Recognition

Facial recognition is a one to many matching problem [15]. Presented with an unknown face, a claimant, the system's task is to find a matching identity from a data store of identities. This process is an extension of identification. There exist numerous development packages and software is available to perform facial recognition. The vast majority of facial recognition packages use 2D methods for recognition. Most packages found are used for commercial software and therefore require purchase to use although some have limited time evaluation periods.

5. Automated Facial Recognition

What follows is a brief description of various techniques and methods employed in automated facial recognition. Most of the techniques presented stem from lessons learned in signal processing and later applied to image processing. Automated facial recognition is accomplished using either standard 2D images or 3D range images. Two-dimensional facial recognition is a long-standing area of study in many scientific areas but has received intense

attention from computer science research for more than thirty years. Many methods and algorithms have been proposed to perform facial recognition from 2D imagery. Sources for 2D images vary from mug shot style photographs to frame capture and analysis of video images. More recently 3D facial recognition has been at the forefront of facial recognition research. The reasoning for this is that 2D imagery does not have enough discriminatory information to positively identify an individual. Three-dimensional images capture the same information as 2D images plus an additional distance vector thereby providing more discriminatory information than 2D images. It has been show 3D facial recognition systems are able to outperform 2D facial recognition systems. Information about 2D automated recognition methods is presented here because most 3D methods utilize techniques based on 2D algorithms.

5.1. 2D Facial Recognition Methods

Many methods have been applied to the problem of recognizing faces in 2D images. Generally, these methods can be broken down into feature based or holistic measurement techniques [16]. Both techniques perform about equally when compared to each other. Studies combining feature based and holistic measurement techniques do not lead to significant improvement in facial recognition results so typically they are not implemented at the same time.

5.1.1. Feature Based Methods

Feature based methods use locations or features of the face such as the eyes, mouth, nose, ears, etc. to define a set of points in 2D space. The points are used to calculate the distances and or ratios between points and are stored as a vector. The vector is statistically compared with other vectors stored in the system using standard pattern matching techniques. An advantage of this method is the compactness of the data and the point detection process is separate from the facial matching process. A similar feature based method is the elastic bunch

graph wherein the points are used to create a fully connected graph which has been shown to be more accurate though much more computationally expensive. The selection of points to use for feature based facial recognition is generally chosen at random with little thought to their statistical contribution or significance.

5.1.2. Holistic Methods

In holistic analysis the entire image including the face is used as input and re-projected into smaller spaces for classification. The two main holistic methods are Eigen Faces and Fisher Faces. Eigen Faces uses principal component analysis projection for comparisons and work best when single images of each individual are stored for comparison. Linear discriminate analysis is used for Fisher Faces and works best when there are multiple images available for each individual. Essentially both methods scan the 2D image to create a re-projection of its data into a smaller coordinate system. The re-projected data is compared with similar data stored for matching purposes. Advantages of both Eigen Faces and Fisher Faces are their ability to reconstruct the original image from a limited number of re-projections and the compactness of the data stored for comparison.

5.1.3. Limitations on 2D Methods

Systems relying on feature-based methods are frequently compromised by the environment, physical positioning and size of the facial image. Other feature based disadvantages are the similarity between individual faces makes specific identification of an individual difficult and the accurate location of facial points. Holistic methods are able to overcome some of the limitations of feature-based systems but the tradeoff is they are highly susceptible to lighting differences and background noise. Both systems can be fooled using static images because they rely on simple 2D imagery for data input. Current facial recognition

systems relying on solely on 2D imagery begin to fail by returning an increasing number of false positives as the numbers of individuals in a system rise.

5.2. 3D Facial Recognition Methods

The methods used for analysis of 3D images are similar to 2D feature based methods [4]. What is different is an image depth or range map is used either alone or in conjunction with a 2D image. The depth map or range image is a 3D image containing similar information as a 2D image but also includes a distance vector. The distance vector measures the distance from the imaging system for each pixel in the image. The advantage of locating facial landmarks using 3D images is it provides similar information as a 2D image, the *x* and *y* coordinate, plus the distance vector, a *z* coordinate, to locate the facial point on three planes, *x*, *y* and *z*. A 3D image also allows rotation of the 3D and the image about the axis of each plane without distortion to overcome positioning issues hampering 2D feature based methods. A 3D image can also be scaled accurately in all three planes to resize the 3D images to a standard size for image processing. Normal 2D images taken with the same camera from the same perspective can also be repositioned and resized using the 3D repositioning and resizing calculations. The ability to rotate the both the 2D and 3D image on three axis permits the image map to be repositioned into a canonical position for image analysis [4]. Another feature of 3D imagery is its invariance to lighting conditions and background noise that are concerns of 2D holistic methods that tightly control these attributes in order to improve image quality.

6. Facial Landmark Selection

The selection of landmarks for use in anthropometric work should distinguish between individuals. Specifically the range and standard deviation of measurements, be they Euclidean distances, geodesic distances or proportions, the landmark selection for measurements should yield be the most diverse of measurements in order to better discern between distinct

S. No	Anthropometric Proportion	Σ	S. No	Anthropometric Proportion	Σ
1.	O3 = (ex − en, l)/(en −en)	7.75	13.	N30 = (mf −mf)/(en −en)	6.06
2.	O10 = (en −en)/(al − al)	8.29	14.	N31 = (ex – m'_{sag}, l)/(en −en)	7.01
3.	O12 = (en −en)(ch −ch)	6.02	15.	N32 = (al −al)/ch –ch	5.04
4.	F32 = (n −sto1)/(ex −ex)	5.30	16.	N33 = (sn −prn)/(sn− sto1)	13.8
5.	N1 = (al −al)/(n −sn)	5.81	17.	L1 = (sn −sto1)/(ch −ch)	5.40
6.	N2 = (mf − mf)/(al −al)	7.08	18.	L4 = (sn −ls)/(sbal – ls', l)	10.2
7.	N4 = (sbal −sn, l + r)/(al −al)	8.80	19.	L5 = (sn −ls)/(sn −sto1)	5.97
8.	N6 = (ex –m'_{sag}, l)/(mf −mf	14.6	20.	L6 = (ls − sto1)/(sn −sto1)	7.10
9.	N7 = (sn− prn)/(al − al)	6.28	21.	L7 = (ls − sto1)/(sn −ls)	13.3
10.	N8 = (sn− prn)/(sbal −sn, l + r)	12.8	22.	L9 = (ls − sto1)/(sto2 −li)	16.9
11.	N15 = (en –m'_{sag}, l)/(sn− prn)	11.2	23.	L14 = (sn−sto1)/(n −sn)	5.10
12.	N16 = (en –m'_{sag}, l)/(en − m,l)	7.26			

Table 6-1 "The 23 most variable anthropometric facial proportions for adult humans along with their standard deviation values (Farkas 1987). The corresponding fiducial points are presented in Figure 6-1. N denotes nasal proportions, O denotes orbital proportions, L denotes proportions related to the mouth region, and F denotes facial proportions" Gupta [17].

individuals. Gupta, using a list created by Farkas and Munro, selected 23 anthropometric proportions with the highest standard deviations (Table 6-1) resulting in a list of 25 facial landmarks [17] [18].

The 25 landmarks still represented a large search space of 300 measurements. With further analysis Gupta was able to reduce the number of points used to only ten which reduced the search space even further resulting in 45 3D Euclidean and 45 geodesic distances. The ten points used for identification are shown in Figure 6-2 and described below:

Exocanthion (ex): the point at the outer corner of the eye. [bilateral].

Endocanthion (en): the point at the inner corner of the eye [bilateral].

Nasion (m'): the point in the midline of both the nasal root and the nasofrontal suture.

Alare (al): the most lateral point on each nostril contour [bilateral].

Pronasale (prn): the most protruded point of the nasal tip.

Cheilion (ch): the point located at each labial corner [bilateral].

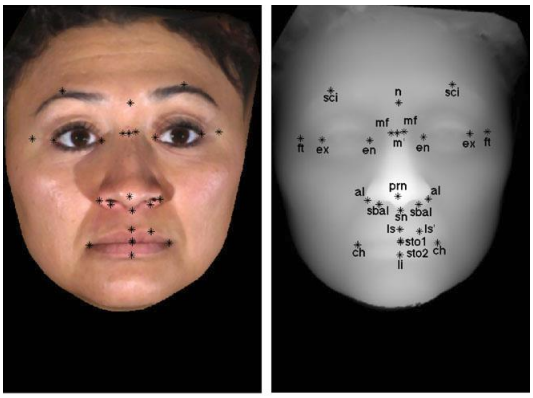

Figure 6-1 "The 25 facial fiducial points associated with highly variable anthropometric facial proportions on (a) a color image, and (b) a range image" Gupta [17]

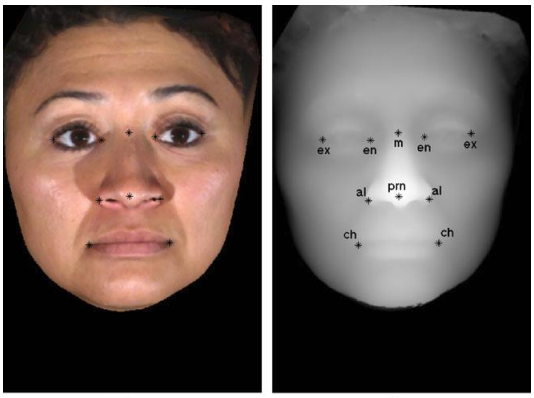

Figure 6-2 "The subset of 10 anthropometric facial fiducial points that were employed for the final automatic Anthroface 3D algorithm depicted on a (a) color, and (b) range facial image" Gupta [17]

After the reduction in the number of facial points, the experiments were run again to confirm the system performed as before at acceptable performance levels. The automatic system recognition rate only using ten points was 96.8% compared to the 97.9% of the 25 points. Additionally the Anthroface system performed significantly better than the holistic methods tested on the same data set where the best performing holistic method only achieved a 92.6% recognition rate [18].

7. Thesis Details

Previous studies have shown biometric measurements can be successfully repeated and by using high resolution 3D imagery this method can discriminate between individuals [17]. Additionally, studies have shown different imaging systems may be used to capture 3D images for use in facial recognition algorithms with equal reliability and recognition rates [13]. It has also been demonstrated the selection of facial landmarks with a high degree of statistical magnitude plays a significant role in accurate facial recognition. Gupta combined much of this previous research to create an anthropometric 3D facial recognition system [17]. In Gupta's

research an expensive and high resolution camera system was employed to capture both 2D and 3D images simultaneously and was the only source of images for the system.

The purpose of this thesis project extends Gupta's original project to use images from an alternative imaging system utilizing inexpensive components off the shelf (COTS). The original plan was to reuse the code and images from the Gupta project as a starting point thereby directly validating the original work along with a direct comparison of capabilities between the new imaging system against the original system. While both the code and images were originally promised only the images were provided. Therefore, this work also recreates the anthropometric 3D facial recognition system based on the information provided in the Anthropometric 3D Face Recognition paper by Gupta [17].

The alternative imaging system to be tested is a matched pair of web cameras arranged as a stereo pair for photogrammetry. The cameras will be arranged as a horizontal pair. The cameras are models currently available and inexpensive. The software package used to create the imaging system is also currently available and is open source making it a no cost alternative to a commercially available API.

7.1. Development Hardware

The primary system used for the project is a Dell Latitude E6430 with an Intel Core i5-3340M CPU running at 2.7 GHz, 8GB of system memory and the operating system is Windows 7 Enterprise 64 Bit Service Pack 1. Two cameras were used during the development of the image capture and facial recognition software. One camera was a Logitech Webcam C250. This is a low quality manual focus style web camera with a USB interface. The maximum true image size is 640x480 VGA style image on a CMOS sensor. The field of view is 63˚ and the Webcam C250 has a 2mm focal length. The second web camera was a Microsoft LifeCam VX-3000. This camera is also a low quality manual focus camera with a USB interface. The maximum image

size is also 640x480 VGA using a CMOS sensor. The field of view is less at 55˚. The lens focal length is not available for the camera. Both cameras are USB 1.1 devices.

Though not a matched pair of cameras their purpose was to provide inputs for the development of the photogrammetry software. The cameras provided a platform to enable the development of the individual camera calibration routines and the stereo calibration of a pair of cameras. No images from these cameras were used for facial point detection or range image creation during the project.

7.2. Testing Hardware

The same computer and operating system was used for the testing. Two Logitech C310 USB web cameras were used for image capture and range image creation. The C310 cameras have a fixed focus eliminating a calibration issue with different focal distances between the cameras. The native true image capture size is 1280x960 using a CMOS sensor with a focal length of 4.4mm and field of view is slightly smaller at 60˚. The video capture image size is still 640x480 VGA.

The C310 camera is a vast improvement in image quality over the previous cameras. The lenses of the new cameras are much more uniform having less distortion than the development cameras. The true image capture size is also much larger and although for video capture the image size is downsized to 640x480 the increased size of the CMOS sensor is apparent in the image quality. Video capture is used over single image capture to provide a live preview of the subject during the capture process in order to ensure the subject appears appropriately in the frame of both cameras.

7.3. Software Development

The development of the software was accomplished as two separate modules. The first module was to create an imaging system capable of creating range images using

photogrammetry and the second part is an image processing component used to detect facial points. Developing these modules independently and separately permitted isolating the imaging system setup from the image capturing routine and depth map creation. The software is written in C++ and is compiled into a single menu driven program for operational simplicity.

7.3.1. Open Source Computer Vision Library

The Open Source Computer Vision Library (OpenCV) is a computer library of image processing functions. Officially launched by Intel in 1999 as a computer vision project OpenCV has since expanded to over 2500 algorithms many of which are optimized. It is now freely available under the Berkeley Software Distribution (BSD) license. OpenCV has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS making it a versatile and portable choice for developing computer software which could be widely distributed. OpenCV has utilities to perform the complex calculations to create a stereo pair of cameras suitable for photogrammetry and also has utilities to convert photogrammetry images into depth maps. These properties of OpenCV made it an obvious choice to use for the development of imaging manipulation software.

7.3.2. Photogrammetry Software Development

There exist several code examples and tutorials describing various aspects of photogrammetry using OpenCV, however, none are complete solutions. The creation of a photogrammetry system in OpenCV is called creating a stereo pair. The creation of a stereo pair requires three main parts. First each camera must be individually calibrated to obtain the intrinsic properties of the camera[1]. Next, the cameras as a pair must be calibrated to each other to rectify the images taken by each camera. Finally the rectified images are used to create a

[1] Intrinsic camera properties are the geometric distortions introduced by the camera's lens and the distortions introduced by the alignment of the focal plane with the lens.

range image or depth map from the stereo pair which is detailed in section 7.3.3.1 later in this paper.

7.3.2.1. Calibrating Individual Cameras

The first step of creating a stereo pair is to calculate each camera's intrinsic properties. Cameras of most types have lenses which tend to distort images. These distortions are unique to each camera and the distortions must be removed from images taken by the camera prior to the image rectification process. OpenCV camera calibration uses a pin hole camera model for calibration. The real world when viewed through a lens is radially distorted, that is the center of the lens at its focal length has no distortion but moving away from the center distorts the image. Lens distortion is commonly seen as the fish-eye effect. Ideally the focal plane of the camera is perpendicular to the lens but if not, a tangential distortion is introduced to the image. Tangential distortions must also be removed from images captured by the camera prior to processing the images. The camera calibration provides the intrinsic and extrinsic camera properties and must be completed on each camera individually. The process tocalibrate the camera starts with capturing images of a calibration pattern, it then records the calibration pattern points and finally calculates the calibration parameters.

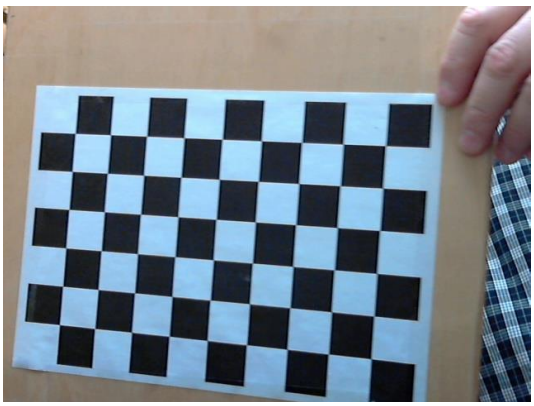

Figure 7-1 Chessboard calibration pattern used during calibration of individual cameras and the stereo pair

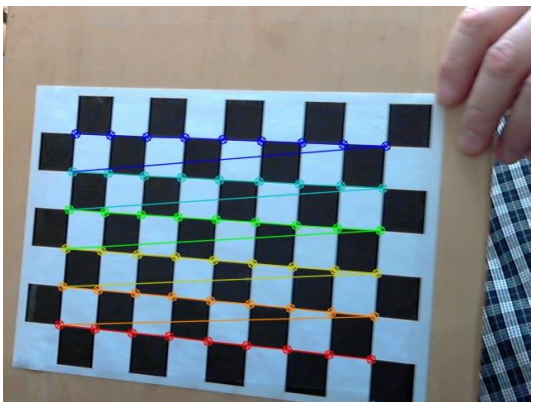

Figure 7-2 Image showing the detected corners in the calibration pattern after refinement by cornerSubPix()

```
<?xml version="1.0"?>
<opencv_storage>
<calibration_Time>"02/10/14 17:41:12"</calibration_Time>
<Camera_0_Matrix type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    8.2893656874672467e+002 0. 3.3605186008494741e+002 0.
    8.3343473661358769e+002 2.1526694019300902e+002 0. 0. 1.
  </data></Camera_0_Matrix>
<Camera_0_Distortion_Coefficients type_id="opencv-matrix">
  <rows>1</rows>
  <cols>8</cols>
  <dt>d</dt>
  <data>
    1.4965613227198144e+001 2.0567968369298391e+000
    -1.3526565029281793e-002 7.9077444540528678e-003
    -3.1622035238879118e+001 1.5029913940334481e+001
    2.6033090951437354e-001 -2.3602277330033541e+001
  </data></Camera_0_Distortion_Coefficients>
</opencv_storage>
```

Figure 7-3 Example single camera calibration data showing the extrinsic (matrix) and intrinsic (distortion coefficients) camera matrices.

OpenCV uses either a circle or square chessboard pattern for the calibration process. The camera is used to capture a series of images showing the calibration pattern. Only a small number of images are needed to calibrate a camera, however a larger number of images provides better calibration results. An OpenCV utility function, `findChessboardCorners()`, locates the calibration pattern in the image and determines the pixel coordinates of the pattern. These coordinates are then refined to a sub pixel accuracy using another utility, `cornerSubPix()`, which are saved in a matrix. Once all the images have been processed and calibration points saved they are passed to the `calibrateCamera()` utility which computes the intrinsic and extrinsic camera properties to remove lens distortions. The camera properties are saved and used in the calibration of the stereo pair or to remove the distortion from individual images.

7.3.2.2. Calibrating the Stereo Pair for Photogrammetry

To create a stereo pair of cameras the physical geometry between the cameras must be determined. Even with great care it is not possible to perfectly align both the principle point and focal plane exactly between two or more cameras, therefore, calibration is required. The

parameters produced by the calibration process virtually align the principle points of the cameras vertically and virtually align the focal planes to lie on the same plane. The calibration process is similar to single camera calibration involving the calibration pattern, nearly simultaneous image capture by both cameras, image pattern detection and finally the stereo rectification parameters are calculated.

Much like single camera calibration either the circle or square chessboard pattern is used in the calibration. Although the calibration algorithms in OpenCV can initially estimate the individual camera parameters during calibration, better results are obtained by using the pre-computed intrinsic and extrinsic camera properties obtained from performing the single camera calibration. Like single camera calibration, larger numbers of captured images containing the calibration pattern provides better calibration of the stereo pair than just a few images. Image capture for stereo calibration requires simultaneously capturing the same scene. Though it is not possible to perform a truly simultaneous image capture, programmatically, OpenCV has an alternative two-step image capture process which first captures the raw camera data without processing the data into an image format. Capturing the raw camera data is very fast providing nearly simultaneous data capture. The raw data is then processed into images later for the rest of the calibration sequence.

Each camera image is processed by the same utility functions used for single camera calibration to locate the calibration pattern and to refine the sub-pixel coordinates. The data generated is saved separately for each camera. Once all captured images have been processed and data stored, the stereo calibration utility, `stereoCalibrate()`, is called which produces the rotation and translation matrices required to later rectify stereo images for 3D image processing. The stereo calibration utility with the correct parameters may also modify the

```
<?xml version="1.0"?>
<opencv_storage>
<calibration_Time>"02/12/14 13:23:58"</calibration_Time>
<Camera_0_Matrix type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    8.2198494105684244e+002 0. 3.0500031186848827e+002 0.
    8.2735997321004834e+002 2.2472876340869371e+002 0. 0. 1.</data></Camera_0_Matrix>
<Camera_1_Matrix type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    8.2199380050031164e+002 0. 2.9813647364329819e+002 0.
    8.2758558726540582e+002 2.2778843221502066e+002 0. 0. 1.</data></Camera_1_Matrix>
<Camera_0_Distortion_Coefficients type_id="opencv-matrix">
  <rows>1</rows>
  <cols>8</cols>
  <dt>d</dt>
  <data>
    1.9654826992518516e-002 2.6734744590365449e+000
    -1.0149620955787142e-002 -5.5092494605702988e-003
    1.6290982698817487e+001 -1.5004422819375411e-002
    2.9728485743185833e+000
1.5257190064722012e+001</data></Camera_0_Distortion_Coefficients>
<Camera_1_Distortion_Coefficients type_id="opencv-matrix">
  <rows>1</rows>
  <cols>8</cols>
  <dt>d</dt>
  <data>
    -2.8679209812347844e-001 -1.0167566231306061e+001
    -1.1080473508340231e-002 -1.0758234181904417e-002
    9.8855694132002981e+001 -2.9426755497812901e-001
    -1.0638115575374641e+001
1.0222729773977079e+002</data></Camera_1_Distortion_Coefficients>
<Mat_R type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    9.9958598010511335e-001 -2.3332255256444109e-002
    1.6836693319891821e-002 2.3456548099032862e-002
    9.9969876461324936e-001 -7.2229067570324214e-003
    -1.6663094807919429e-002 7.6148470366228004e-003
    9.9983216360348759e-001</data></Mat_R>
<Mat_T type_id="opencv-matrix">
  <rows>3</rows>
  <cols>1</cols>
  <dt>d</dt>
  <data>
    -7.4036862143921354e+001 -3.5048814343695756e-001
    1.3532140664111973e+000</data></Mat_T>
</opencv_storage>
```

Figure 7-4 Stereo calibration parameters with modified individual camera calibration data and stereo rotation (Mat_R) and translation (Mat_T) matrices.

input individual camera calibration intrinsic values. For this project the stereo calibration utility was set to modify the individual camera calibration intrinsic values.

7.3.3. Image Capture and Depth Map Creation

After calibration, the stereo camera pair is ready for image capture. The process of creating a depth map from a pair of images is again accomplished using a variety of OpenCV utilities. First, the images must be rectified to each other. Once rectified, the two images are combined to produce a depth map suitable for use in the facial recognition portion of the project. Image rectification and depth map creation are discussed in detail below.

7.3.3.1. Creating Rectified Images

To create a range image from a stereo pair the images from each camera must be rectified to each other. This process re-projects each image to lie on the same coordinate plane. The rectification process also aligns the pixels in the resulting images so each pixel row in the first image has corresponding pixels in the same row of the second image. Rectifying the images in this way simplifies depth map creation so only a single row of pixels is considered for correspondence.

The first step is to compute the *x* and *y* coordinate matrices for the re-projection using OpenCV's `initUndistortRectifyMap()` utility which uses each camera's intrinsic and extrinsic camera properties (modified by the stereo calibration) along with the rotation and translation matrices in the computation. The *x* and *y* coordinate matrices are calculated for each camera independently. Each captured image is then remapped using the *x* and *y* coordinate matrices through another OpenCV utility, `remap()`, which performs the image manipulations to produce rectified images. At this point each image is rectified and ready for use to create a depth map. Figure 7-5 below shows an image prior to the rectification process; Figure 7-6 shows the same image after rectification.

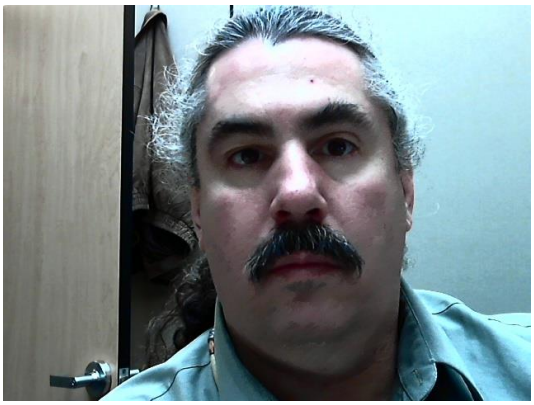

Figure 7-5 Original image prior to rectification

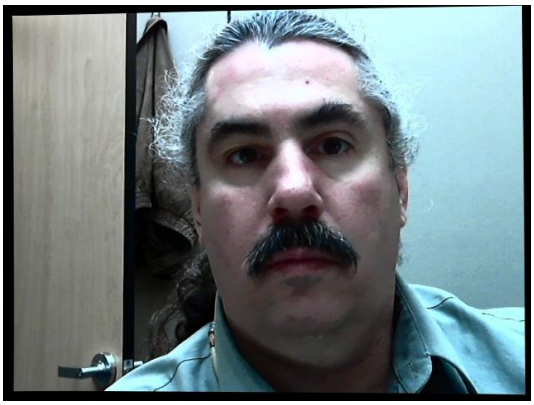

Figure 7-6 Image from Figure 7-5 after rectification

Correspondence lines are the lines of pixels in one image that are on the same row as pixels in the second image. A region of interest is also computed and saved as a rectangle object. The region of interest includes all pixels from one rectified camera image also appearing in the second rectified camera image on the same correspondence row. The regions of interest are the same size and contain only pixels on the same correspondence epilines; pixels not on the same correspondence line in both images are excluded.

7.3.3.2. Creating Depth Maps from Rectified Images

Rectified images are used to create a depth map. The OpenCV has two utility methods which may be used to create a disparity map both of which are block matching algorithms. The first is the `StereoBM()` utility and the second is the `StereoSGBM()` utility. Of the two the

Figure 7-7 is an example of a rectified stereo pair showing the regions of interest and correspondence epilines.

utilities the second is a semi-global block matching method having better overall results but the tradeoff is that the algorithm is slower. For this project, since speed is not a consideration, the `StereoSGBM()` utility is used to create the disparity map. The disparity map produced contains the data necessary to create the depth map but the data converted to image format produces a seemingly blank image. To make the disparity map visible for interpretation the disparity map is scaled to enhance the visibility of the disparity regions. The scaling is what creates a usable depth map. The disparity image can also be used to create a point cloud where each pixel coordinate then contains a distance from the stereo cameras. Although the calculated distances are not used in this project due to different means of calculating the *z* coordinate the calculated distances produced from the disparity image may be useful in other facial recognition systems.

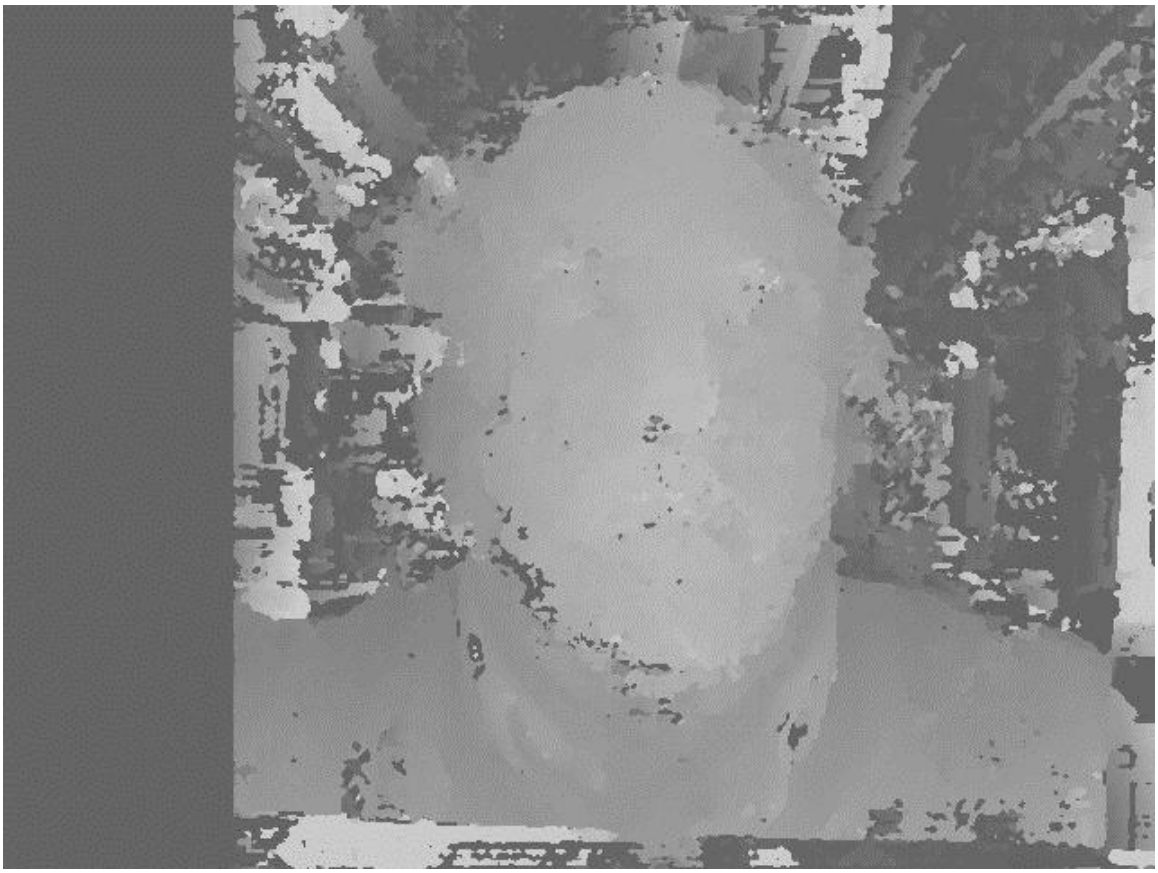

Figure 7-8 Example disparity image after scaling showing a depth map produced from a calibrated stereo pair.

7.4. Image Manipulation

For the facial detection algorithm to work, both a color portrait image and a grayscale depth map are needed. To better approximate the range images contained in the Texas 3D Face Recognition Database (3DFRD) the images captured by the photogrammetry system needed to be cropped, scaled and modified. Some modifications were made to make the images work better with the fiducial point detection algorithm created for this project while other modifications were used to clean up the images[2]. For the portrait image, the left camera rectified image was used because the face position in the image was identical in size and position to the face image in the depth map.

The depth map produced by the `StereoSGBM()` utility is poor, visually, even after scaling. Analyzing the image showed it had a deficiency in the histogram where most of the spectrum was grouped around the central area of the spectrum. Equalizing an image remaps the image spectrum in a way that spreads the spectrum as much as possible over the entire range of values and also flattens the histogram curve. Equalizing the histogram also has the effect of bringing out the details contained in an image. The scaled depth maps were first converted to grayscale images and then equalized using the OpenCV `equalizeHist()` utility. Doing this produced a depth map with more definition better representing the face. Similarly the portrait image was

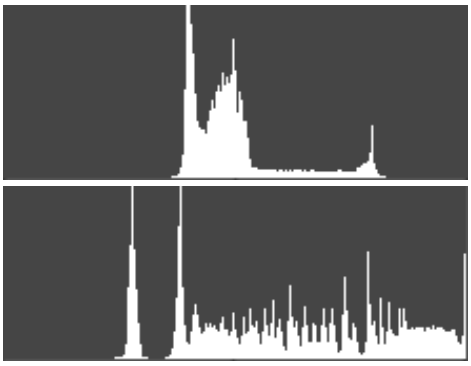

Figure 7-9 Depth map histograms. The upper histogram is before equalization; the lower histogram is after equalization.

[2] The fiducial points used are detailed in section 6 above and the algorithms used to detect the points are detailed in sections 7.4.1.1 thru 7.4.1.5 below.

also equalized but it was converted to a YCrCb color space[3]. The image channels were split apart and the luminosity component, Y, was equalized. The channels were recombined to restore the color portrait image and reconverted to the BRG color space.

To bring out more detail in the images, a sharpening step was utilized. The method of unsharp masking has been used for many years to increase the apparent resolution of an image. The unsharp masking process used was a simplistic two step procedure. The first step is to blur the original image which is done with the OpenCV `GaussianBlur()` utility. Using a Gaussian blur function can potentially eliminate some of the edges which may be present in the image in a sharpened final image with fewer edges. The second step is to add original and blurred images together weighting each image independently. The original image is given a positive weighting while the blurred image is given a negative weighting. The overall result is a sharpened image which looks as if it was taken at a higher resolution. Unsharp masking was performed on both the depth map and the portrait image.

The next step in the image processing is to detect the facial region and blacken everything outside the face. OpenCV has built-in machine learning capabilities however it also has prebuilt classifiers. Using the `CascadeClassifier()` utility and loading the already trained data contained in the `haarcascade_frontalface_alt.xml` file it was a simple matter to locate the face in both the depth map and portrait image. In some cases more than one face was found in the depth map or portrait image; in such cases the larger of the two facial regions was retained as the final facial region. The rectangular facial regions returned for the depth map and portrait image are compared and the larger of the two was again retained. All pixels outside the detected facial region are set to black using the final retained rectangle.

[3] The YCrCb color space separates luminance, or brightness, from color. Luminance is represented by the Y component. The Cr and Cb components are color scales. The Cr component is the red-cyan color contribution while Cb component is the blue-yellow contribution.

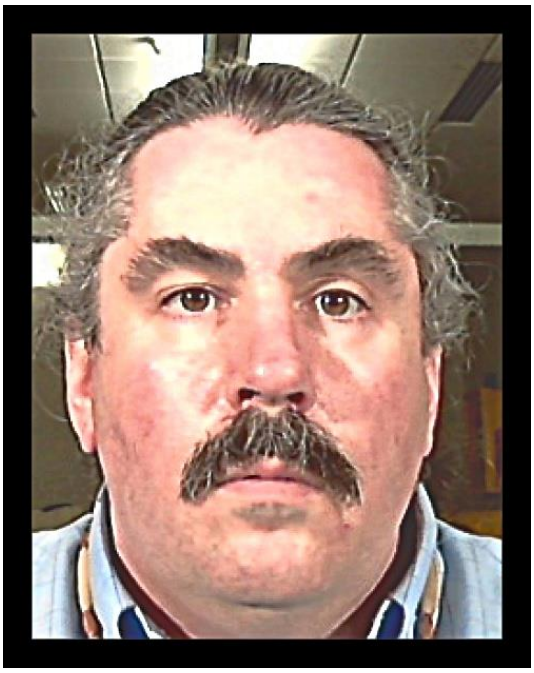

Figure 7-10 Final portrait image after cropping, scaling, equalizing and sharpening

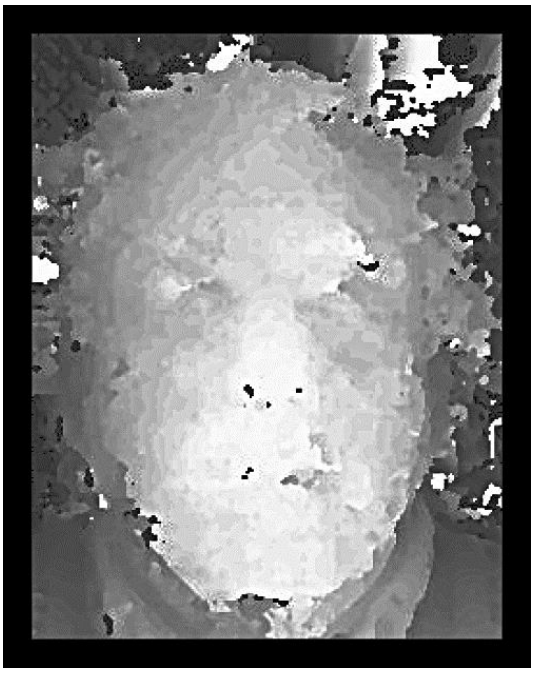

Figure 7-11 Final depth map after cropping, scaling, equalizing and sharpening

The resulting images were now a closer match to those contained by the 3DFRD dataset. However, captured images still differed significantly in size. The facial region previously used to blacken the area outside the face was again used to crop the images. The facial region was expanded by fifteen pixels on each side to retain a black border around the face. Both the depth map and color portrait images were cropped from the original images using the expanded facial region. Both cropped images are then scaled to be the same width as the 3DFRD images leaving them to differ in size only by height. As a final step in the image modification process both the range and portrait images are again equalized and sharpened as previously described.

7.4.1. Anthropometric 3D Point Detection Development

Unable to obtain the original software written by Gupta it was necessary to create a facial landmark detection system. Using Gupta's work on choosing which landmarks to include in the facial recognition process and the order of detection, work ensued to detect each facial landmark. The major difference between this detection system and the Gupta system are the methods used to locate each facial landmark. Gupta frequently used a Gaussian curvature

algorithm in locating specific features like *prn*, the tip of the nose. OpenCV does not have a Gaussian curvature function so alternative methods were employed. The techniques used include scanning images for average or specific values, using sliding windows and employing various OpenCV utilities. Like Gupta both range and portrait images were used in the feature detection process. Prior to trying to locate each feature general observation of both the range and portrait images along with the data they contained was considered for solving the feature location problem. The ideas and development for locating each facial feature are detailed below.

7.4.1.1. Nose Tip (*prn*)

To locate the nose tip it was observed in both the range and portrait images that the tip of the nose was a bright region in each image. The data contained in the portrait image is color intensity information for each of the color channels in the image while the range images held only a single intensity value. A problem with using the intensity value is any single pixel may have a higher intensity value than any pixel in the nose tip so a sliding window was used. A sliding window over the portrait image alleviates the single pixel problem but still would not work because the white in the eye sclera has a higher intensity than that of the tip of the nose. The range image is a greyscale image which does not have the problems of the color portrait. In the range image the nose tip is the brightest portion of the image although it still suffers from the single pixel problem, here though, the sliding window does solve the single pixel problem.

To locate the nose tip a sliding window is passed over the entire image and the sum of the intensities for the pixels within the box is calculated. The size of the sliding window is set to nine pixels but can be as small as three. However, it must always be an odd number in order for it to have a central pixel. The current sum is compared to the maximum value thus far obtained with one of three possible results. The first result is when the current sum is smaller than the

```
61
62      // ******************
63      // **** Finds PRN (nose tip)
64      // **** Goes through entire range image and looks for brightest spot using a sliding box.
65      // **** Adds all pixels together in order to come up with a value for the box.  If the new
66      // **** box value is larger then all old points are dropped and new center point selected.
67      // **** If the values are the same a new center point is added to the vector.  If more
68      // **** than one point is found the x values are averaged and the y values are averaged to
69      // **** determine the location of PRN.
70      for(int i = 0; i + BOX_SIZE < image_range_gray.cols; i++){                    // Walk through image making sure not to go outside of image boundary
71          for(int j = 0; j + BOX_SIZE < image_range_gray.rows; j++){
72              sum = 0;
73              for(int x = 0; x < BOX_SIZE; x++){                                   // Loop through the box and calculate a new sum of pixel values
74                  for(int y = 0; y < BOX_SIZE; y++){
75                      unsigned char curr = data[(image_range_gray.cols * (i + y)) + (j + x)];
76                      sum += (int)curr;
77                  }
78              }
79              if(max < sum){                                                       // Dump all previous points for new brighter center
80                  points.clear();
81                  points.push_back(Point(j + (BOX_SIZE / 2), i + (BOX_SIZE / 2)));
82                  max = sum;
83              }else if(max == sum){                                                // Add new point to vector of points with same brightness
84                  points.push_back(Point(j + (BOX_SIZE / 2), i + (BOX_SIZE / 2)));
85              }
86          }
87      }
88      for(size_t i = 0; i < points.size(); i++){                                   // Sum up x and y values for located points
89          prn.x += points[i].x;
90          prn.y += points[i].y;
91      }
92      prn.x = prn.x / (int)points.size();                                          // Divide by number of points to get average
93      prn.y = prn.y / (int)points.size();
94
95      facial_points_result.push_back(prn);                                         // Add points to result vector
96
```

Figure 7-12 Code segment from automatic.cpp showing the algorithm for finding the tip of the nose (prn).

maximum it is simply discarded. The second option is when the current sum is the same as the maximum the location of the central pixel from the window is saved. The last option is when the current sum is larger than the maximum it then discards all previously saved points and saves the current window's central pixel location. After the range image scan has completed, the saved central pixel locations *x* and *y* coordinate positions are averaged to compute the nose tip *x* and *y* coordinate location.

7.4.1.2. Nose Width (*al* – *al*)

The width of the nose alar is clearly visible in both the range and portrait images. The outer shape of the nose in this area presents as a smooth curved line suitable for edge detection. Using OpenCV's Canny Edge Detection algorithm both the range and portrait images produced lines in this area. However the range image presented a cleaner and

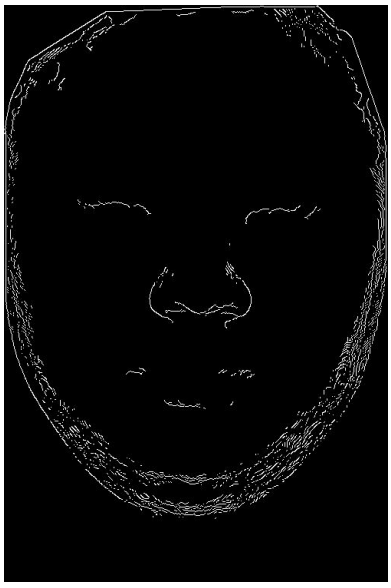

Figure 7-13 Image showing edges detected by the Canny Edge detector

```
101     // ******************
102     // **** Finds right AL location (nose width)
103     // **** Using range image run it through the Canny edge detector to locate the nose width
104     // **** outline.  Starting with PRN go right until white pixel is found.  Once found keep
105     // **** going right one pixel at a time until a column with no white pixels is found.
106     // **** Locate right AL in center of last white line.
107     // ******************
108     dst = detect_canny_edges(image_range, lowThreshold, kernel_size);                    // detect edges in range image
109     cvtColor(dst, dst_gray, CV_BGR2GRAY);
110     data = (unsigned char*)dst_gray.data;
111
112     al_right = prn;
113     x_stop = prn.x - SEARCH;                                                             // set stop value, in pixels
114     if(x_stop < 0)x_stop = 0;                                                            // stop must not be less than zero
115     y_start = prn.y - SEARCH_AL;                                                         // set start value - in pixels
116     if(y_start < 0)y_start = 0;                                                          // start value must not be less than zero
117     y_stop = prn.y + SEARCH_AL;                                                          // set stop value - in pixels
118     if(y_stop > dst_gray.rows)y_stop = dst_gray.rows;                                    // stop vlaue must not be larger than picture he
119
120     for(int x = prn.x; x > x_stop; x--){                                                 // move right of prn until white pixel is found
121         if((int)data[(dst_gray.cols * prn.y) + x] == GRAYSCALE_WHITE){
122             al_right.x = x;
123             break;
124         }
125     }
126
127     count = 0;                                                                           // for finding al, exit when value is 3
128     for(int x = al_right.x; x > x_stop; x--){                                            // look further right for column of pixels with
129         int top = -1, bottom = -1;                                                       // top and bottom of white line
130         for(int y = y_start; y < y_stop; y++){                                           // look down column for white pixel
131             int curr = (int)data[(dst_gray.cols * y) + x];
132             if(curr == GRAYSCALE_WHITE && top == -1){                                    // found top of white line
133                 top = y;
134                 bottom = y;
135             }else if(curr == GRAYSCALE_WHITE && top != -1){                              // found bottom of white line
136                 bottom = y;
137             }
138         }
139         if(top != -1){                                                                   // found white pixels in column
140             al_right.x = x;                                                              // set new column value
141             al_right.y = top + ((bottom - top) / 2);                                     // set new row value half way down white line
142             count = 0;
143         }else{
144             if(count == 3)break;
145             count++;
146         }
147     }
148
149     facial_points result.push_back(al_right);                                            // add point to results
```

Figure 7-14 Code segment from automatic.cpp showing the algorithm for finding the right nose width (al). The algorithm for the left nose width is similar.

distinct image. By observation, the left and right alar are both lateral of and normally below the tip of the nose. Therefore, a vertical search region was set to be forty pixels above and below the detected nose tip. The horizontal search region stops 85 pixels from the nose tip. The detection of the right nose alar is accomplished by moving to the right of the previously detected nose tip until a line of white pixels is found. Once a line is found, vertical columns of pixels are scanned for the continuation of the line. Each column where a line is detected, the line center is calculated and the point for the right alar is set. Scanning columns continues until three columns of pixels are scanned where no line was found at which time the algorithm exits. Locating the left alar uses a similar process but obviously searches the opposite direction.

7.4.1.3. Nose Bridge (*m'*)

The point for the nose bridge, *m'*, is located at the juncture of frontal suture and nasal bone suture. Obviously it is not possible to directly locate this point on living subjects therefore it is approximated to be the lowest point of the nose bridge above the nose tip. Only the range image can provide this information so it was used to determine this point. A starting point for the search was offset twenty pixels above the nose tip while a stopping point was selected to be 150 pixels above the nose tip. The algorithm simply moves directly upwards in the range image checking sequentially for lower locations. Locations having the same height are simply counted. Once a location is found that is higher it stops and calculates the *y* coordinate of the nose bridge. If the count is greater than one, one half of the count is subtracted from the current *y*

```
197     // ******************
198     // **** Finds M' (nose bridge)
199     // **** Using range image move up from the tip of the nose which should always be darker
200     // **** than the tip.  Then when it starts climbing stop, move left and right from that
201     // **** point looking for highest point.
202     // ******************
203     m_prime = prn;
204     data = (unsigned char*)image_range_gray.data;
205     y_start = prn.y - 20;                                          // set start value
206     if(y_start < 0)y_start = 0;                                    // start value must not be less than zero
207     y_stop = prn.y - 150;                                          // set stop value, in pixels
208     if(y_stop < 0) y_stop = 0;                                     // stop value must be no less than zero
209     last = GRAYSCALE_WHITE;
210     count = 0;
211     for(; y_start > y_stop; y_start--){                            // move up from prn to find low spot
212         int curr = (int)data[(image_range_gray.cols * y_start) + m_prime.x];
213         if(last == curr){                                          // no change in height
214             count++;
215         }else if(last > curr){                                     // moved lower
216             count = 1;
217             last = curr;
218         }else if(last < curr){                                     // moved higher, set y and exit loop
219             m_prime.y = y_start + (count / 2);
220             break;
221         }
222     }
223
224     x_start = prn.x - BRIDGE_SEARCH;                               // set start value
225     if(x_start < 0) x_start = 0;                                   // start value must not be less than zero
226     x_stop = prn.x + BRIDGE_SEARCH;                                // set stop value
227     if(x_stop > image_range_gray.cols)x_stop = image_range_gray.cols;  // stop value must not be wider than image
228     last = GRAYSCALE_BLACK;
229     count = 0;
230     for(;x_start < x_stop; x_start++){                             // move left to right to find high spot
231         int curr = (int)data[(image_range_gray.cols * m_prime.y) + x_start];
232         if(last == curr){                                          // no change in height
233             count++;
234         }else if(last > curr){                                     // moved lower, set x and exit loop
235             m_prime.x = x_start - (count / 2);
236             break;
237         }else if(last < curr){                                     // moved higher
238             count = 1;
239             last = curr;
240         }
241     }
242
243     facial_points_result.push_back(m_prime);                       // add m_prime to vector of facial points
```

Figure 7-15 Code segment from automatic.cpp showing the algorithm for finding the nose bridge (m')

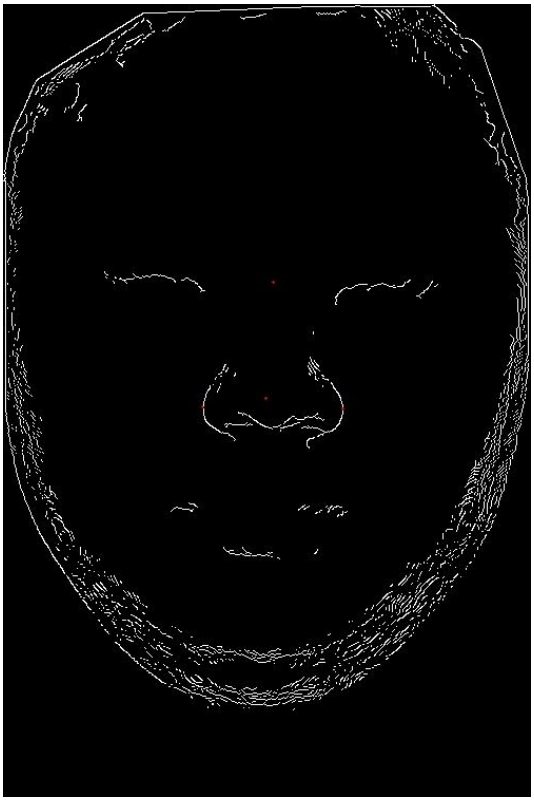

Figure 7-16 Image showing Canny detected edges with nose points prn, al, al and m'

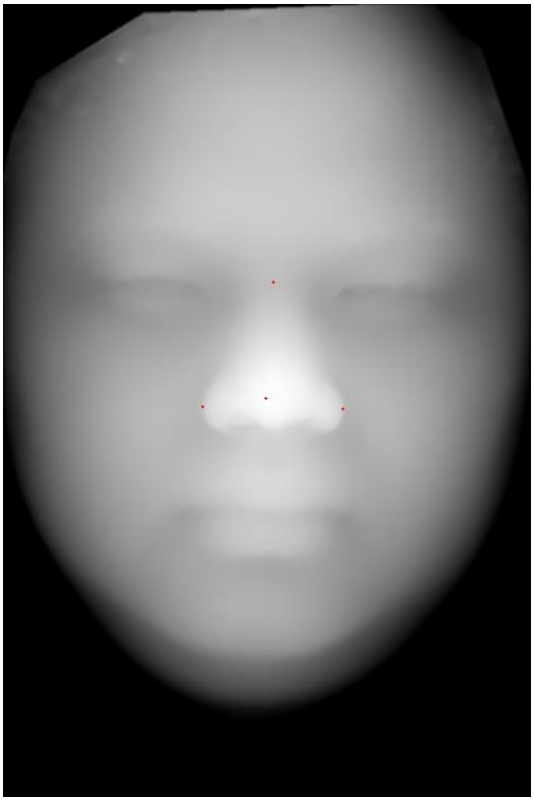

Figure 7-17 Range image showing the location of the nose points prn, al, al and m'

coordinate. The *x* coordinate is located similarly by moving to the right of the current location by fifty pixels but this time the algorithm is locating the highest point.

7.4.1.4. Eye Corners (*en* and *ex*)

The eye corners are more clearly observable in the portrait image than in the range image therefore the portrait image was used. Through observation of the eye portion of the portrait image it was clear the iris and pupil formed a distinct feature essentially round in nature and approximately the same height in the image as the eye corners and centrally located between the eye corners. It was also observed the eye centers were closely aligned with the previously detected nose bridge, *m'*. Locating the eye centers was therefore considered important in locating the eye corners. The algorithm for locating the eye corners for the left and right eyes is the same although starting and ending search locations are different. The algorithm is the most

complex of all the facial point detection algorithms. The algorithm first estimates the pupil centers followed by locating the eye corners using the estimated pupil centers.

A similar approach to finding the noses tip was applied to locating the center of the pupil. As a starting point the nose bridge location is used to create vertical search regions forty pixels above the nose bridge to fifty pixels below the nose bridge. Horizontally the search region is offset from the nose bridge by 200 pixels. A search window is this passed over the search region looking for the least luminescent region. A region with a higher luminosity value than the

```
// ******************
// **** Find eye corners
// **** Locates the en and ex using the grayscale portrait image.  Method uses the same
// **** goodFeaturesToTrack method to locate points of interest in the image.  Points found
// **** are then reduced to those around the eye region with the final points seleted as the
// **** eye corners (en and ex).
// **** Input:      Point center - the approximated center location of the eye
// ****             int side - 0 indicates left, 1 indicates right
// ****             Mat image - grayscaled portriat image
// **** Returns:    Vector<Point> corners - the located eye corners ordered inside (en) then
// ****                 outside (ex)
// ******************
vector<Point> find_eye_corners(Point center, int side, Mat image){
    vector<Point> corners;                                                          // holds the located eye corners
    vector<Point2f>  good_corners, temp_corners;                                    // temporary containers for holding possible cc
    goodFeaturesToTrack(image, good_corners, 300, 0.01, 20.0, noArray(), 5, false, 0.04);

    // Add found corners to temp_corners
    for(unsigned int i = 0; i < good_corners.size(); i++){                          // Go through found corners and find all those
        if(good_corners[i].x <= center.x + 70 && good_corners[i].x >= center.x - 70){       // horizontal region around eye center
            if(good_corners[i].y <= center.y + 30 && good_corners[i].y >= center.y - 30){   // vertical region around eye center
                temp_corners.push_back(good_corners[i]);                            // save corners in eye region
            }
        }
    }

    good_corners.clear();                                                           // clear all found corners
    good_corners.swap(temp_corners);                                                // replace with those from eye region

    do{                                                                             // Find two corners closes to the eye center ve
        unsigned int smallest = UINT_MAX;
        int distance = INT_MAX;
        for(unsigned int i = 0; i < (unsigned int)good_corners.size(); i++){
            if(cvRound(abs(center.y - good_corners[i].y)) < distance){
                if(good_corners[i].x < center.x - 25 || good_corners[i].x > center.x + 25){ // corner must be outside of the iris region
                    smallest = i;
                    distance = cvRound(abs(center.y - good_corners[i].y));
                }
            }
        }
        temp_corners.push_back(good_corners[smallest]);                             // save corner with smallest vertical distance
        good_corners.erase(good_corners.begin() + smallest);                        // delete the found corner from the source vect
    }while((int)temp_corners.size() < 2);                                           // stop when two corners are found

    if(side == LEFT){                                                               // order eye corners by inside then outside
        if(temp_corners[0].x < center.x){
            corners.push_back(temp_corners[0]);
            corners.push_back(temp_corners[1]);
        }else{
            corners.push_back(temp_corners[1]);
            corners.push_back(temp_corners[0]);
        }
    }else{
        if(temp_corners[0].x < center.x){
            corners.push_back(temp_corners[1]);
            corners.push_back(temp_corners[0]);
        }else{
            corners.push_back(temp_corners[0]);
            corners.push_back(temp_corners[1]);
        }
    }
    return corners;
}
```

Figure 7-18 Code segment from automatic.cpp showing the algorithm for finding the eye corner points (en and ex) based on the eye center locations

current value is skipped while a region with a lower luminosity value causes the new value to replace the current value and all previously stored center points are discarded; if the region is the same luminosity then the central point of the search window is stored. The final location of the eye center is averaged from all the located least luminous points found.

To locate the eye corners the OpenCV `goodFeaturesToTrack()` utility is used to identify all points in the portrait image which may be of interest. The resulting list of points is trimmed to contain only those within seventy horizontal pixels and thirty vertical pixels of the eye center. The reduced set is iterated over to locate the two points with the smallest difference in vertical distance from the eye center and also outside the iris region set to be a 25x25 pixel box around the detected eye center. A point selected to be an eye corner is removed from the reduced set of points and the process repeats until two eye corners are found. The process of locating the eye corners is run separately for the left and right eyes.

7.4.1.5. Mouth Corners (*ch*)

While working on the eye corners algorithm using the portrait image it was observed that the OpenCV `goodFeaturesToTrack()` method also detected two points, *ls* and *sto1* in Figure 6-1. Looking at the images it was clear the points *ls* and *sto1* could be used to detect the mouth corners, *ch*, because they are nearly vertically inline and below the already detected nose tip, *prn*. In addition *sto1* is nearly horizontally aligned with the left and right mouth corners being sought. A final benefit of OpenCV's `goodFeaturesToTrack()` method is that it also correctly located the two mouth corners, *ch*.

The point set returned by OpenCV's `goodFeaturesToTrack()` method are first reduced to three subsets of points using the already detected nose tip as a reference point. The first set contains all the points 130 pixels horizontally offset to the left and right of the nose tip and 50 to 170 pixels vertically below the nose tip. The second and third sets of points are left and right

divisions of the first set again using the nose tip *x* coordinate as the determinant. The first set of points is then searched for the two points with the smallest horizontal difference from the nose tip; the point with the lowest vertical location is retained as *sto1*. The left subset is searched for the point most closely matching the vertical height of *sto1* which is identified as the left mouth corner. The right mouth corner is located similarly using the right subset of points.

```
324     // ******************
325     // **** Find mouth corners
326     // **** Locates the left and right mouth corners using the portrait image.  Starting with
327     // **** prn and using goodFeaturesToTrack() points are located in a region of interest
328     // **** around the mouth.  The nose tip, prn is used to locate sto1 and sto2.  The point
329     // **** sto2 is used to locate both mouth corners, ch.
330     // ******************
331     goodFeaturesToTrack(image_gray, good_corners, 300, 0.01, 20.0, noArray(), 4, false, 0.04);
332
333     for(unsigned int i = 0; i < good_corners.size(); i++){                          // Go through found corners and find all those
334         if(good_corners[i].x <= prn.x + 130 && good_corners[i].x >= prn.x - 130){
335             if(good_corners[i].y >= prn.y + 50 && good_corners[i].y <= prn.y + 170){
336                 temp_corners.push_back(good_corners[i]);                            // add all corners in mouth region to temp_corr
337                 if(good_corners[i].x >= al_left.x)left_corners.push_back(good_corners[i]);  // add all corners in mouth region and left of
338                 if(good_corners[i].x <= al_right.x)right_corners.push_back(good_corners[i]);// add all corners in mouth region and right of
339             }
340         }
341     }
342
343     good_corners.clear();                                                           // Clear all found corners and replace with the
344     good_corners.swap(temp_corners);                                                // empty temp into good
345
346     do{                                                                             // Find two corners closest to PRN horizontally
347         smallest = UINT_MAX;                                                        //      reason is they were always detected and
348         distance = INT_MAX;
349         for(unsigned int i = 0; i < (unsigned int)good_corners.size(); i++){
350             if(cvRound(abs(prn.x - good_corners[i].x)) < distance){
351                 smallest = i;
352                 distance = cvRound(abs(prn.x - good_corners[i].x));
353             }
354         }
355         temp_corners.push_back(good_corners[smallest]);                             // add smallest to temp
356         good_corners.erase(good_corners.begin() + smallest);                        // remove smallest from vector
357     }while((int)temp_corners.size() < 2);                                           // exit when two points are found
358
359     if(temp_corners[0].y < temp_corners[1].y){                                      // determine stomian (where lips meet at center
360         stomian = temp_corners[1];
361     }else{
362         stomian = temp_corners[0];
363     }
364
365     smallest = UINT_MAX;
366     distance = INT_MAX;
367     for(unsigned int i = 0; i < left_corners.size(); i++){                          // find point in left corners closes to stomian
368         if(cvRound(abs(stomian.y - left_corners[i].y)) < distance){
369             smallest = i;
370             distance = cvRound(abs(stomian.y - left_corners[i].y));
371         }
372     }
373     ch_left = left_corners[smallest];
374
375     smallest = UINT_MAX;
376     distance = INT_MAX;
377     for(unsigned int i = 0; i < right_corners.size(); i++){                         // find point in left corners closes to stomian
378         if(cvRound(abs(stomian.y - right_corners[i].y)) < distance){
379             smallest = i;
380             distance = cvRound(abs(stomian.y - right_corners[i].y));
381         }
382     }
383     ch_right = right_corners[smallest];
384
385     facial_points_result.push_back(ch_left);                                        // add mouth corners to the facial points
386     facial_points_result.push_back(ch_right);                                       // add mouth corners to the facial points
```

Figure 7-19 Code segment from automatic.cpp showing the algorithm for finding the left and right mouth corners (ch)

7.4.2. Distance Calculations

To replicate the Gupta method for facial recognition the calculation of distances between detected points is used. Both the Euclidean and geodesic distances are computed for each pair of points[4]. Choosing all pairs of points $\left(\frac{45}{2}\right)$ resulted in a total of 45 3D Euclidean and 45 geodesic distances to be used later for facial recognition

7.4.2.1. 3D Euclidean Distances

Once all the facial points are located, the 3D Euclidean distances between each pair of points is calculated. The calculation of a Euclidean distance in 3D space is a straightforward extension of a 2D Euclidean distance, see Equation 1. The *x* and *y* values used for the calculation were the pixel *x* and *y* coordinates. For the *z* value the intensity value of the range image at the pixel coordinate was used without modification. The Euclidean distances are saved as a vector for storage.

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \qquad \text{Equation 1}$$

7.4.2.2. Geodesic Distances

The distance between pixels in a range image is regular and discrete forming a grid of rows and columns. A line drawn between any two pixel locations may cross discrete row and column locations but is more likely to cross between discrete row and column locations complicating path creation. This project used a simple geodesic distance approximating the path of the straight line between the end points to calculate the geodesic distance. The first task is to create a path between each pair of points which as closely as possible follows the straight line path between the two end points. Once the path is determined the geodesic distance is calculated by summing the 3D Euclidean distances from one point to the next along the path.

[4] A geodesic distance is the distance between two points in three-dimensional space following the surface curvature of the space.

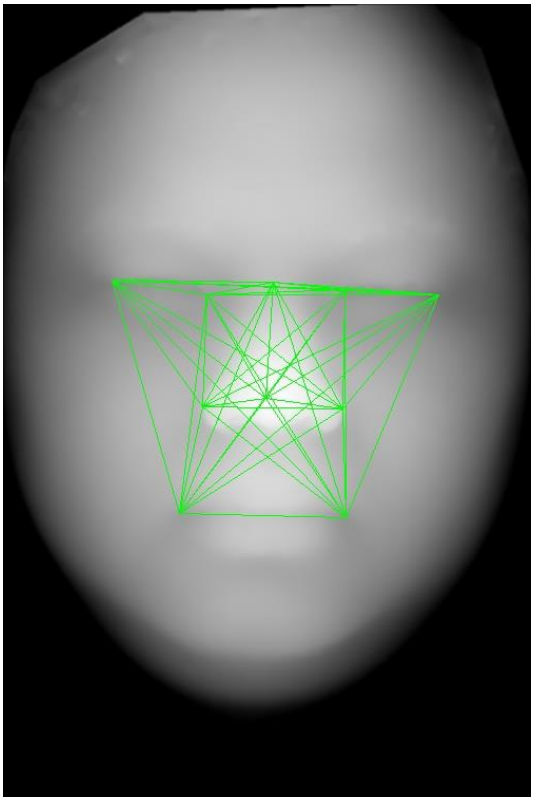

Figure 7-20 Image showing the 45 Euclidean lines on a range image.

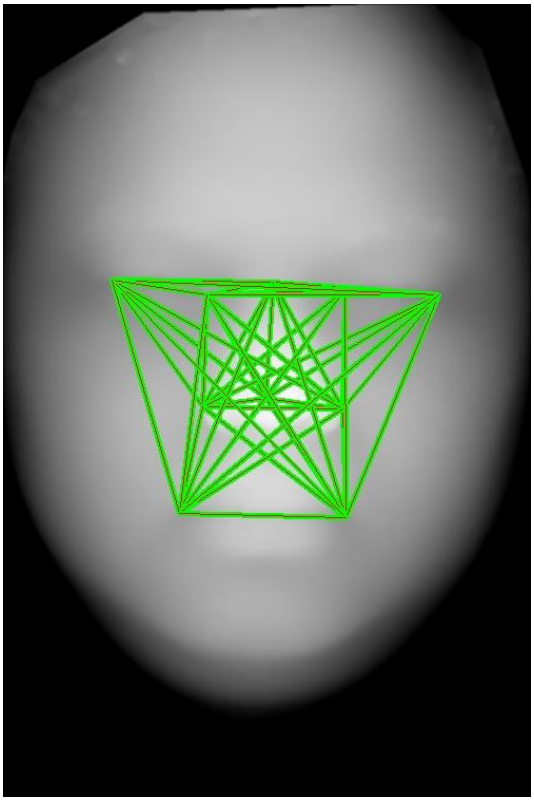

Figure 7-21 Image showing the 45 geodesic lines (red) drawn on top of the Euclidean lines (green)

The resulting distance closely approximates the distance along the surface between the two points.

To create a path of discrete points, a simple distance technique was used to select individual points along the path. The first step is to reduce the problem to a 2D problem by only considering the *(x, y)* coordinate plane and ignoring the *z* values. Also, to create a path at most the change in the *x* and *y* coordinate can be only one pixel at any time in one or both directions. Considering a coordinate plane with start and destination points *A* and *B* any change in *x* or *y* coordinates of the start location more distant from the *x* and *y* coordinates of the destination are in the wrong direction. This information is used to create three candidate points which are all closer to the destination but only one pixel away from the start location. Using the start and end points to define a line *AB* the distance to each candidate point along a line perpendicular to

AB is calculated. The candidate point having the shortest distance to the line *AB* is retained as a point along the path, the start position is updated and the algorithm repeats.

```
/*
The method follows the line segments found in vector<Point> facial_points_result generating
a list of descrete points which follow the line segment as closely as possible. This will
result in a big array with 45 vectors of points.

Return: vector<vector<vector<Point> > > paths       big array of path points

NOTE: The method does not guard agains cases where the line to follow is exactly horizontal
or vertical as this case is not very likely although possible.
*/
vector<vector<vector<Point> > > find_midpoints(){
    paths.clear();                                                          // make sure there is nothing sav
    for(unsigned i = 0; i < facial_points_result.size(); i++){
        vector<vector<Point> > curr_point_path;
        Point curr = facial_points_result[i];                               // one end point of line and star
        curr_point_path.clear();
        for(unsigned j = i + 1; j < facial_points_result.size(); j++){
            vector<Point> path;                                             // stores the path for one line
            int change_x, change_y;                                         // holds change values for x and
            int quad;                                                       // quadrant for direction change
            Point destination = facial_points_result[j];                    // destination
            Point start = Point(curr.x, curr.y);                            // used to walk the line from sta
            path.clear();                                                   // clear the path for use - just

            path.push_back(start);                                          // add the starting point to the

            int diff_x = abs(curr.x - destination.x);                       // change in x
            int diff_y = abs(curr.y - destination.y);                       // change in y
            if(diff_x == 0 && diff_y == 0)continue;                         // same point so skip otherwise w

            abs((curr.x + 1) - destination.x) < diff_x ? change_x = 1 : change_x = -1;      // setting the change in x
            abs((curr.y + 1) - destination.y) < diff_y ? change_y = 1 : change_y = -1;      // setting the change in y

            if(change_x == 1){                                              // determines quadrant using curr
                change_y == 1 ? quad = 4 : quad = 1;
            }else{
                change_y == 1 ? quad = 3 : quad = 2;
            }

            do{
                Point a = Point(start.x + change_x, start.y);               // create new testing points
                Point b = Point(start.x, start.y + change_y);
                Point c = Point(start.x + change_x, start.y + change_y);

                CvPoint2D64f a_perp = closest_point(curr, destination, a, quad);        // get points perpendicular to th
                CvPoint2D64f b_perp = closest_point(curr, destination, b, quad);
                CvPoint2D64f c_perp = closest_point(curr, destination, c, quad);

                double dist_a = euclidean_distance(a, a_perp);              // get distances for each point..
                double dist_b = euclidean_distance(b, b_perp);
                double dist_c = euclidean_distance(c, c_perp);

                if(dist_a <= dist_b && dist_a <= dist_c){                   // save point with closest distar
                    path.push_back(a);
                    start = a;
                }else if(dist_b <= dist_a && dist_b <= dist_c){
                    path.push_back(b);
                    start = b;
                }else{
                    path.push_back(c);
                    start = c;
                }

            }while(path.back().x != destination.x || path.back().y != destination.y);       // do until last element in path
            curr_point_path.push_back(path);                                // save the current path
        }
        paths.push_back(curr_point_path);
    }
    return paths;
}
```

Figure 7-22 Code segment from distances.cpp showing the algorithm for computing the path between two points

After computing a path for each pair of points the geodesic distance is calculated. The geodesic distance is arrived at by summing the Euclidean 3D distances from point to point along the path. The forty-five geodesic distances are saved as a vector for storage in the same order as the Euclidean distances.

8. Project Analysis

There are three main areas of the project to be analyzed. First is the photogrammetry system, next is the modification of the captured images and third is the fiducial point detection system. Each of these areas is briefly discussed citing the successes and some improvements which may be implemented. In general, the overall system did not function as expected however it is clear the system using inexpensive components and software is capable of capturing images, computing disparity and creating depth maps and necessary information useful for facial recognition.

8.1. Photogrammetry System

The discussion of the photogrammetry system is broken into separate discussions of the hardware, photogrammetry software and the OpenCV `StereoSGBM()` utility. Each is discussed separately as they have unique contributions to the outcome of the project as a whole.

8.1.1. Hardware

The desired outcome was to produce depth maps with a high enough resolution to distinctly portray the ten key points of the face as chosen by Gupta. Cost and availability of hardware were driving forces for the selection of the specific cameras used in the project. The cameras were required to be both inexpensive and highly available. In addition the ability to fix the camera focus manually, via a camera API or have a fixed focus was required. The cameras selected met all of these requirements, however, their native resolution is low at only 640x480 pixels. This relatively low resolution negatively affected the detail of the resulting depth maps.

The resulting depth maps have poor detail in all regions of interest. When compared with the depth maps from the 3DFRD collection facial details like the alar of the nose, endocanthion and exocanthion are virtually nonexistent and unsuitable for use with the photogrammetry software. Some of the produced depth maps were in fact so poor some people would have difficulty distinguishing a face in the image. Regardless of the depth map's level of detail the project clearly demonstrated that depth maps of the face can be produced with less expensive camera systems.

8.1.2. Photogrammetry Software

Several steps are required to prepare for capturing images suitable for photogrammetry. Each step must be performed in the correct order and pass appropriate parameters to the next step the process. Once the setup steps are completed and as long as the cameras are not repositioned the process of capturing images for photogrammetry is simply a matter of capturing one image from each of the cameras. The actual photogrammetry can be accomplished at a later time using the saved setup parameters, however, it is prudent to view the range image at time of capture to verify the system is working correctly and that the lighting and subject's distance from the camera are correct. In a production environment, much like photographs taken for identification, the camera system, lighting and subject positioning are all predetermined; similarly the photogrammetry system setup could be predetermined.

8.1.3. OpenCV StereoSGBM() Utility

The OpenCV utilities used to create depth maps are `StereoBM()` and `StereoSGBM()`. Both of these utilities have been developed with segmentation for object detection as their primary purpose. Due to the designed purpose of these utilities the depth maps produced by the underlying algorithms are splotchy in nature somewhat resembling topographic maps. The splotchy nature, though desirable for segmentation, is undesirable for generating smooth

transitions from one area to another within the depth map. The splotchy depth maps, though visually passable as faces, have little definition or detail necessary for use in facial recognition. The reason the depth maps produced in this project are not suitable is the depth at any specific pixel is a depth generalization of the area surrounding the pixel. Countering the detractors above, the `StereoSGBM()` utility is quit fast. It takes about one second to produce a disparity image and scale the disparity image to values that can be viewed visually.

8.2. Range and Portrait Image Modification

When modifying the images captured and produced by the photogrammetry system great care must be taken to prevent distortions. All operations performed on one image should also be performed on its counterpart. In this project each step undertaken to modify the images primarily concerned making the images more distinct, of appropriate size and to remove background noise. Histogram equalizations were performed on both the depth map and the portrait image. Different methods of equalization were used only due to the difference in the types of images. The depth map is an eight bit grayscale image that is suitable for the OpenCV `equalizeHist()` utility where the three channel color portrait image requires color space conversion prior to using the same utility on the brightness channel. Other than that minor difference, both the depth map and the color portrait images underwent exactly the same modification steps in the same order.

The overall goal was to try and create images of nearly the same type and style as those in the 3DFRD collection. The range images in the 3DFRD collection are eight bit grayscale images while the portrait images are three channel color images. The depth map produced by `StereoSGBM()` is a grayscale image and the portrait image captured by the cameras are three channel color images. Once captured, the chief difference was in size and proportion. Cropping

the images out of the source and then scaling the images to match the 501 pixel width produced a set of images nearly the same proportions as those in the 3DFRD collection.

8.3. Facial Point Detection

Though it was not the focus of this study an attempt was made to detect the ten fiducial points in the imagery. The reason this portion was required is that there needed to be some means to compare the two imaging systems. It had been hoped to acquire the original Anthroface software used by Gupta in her study [17]. As the Anthroface software was unavailable, it would still be possible to compare the two imaging systems as long as the method for locating points within the images was the same for each collection. Therefore, my own method of fiducial point detection was developed.

A single pair of images, range and portrait, from the 3DFRD collection was used to develop the algorithm. Initially little success was met with the correct identification of any of the fiducial points. Yet, by rethinking and modifying how each of the points was detected progress ensued. Once all the points were detected individually, the individual point detection processes were combined and streamlined into a single function. The rest of the images contained in the 3DFRD set were processed through an automated point detection system. From the entire set of 1149 image pairs, portrait and range, only 488 image pairs successfully made it through the entire detection process. Even more disappointing is the fact only twenty-four of the 488 image pairs correctly identified all ten fiducial points. In the 3DFRD collection the most reliably detected fiducial point is *prn* followed by both left and right *al* and then *m'*. Using the sixteen modified image pairs captured with this photogrammetry system, three sets made it through the fiducial point detection process. Of those none of the three image sets detected the fiducial points properly. The reason for this is the point *prn*, the first point detected and upon which all other detections depend, is not correctly detected

9. Results

The system as is currently implemented with the current hardware is a proof of concept. The purpose of the project was designed to test the theory of the possibility to replace the original camera system used in the Gupta study with easily obtainable and inexpensive components currently available utilizing photogrammetry techniques [17]. Though direct empirical comparisons were not possible due to the resulting image detail being too low what has been demonstrated is that the basic theory is sound – it is possible to replace the expensive camera system with inexpensive off the shelf components. The resulting depth maps produced from the photogrammetry system are capable of showing facial details even with very low quality cameras. The depth maps clearly showed the nose, mouth, and eyes of the subject. Like the expensive camera system color portrait image were simultaneously captured with the images used to create the depth map. The cameras used in the photogrammetry system cost less than $100.00 USD which is far less than the specialty range camera system used to capture the 3DFRC image collection and less than inexpensive range imaging cameras. The software used to capture the images, create the depth maps and locate the fiducial points is open source eliminating the cost of expensive commercially available API's currently on the market.

10. Future Work

Three main areas of investigation remain: better cameras, modifying the OpenCV `StereoSGBM()` utility and using an inexpensive range imaging camera system. There are of course other avenues of investigation that may produce better overall depth maps from the current implementation but these are left for the reader to discover and investigate.

The replacement of the current cameras with other commercially available cameras having higher native resolutions is probably the first step to be taken. Currently in the market, there exist a myriad of cameras with higher native resolutions however they are typically more

expensive. Care must also be taken to ensure the cameras selected are either of the fixed focus type like the ones used in this study or have the capability to fix the focus either through hardware or software. Increasing the camera resolution is likely to yield more detailed and accurate depth maps from the rest of the system. Replacing the cameras will be the easiest means to repeat this study yet obtain better results.

The next area of investigation is to delve into the OpenCV `StereoSGBM()` utility. Currently this utility is optimized for segmentation producing a splotchy depth map. Modifying or rewriting the `StereoSGBM()` utility to produce gradient style depth map may produce smoother and possibly more detailed depth maps better matching those in the 3DFRD collection. Because OpenCV is open source software obtaining and modifying the `StereoSGBM()` code is possible and likely to produce better overall results.

Finally another area which requires investigation is the use of range imaging cameras. On the market today there are some range imaging camera systems. Some come paired with standard optical cameras and some do not. For example the Windows Kinect device has both a range imaging and optical camera in one package and the API for the device is freely available. Some investigation of this device showed it had the capability to acquire range images with single digit millimeter precision. A range image camera such as this should be capable of producing better depth maps than the ones obtained in this work yet still be reasonably priced.

11. Conclusion

What has been demonstrated in this paper is an alternative 3D image capturing system for use with facial recognition systems. The approach is practical, has limited cost and uses components and software which are readily available. Facial recognition systems are becoming more prevalent however most of these systems rely on older feature or holistic based methods of image analysis. Feature based methods require specific physical positioning of the subject to

be effective while holistic methods are strongly influenced by environment variables like lighting and background noise. Using 3D range images or depth maps avoids the problems associated with 2D facial recognition. The use of depth maps or range images provides additional information, a depth or distance from the imaging device, not present in 2D imagery. This additional information is valuable because it permits undistorted image rotation and resizing about three axes to bring the 3D image into a standardized size and position; the same translations may be applied to 2D imagery taken at the same time from the same camera. Standardized images can then be processed using feature based techniques and combined with the 3D information to produce a fully connected graph representation of an individual's face. Graph matching techniques can be employed to perform facial recognition of individuals with more precision and accuracy of 2D facial recognition methods.

This paper has presented a method of capturing 3D depth map images for use in facial recognition systems. The proposed 3D imaging system is adaptable to many different camera systems and is capable of producing detailed depth maps of the human face. The system is easy to configure and use. Practical applications will not require an operator after configuration and subject interaction is limited to looking towards the camera system.

Appendix 1

The project is written in C++ using Microsoft Visual Studio 2010 Ultimate edition. The OpenCV version used for the project was OpenCV 2.4.3 dated 11/2/2012. In the pocket there is a DVD containing the Visual Studio project, the 3DFRD image collection and the testing images captured with the described camera system. Also included are electronic copies of this paper and a presentation based on the paper's contents.

[1] C. Wilkinson, "Facial reconstruction – anatomical art or artistic anatomy?," *Journal of Anatomy,* vol. 216, no. 2, pp. 235-250, 2010.

[2] K. F. Kleinberg, "Facial anthropometry as an evidential tool in forensic image comparison," Glasgow Theses Service, Glasgow, 2008.

[3] M. D. Onis, "Reliability of anthropometric measurements in the WHO Multicentre Growth Reference Study," *Acta Paediatrica, Supplement,* vol. 95, pp. 38-46, 2 4 2006.

[4] S. Gupta, M. K. Karkey and A. C. Bovik, "Advances and Challenges in 3D and 2D+3D Human Face Recognition," *Pattern Recognition in Biology,* vol. 6, 2007.

[5] Online Etymology Dictionary, "bio-," Online Etymology Dictionary, 2013. [Online]. Available: http://www.etymonline.com/index.php?term=bio-. [Accessed 1 June 2013].

[6] MyEtymology.com, "Etymology of the Greek word metron (μέτρον)," MyEtymology.com, 2008. [Online]. Available: http://www.myetymology.com/greek/metron.html.. [Accessed 1 June 2013].

[7] A. K. Jain, A. Ross and S. Pankanti, "Biometrics: A Tool for Information Security," *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY,* vol. 1, no. 2, pp. 125-143, 2006.

[8] Centers for Disease Control and Prevention, "Workplace Safety & Health Topics," Centers for Disease Control and Prevention, 2013. [Online]. Available: http://www.cdc.gov/niosh/topics/anthropometry/. [Accessed 24 May 2013].

[9] A. A. Moenssens, Fingerprint Techniques, Philadelphia, PA: Chilton Book Compan, 1971.

[10] K. F. Kleinberg, P. Vanezis and A. M. Burton, "Failure of Anthropometry as a Facial Identification Technique Using High-Quality Photographs," *Journal of Forensic Science,* vol. 52, no. 4, pp. 779-783, 2007.

[11] R. S. S. Kramer, A. L. Jones, R. Ward and T. Mapp, "A Lack of Sexual Dimorphism in Width-to-Height Ratio in White European Faces Using 2D Photographs, 3D Scans, and Anthropometry," *PLoS ONE,* vol. 7, no. 8, pp. 1-6, 8 2012.

[12] J. P. Davis, T. Valentine and R. E. Davis, "Computer assisted photo-anthropometric analyses of full-face and profile facial images," *Forensic Science International (Online),* vol. 200, no. 1, pp. 165-176, 2010.

[13] Z. Fourie, J. Damstra, P. O. Gerrits and Y. Ren, "Evaluation of anthropometric accuracy and reliability using different three-dimensional scanning systems," *Forensic Science International (Online),* vol. 207, no. 1, pp. 127-134, 2007.

[14] Z. Zhuang, D. Landsittel, S. Benson, R. Roberge and R. Shaffer, "Facial Anthropometric Differences among Gender, Ethnicity, and Age. Groups," *Annals of Occupational Hygiene,* vol. 54, no. 4, pp. 391-402, 1 June 2010.

[15] R. Jafri and H. R. Arabnia, "A Survey of Face Recognition Techniques," *Journal of Information Processing Systems,* vol. 5, no. 2, pp. 41-68, 2009.

[16] R. Jafri and H. R. Arabnia, "A Survey of Face Recognition Techniques," *Journal of Information Processing Systems,* vol. 5, no. 2, pp. 41-68, 2009.

[17] S. Gupta, M. K. Karkey and A. C. Bovik, "Anthropometric 3D Face Recognition," *International Journal of Computer Vision,* vol. 90, no. 3, 2010.

[18] L. G. Farkas, Anthropometric Facial Proportions in Medicine, Springfield, IL.: Thomas Books, 1987.

VITA

Author: James E. Pearson, Jr.

Place of Birth: Wichita Falls, Texas

Undergraduate Schools Attended: Spokane Community College,
Eastern Washington University

Degrees Awarded:

Associates of Applied Science in Fire Science, 1997, Spokane Community College
Associates of Applied Science in Network Engineering, 1998, Spokane Community College
Bachelor of Science in Computer Information Systems, 2012, Eastern Washington University

Honors and Awards:

Graduate Assistantship, Computer Science Department, 2012-2014, Eastern Washington University
Graduated Magna Cum Laude, Eastern Washington University, 2012

Professional Experience:

Internship, Software Development Manager, Lingwee, Spokane Valley, 2010
COO and Project Manager, Tometa Software, Liberty Lake, Washington, 2006-2009