ASTRO3DO

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

MAY 2020

By

Michael J. Omori

Thesis Committee:

John Shepherd, Chairperson
Peter Sadowski
Kyungim Baek

Keywords: theses, dissertations, graduating, computer science, body composition, machine learning

To those in need,

Those with or destined for cancer,

and the group.

ACKNOWLEDGMENTS

I want to thank god for this.

ABSTRACT

There are several key components of health, which can be monitored through means of body composition and shape analysis. Metrics such as fat mass index and fat free mass index are simple features of the humany body that can aid our understanding in their overall health. How these are measured have been extensively studied throughout the years. Means such as DXA and bioimpedance can accurately measure body fat. While more recently, 3d imaging has proven to be able to match the performance of its predecessors whilst providing certain benefits such as lower cost. The Shepherd Lab has demonstrated such findings by showing similar results between DXA and 3d imaging. The project is ongoing and this thesisis is the culmination of my progress relating to 3d imaging methods and testing for applications in space. It provides an overview of my work and the starting point of much future effort.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Long duration space flights induce loss of muscle and bone mass, which in turn leads to greater risks of injury and death. To capture such changes, body composition measures are traditionally measured through Dual-energy X-ray absorptiometry (DXA); however, this is not as feasible in space. 3D optical (3DO) imaging has been demonstrated to be able to capture body composition as well as DXA, while also being hypothesized to be more easily deployable in space because of its low power consumption, higher mobility, and retainment of accuracy. We have accurate terrestrial models and the main goal of this study is to demonstrate the feasibility of our system to capture body composition in space.

Dual-energy x-ray absorptiometry (DXA) is commonly used to measure bone density by measuring the abosrption of soft tissue and bone using dual x rays [1]. The radiation used is considered lower than normal; however, they should still be minimized in their usage which is why one would not want to take these every single day for example both for thealth impacts and costs. Due to it's versatility, DXA can also be used for total body composition and fat content measurement.

Body composition comprises of fat, bone, water, and muscle percentage. In what is known as the four compartment model [6]:

$$\frac{1}{D_b} = \frac{w}{D_w} + \frac{f}{D_f} + \frac{p}{D_p} + \frac{m}{D_m} \tag{1.1}$$

Db: overall body density, w: proportion of water, f: proportion of fat, p:proportion of protein, m: proportion of mineral, Dw: density of water, Df: density of fat, Dp: density of protein, Dm: density of mineral

Bio-electrical impedance (BIA) is used to measure volumnes, shape, or tissue electrical properties using a the general formula [13]:

$$Z = \frac{\rho L}{A} \tag{1.2}$$

ρ is the resistance which is a metric of the tissue. L is the distance between the electrodes placed on the body. A is the area. Z is the impedance. By placing an electrode on the person's right hand and foot and sending a current through the body one measures the voltage and current with respect to time on each electrode and can determine body water and subsequently body fat. Fat and bone have higher resistance than fluid and electrolytes and thus the electrons travel differently through these mediums.

In regards to the sensors there are two main types, stereo cameras and time of flight. Along with ones such as lidar. The ideal imaging equipment roots its foundation on the attunement to the capture and reconstruction of precise depth measurements. The sensor so called, to include

various imaging modalities such as time of flight and cameras, is required to be lightweight because it needs to be mounted. It must meet basic requirements including frame rate, resolution, precision of acquisition, field of view. The ideal specifications surround image quality and noise. Basic hardware feature that directly affect image quality as such are pixel size and depth reconstruction algorithms. For depth, there are a few important things. The Z-accuracy, which is the depth data accuracy. The fill rate which evaluates the percentage of the depth coverage of the image. The RMS error which evaluates the spatial noise or spatial depth uniformity. And the temporal noise which evaluates the temporal uniformity over sequential frames. Other traits between sensors are mostly comparable and company provided specs are reasonably reliable.

ToF is super-fast, but resolution is low (width x height), accuracy of depth is very good, but they are more susceptible to light. It can be considered a type of lidar, scanner less. A cheap, lidar sensor will be released by Intel soon. Time of flight uses the phase shift of amplitude modulated wave to measure the distance. Stereo is taking two pictures of the scene at different positions. The depth depends on the reconstruction algorithm. Intel camera also had an optional infra-red ToF is super-fast, but resolution is low (width x height), accuracy of depth is very good, but they are more susceptible to light. It can be considered a type of lidar, scanner less. A cheap, lidar sensor will be released by Intel soon. Time of flight uses the phase shift of amplitude modulated wave to measure the distance. Stereo is taking two pictures of the scene at different positions. The depth depends on the reconstruction algorithm. One such way of calculating the depth from time of flight based on a simplified ideal model.

$$c = \lambda * f \tag{1.3}$$

Where c is the speed of light, λ is the wavelength and f is the frequency

$$\varphi = ArcTan(\frac{A_1 - A_3}{A_2 - A_4}) \tag{1.4}$$

Where A represents the amplitudes of the sent and received signals. φ gives us the phase shift.

$$r = \frac{\pi * d * M}{n} \tag{1.5}$$

$$d = \frac{c * \varphi}{4\pi * f} \tag{1.6}$$

The resulting depth value.

To obtain the values that we want we need good input values. Then it's a matter of creating a function that maps these input values to these output values in which an error function is minimized. This is the general setup of learning experiments. Although a machine can automatically do parts of this pipeline. There are certain parts which we consider them incapable of doing for us as of

now. The primary such first example is determining the inputs. What format should the inputs be in? Ideally we could take in all of the input of the universe but this is not the case. We must limit ourselves to some relatively finite input such as images. So these images contain values from sensors. And these sensors can be your normal camera or time of flight or lidar or whatever. Other input may include descriptive information such as categorical variables like demographics: race, gender, age, etc. There are different ways to use and combine these but it is the effort of the researchers to decide on the potential use of these as information comes with a price. In regards to stereovision, it takes in two images as inputs. And matches the features in the two images. With this matching of features, one can calculate the distance from the sensor to the features. This may be primarily considered a type of feature engineering as it is not recorded input but rather engineered on top of the input. Many groups have deemed this feature of depth as supremely useful for various applications involving 3 dimensional volumes.

Thus this work concentrates on utilizing depth data from sensors for 3d reconstruction. It is possible to go directly from the depth data to body composition without the need for feature engineering such as mesh reconstruction, caesar point placement, templating, principal component analysis, and linear regression to get to body composition and shape. One could theoretically go from depth data to body composition and shape. As such this is briefly demonstrated in this work.

This thesis is composed of several main parts. The first is the comparison of the sensors for data collection. The second is the modeling of the data. The third is data simulation, augmentation, and validation through means of distribution approximation methods such as microgravity environments. Finally, the preparation for data acquisition in one such special environments known as parabolic flights.

Here I describe a few file formats that are in use for further clarity of this document. First is the obj file. Second is the ply file. Third is the pp file. Fourth is image data. Fifth is depth data. Starting with image data which most people are familiar with. A color image will often have 3 channels one for red, one for green, and one for blue. An image that is 720 x 480 for example is 720 pixels wide and 480 pixels high. An image will contain 720 x 480 x 3 values. Depending on the acquisition format, there may be a limit to the lower and upper bounds of these values. Commonly it is a byte, 8 bits, so from 0-255. Depth data on the other hand only has one channel usually. And the upper bound of the value is usually limited by the data type used such as a 64 bit structure. It will usually represent distance from the sensor to that point in meters. These are the primary image data. There are several different mesh formats used with the two main ones in this project being the obj and the ply file.

1.0.1 mesh file formats

Beginning with the obj file, this file is a human readable format. It contains information for the position of each vertex, the UV position of texture coordinates, vertex normal, and faces. Vertices are listed in counter-clockwise order. A vertex is represented as v x y z. Vertex normals are optional

because faces define them. They are represented like vn x y z. Optionally, one can specify textures with vt u v. Generally a face will be defined as f v1 v2 v3. The one main requirement for an obj is the vertices. When opening this in a viewer, one can see the points. Many computer graphics, 3d printing and game rendering engines also require the face information to connect the dots along with texture information for how the faces look. In this project, only vertices and face information is used. The ply file is similar to the obj file. There is also the option for encoding and compression

Figure 1.1: Obj File Example

```
# wavefront obj fit3d

v 37.8092 306.632 1033
v 36.2096 304.905 1032.97
v 36.2402 306.649 1030.23
v 36.8483 302.549 1042.33
v 36.0953 302.107 1042.32
v 36.1129 302.559 1040.87
v 39.5937 306.6 1037.71
v 38.154 304.777 1037.68
v 38.1963 306.625 1034.03
v 38.7433 304.557 1042.37
v 38.0954 303.809 1042.36
v 38.1402 304.584 1038.77
v 40.0105 306.563 1042.41
v 37.9286 304.592 1037.68
v 36.1991 304.618 1033.83
```

of this file; otherwise the file is human readable like the obj file. A plus to this format over the obj file is the ease of adding color information. In general the file format is similar. There is the list of vertices and faces and when one wants to add color to a face one can specify a tuple such as (r, g, b) to represent the red green blue values for that face.

Figure 1.2: Ply File Example

```
ply
format binary_little_endian 1.0
comment VCGLIB generated
element vertex 290098
property float x
property float y
property float z
element face 580161
property list uchar int vertex_indices
end_header
L□□\A6>□□\88s?1\AC\BC=\AC□□\A6>□□\88s?□
Ol?y\BE\B5==\F4\A5>
```

Figure 1.3: PP File Example

```
<PickedPoints>
 <DocumentData>
  <DateTime date="2020-03-30" time="14:21:37" />
  <User name="AI" />
  <DataFileName name="001AI.ply" />
  <templateName name="new Template" />
 </DocumentData>
 <point active="1" name="m01Sellion" x="-0.03402664512395859" y="0.5200187563896179" z="0.25226500630378723" />
 <point active="1" name="m02RtInfraorbitale" x="0.002578672021627426" y="0.4228420853614807" z="0.2134241759777069" />
 <point active="1" name="m03LtInfraorbitale" x="0.0727488100528717" y="0.5088114142417908" z="0.24195057153701785" />
 <point active="1" name="m04Supramenton" x="0.032781168818473816" y="0.4124462902545929" z="0.27130043506622314" />
 <point active="1" name="m05RtTragion" x="-0.08633866906166078" y="0.5837581157684326" z="0.262582927942276" />
 <point active="1" name="m06RtGonion" x="-0.06718064844608307" y="0.5779197216033936" z="0.19120559096336365" />
 <point active="1" name="m07LtTragion" x="0.03783474862575531" y="0.4185194969177246" z="0.10840305685997008" />
 <point active="1" name="m08LtGonion" x="0.13412122428417206" y="0.2898013591766357" z="0.15875159204006195" />
```

CHAPTER 2
PREVIOUS WORK

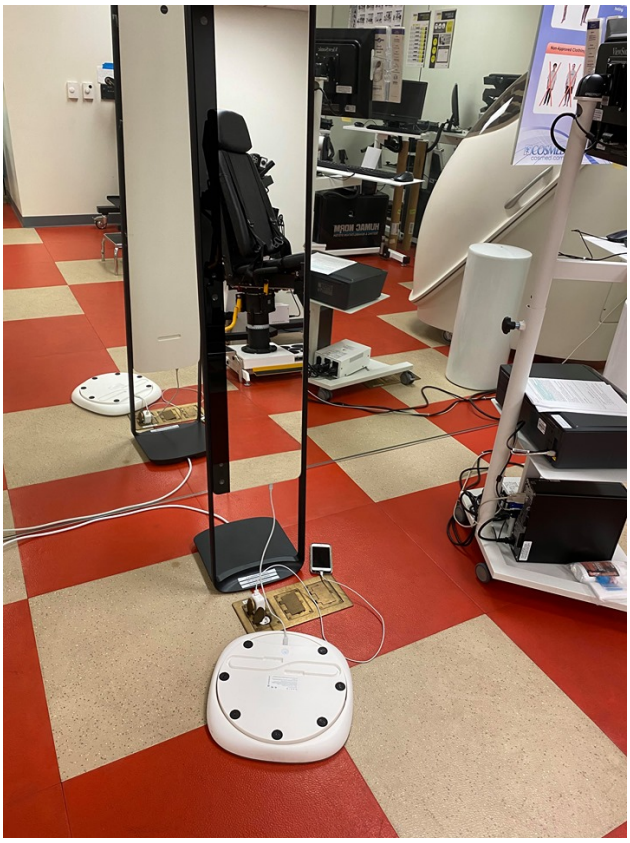
Previous work on this included a variety of 3d scanning machines.

Figure 2.1: Fit3d scanner

2.1 3d optical scanners

These scanners are all good and use a variety of kinect sensors and intel sensors. They are good but rather inflexible in serving the purpose of imaging in space. As such, I built my own system for this. The software for these systems is not open source and depends heavily on the precise geometry of these systems

Figure 2.2: Naked Labs Scanner



2.2 depth imaging

Next I review one of the most fundamental work on 3d imaging [12]. KinectFusion: Real-Time Dense Surface Mapping and Tracking was a seminal work in which Microsoft developed a time of flight sensor and developed algorithms for real time reconstruction of a scene. Their method comprises of 4 main parts. Part 1 is the surface measurement, which includees the raw depth measurements along with a dense vertex map and normal pyramid map being generated. Part 2 is the sensor pose estimation which is done by predicting the what the surface should be like from the previous data and using multi-scale iterative closest points to the current data. Part 3 is updating the reconstruction by fusing the current measurement with the scene model which is maintained with a truncated signed distance function. The final part is predicting the surface from part 3 which feeds back into part 2. Further information and review of state of the art methods is given in the paper:Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era [7].

Figure 2.3: Styku scanner

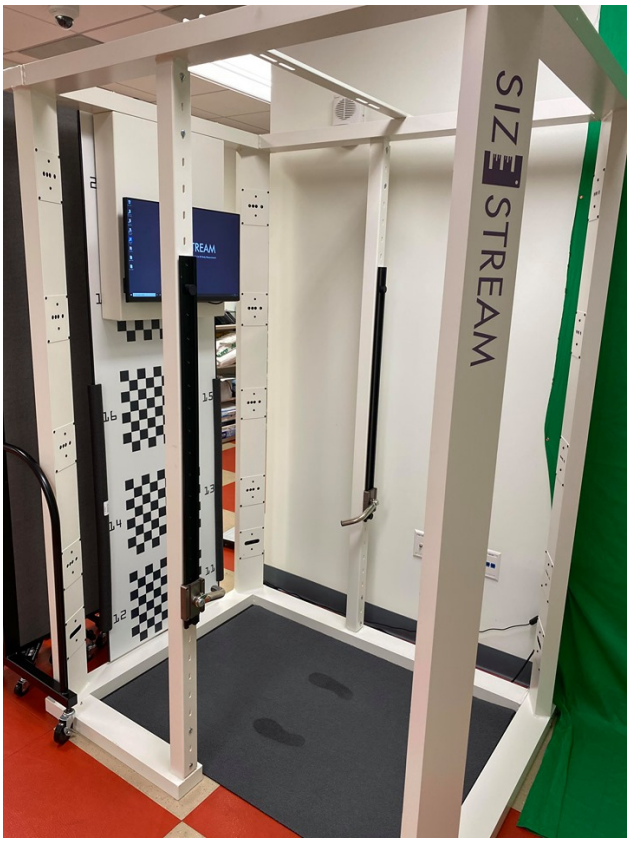
2.3 3d reconstruction

In 2005, Brett Allen wrote his phd thesis on learning body shape models from real-world data [2]. Both sensor technology and algorithms have come a long way since then.

2.4 3d neural networks

One of the seminal deep learning papers on 3d input was PointNet [19]. It was a method used for classification and part/semantic segmentation. A couple years later, dynamic graph convolutional neural network took this further by using dynamic updates to the graph [22]. While these methods are general, there are also more specific methods for reconstruction. A systematic approach for constructing 3d human models from 2d images [23]. The Planck Institute published results on obtaining a mesh from a single 2d image using a combination of their previous convential method and deep learning [15].The method uses a CNN followed by their previous work SMPL and these use a self improving loop. This was tried briefly in this project. Deformable Shape Completion with Graph Convolutional Autoencoders was considered as the paper shows results on taking in an

Figure 2.4: Sizestream scanner



essentially partial mesh and completing it [17]. This was put on hold though because the code is not open source and would have to be re-implemented.

2.5 Sensor testing

Several papers have done sensor testing with depth cameras. Overall methodology was studied on these papers [21] [14] [16] [20].

CHAPTER 3
METHODS

There were several things I worked on. The first is a review of different sensor technologies and subsubsequent testing of the top ones. Next was the development for the Zero G Parabolic Flight which are scheduled for the summer. Third was simulating microgravity and testing them. Fourth was seeing the impact of pose on the resulting steps and final measurements.

I did a review of the sensor technologies out there excluding the commercial systems mentioned in previous works. I looked up their specs such as resolution, depth accuracy, and frame rate. I then benchmarked them on depth accuracy, temporal noise, spatial noise, and fill rate. These benchmarks were also done at varying distances. To do so, I made a simple setup. I used an imaging chart and a laser measure. I used 10 frames for each test. The setup is shown here

Machine Learning and Deep Learning work very well in many domains.

3.1 Graph Neural Network

One such deep learning method that is used for 3d data inputs are graph neural networks. The one I used specifically is called dynamic graph convolutional neural network (dgcnn) [22]. It consists of n vertices, each of which are tuple of 3 numbers. In order to apply the convolution operator, one must define an edge. With 2d images, an edge is any pixel that is adjacent to another pixel within the kernel length. For a graph, different distance metrics can be used. Dgcnn uses pairwise euclidean distances. The k nearest points are used for the subsequent convolution. Next comes the convolution kernel, which the authors define as

$$E_ij = ReLU(\theta_i * (x_j - x_i) + \theta_j * x_i) \tag{3.1}$$

Then comes max pooling for all such node j in the neighboring set of node i:

$$F_ij = Max(E_{ij}) \tag{3.2}$$

Before the point cloud goes into this architecture, they apply a matrix transformation using the coordinates and the differences with their neighbors. They prove certain desirable properties such as permutation invariance and translation invariance. And achieve state of the art results on classification and segmentation in June 2019. While they used their architecture for classification and segmentation, I convert it into regression for use in an automated caesar point placement. In Caesar, 75 landmarks are manually picked which allows a standardized template of 60k nodes to fit to the original mesh that has several hundred thousand points. Usually this takes several minutes for an expert user and can take up to an hour for a beginner. A landmark is defined as an (x,

y, z) coordinate and there are 225 such numbers that are needed to be predicted in a regression case. The other method is segmentation, which is reduced to classification of each point. In which there will be several hundred thousand points which will each need to be classified into 75 classes. Resulting in many predictions and some of these predictions may not even be exactly precise as the landmarks may not coincide directly with a point. It is hypothesized that training maybe easier in the beginning; however, in the long term one would expect a degradation in results because their is much noise. This noise comes from the fact that many points don't belong, and only 75 are really needed. With regression, one can predict the exact point and get to perfect performance. But the beginning of training may be more difficult as the predictions are totally random and the errors will be large. The other method considered was taking in the entire mesh including face information as done by MeshCNN [8]. It is unclear however as to the utility of face information compared to nearest neighbor information from dgcnn. As the previous information is feature extracted through non-deep learning methods in a rather arbitrary sense while neighbor information is computed by the network.

Thus for automated body shape analysis a deep learning model was briefly reoriented from dgcnn. Briefly, a deep learning model was created. 5000 scans from pca to body composition, trained on this data. PackageID from fit3d, compiled database in o drive, filename to subjectID and technology sex, height, weight, bmi, ethnicity were used. Trained to dxa values. Table 1 variables in bennets paper tied to filenames for all the scanners ask Ian were used for joins. Dxa and fit3d has 2 data points for each person, 2 values were averaged. I got a basic catboost model with the non-image features. I run a graph neural network using 1024 vertices, scaled to a unit sphere and augmented. I saved .obj files into .h5 files. .h5 file needs both data and label attributes. Numpy arrays needed to be float 32, and labels as int 64. The output is regression with multiple outputs at once using the same architecture. MSE loss caused exploding gradient thus the S was removed. File named matched to subject id and the output values. The data is mostly in one excel workbook, with different data coming from about 6 different sheets. Subset of features used, like the 2019 shape up pca paper.

3.2 3d reconstruction

Took rgb and depth images of Jon too. Video of Michael and En taken with Intel d435 camera at 15-degree intervals on an inversion table. Along with iphone 11. Used the realsense sdk. Camera is mounted to the end of the inversion table. Set up spin which uses input image, passes it through a CNN, then use SMPLify to get a basic mesh file of Michael.

3.3 Sensor Comparison

3.3.1 specs

Figure 3.1: Sensor Specs

Model	Depth FOV	Resolution	Frame Rate
Azure Kinect	120 x 120	1024 x 1024	15
D435	85.2 x 58	1280 x 720	30
Zed	90 x 60	2208 x 1242	15
Astra Stereo S	67.9 x 45.3	640 x 480	30
XtionPro	58 x 45	640 x 480	30
RC_visard 160	61 x 48	1280 x 960	8
SceneScan	NA	1856 x 1856	10

I review the spec sheets for these sensors. While the above are metrics which people haven't done before on those sensors. I review depth formats such as resolution, frame rate, operating range, the color sensor corresponding features, including field of view, and where information is available online, I include the aperture, cost, sensor pixel size, software development kits (sdk). some specs are publicly available such as these and more in depth online. It was important to us to do custom testing specific to our application in addition.

Next to calculate the approximate distance based on the field of view. We can do the following:

$$Tan(\theta) = \frac{H}{2 * D} \tag{3.3}$$

Where H is the horizontal distance of capture and D is the depth distance for capture.

3.3.2 testing setup

We bought and have an art easel to hold the chart. A tripod for the camera. Along with 480 LED lights 3200-5600K CRI 96+. A custom chart was bought shown in the appendix. Lights are to be positioned behind the camera at 45 degrees, with the test chart in front, figure in appendix. Below is the testing setup. I

3.3.3 Custom testing metrics

In order to choose the best sensor one could use intermediate metrics that define how good that sensor or one could use all of those sensors in the overall imaging setup and analysis and see which one works the best. The metrics I used are similar to the techincal report [11].Ideally, both should be done. I started with the first. The metrics thus defined are Z-accuracy, Fill Rate, spatial noise, and temporal noise. The Z-accuracy says how accurate the depth sensor is in relation to a ground

Figure 3.2: Sensor Testing Setup

truth(GT). We average the differences in depth sensor value and ground truth. Each metric is for a given single frame and was then average across 10 frames.

$$z - accuracy = \frac{1}{n}\sum_{p}(Image - GT) \forall p \in box \tag{3.4}$$

The fill rate relates to how many of the depth pixels are valid. This is useful as some sensors have high accuracy but low fill rate. This is the percentage of pixels that are non-zero.

$$Fill - rate = \frac{\sum_p [I_p >= 0]}{|p|} \forall p \in box \tag{3.5}$$

Spatial noise I define as the standard deviation divided by the mean distance. The RMS error or spatial noise is useful to determine the x-y noise from a plane that is approximately equidistant from the imaging sensor.

$$noise = \sqrt{\frac{1}{n-1}\sum_{p}(I_p - \bar{I_p})^2} \forall p \in box \tag{3.6}$$

Last comes the temporal noise. Which is esentially the same as spatial noise except we take it across frames of the same pixel location.

$$temporal - noise = \sum_{p}\sqrt{\frac{1}{n-1}\sum_{f=1}^{10}(I_{pf} - \bar{I_p})^2} \forall p \in box \tag{3.7}$$

These 4 metrics are what I test. In addition to that, I compare things such as frame rate, field of view, weight, dimensions, resolution, sdks, minimum z distance and maximum z distance.

3.4 Categorical Extreme Gradient Boosting

I used Catboost [5] in order to model body composition using only demographic information and was able to achieve very good performance even better than previously published [18] .

3.5 Parabolic Flight Setup

The original plan was to have parabolic flights during the summer of 2020 but when we started looking into dates for zero g flights, they had regular flights then but only had research flights during the spring and fall. Thus we chose to attempt to get ready for the spring. It was a bit rushed and we almost made it, ultimately though, zero g was not able to work through some deals with other companies and coronavirus also started ramping up at that time, so the flight was postponed. To take a 3d imaging system aboard such a flight required certain things that other people have not taken into consideration before when imaging humans for body composition. Some systems like the naked labs scanner have a mirror which encases the sensors. This is not necessary in zero g as one does not want any object to crack the mirror and the mirror is not needed for imaging. Other scanners such as the sizestream are extremely heavy and bulky and use many sensors. The height of this is larger than 60 inches which is the maximum height an object can be when brought onto the zero g flights. Thus I built my own imaging system. After testing the sensors, I selected the best one. Then tested different imaging parameters for the software such as frame rate, distance from the sensor, etc. The figure below is what the general setup looks like.

The specific maneuver of the aircraft is ascending at 45° followed by flattening the plane out followed by 45° nose low fall. The G-ForceOne contains 36 seats and is 60 feet long and 10 feet wide. Due to safety concerns, the maximum height of any equipment is 60 inches. The aircraft contains attach points that are 20" x 20" (+/-1/16"). Bolt spacers are used with 3/8-24UNF threaded holes along with AN6 steel bolts for securment of equipment. Other requirements included the material be at least 1/2" thick and 24" x 24" minimum. As such, a wood board was used that is 24" x 24". This was chosen over metal because of the lessening of vibration and for ease of drilling. Strap for carrying and anchoring are used including: CGU-1/B, S0203-0338-B, and 104410P. The power type aboard to be used is 115 volts ac, 60HZ, three phase. The aircraft lighting is at 5600 kelvin and are LEDs, suitable for general photography and this type of imaging. On-board tools are properly labeled and to be stored in a toolbox. Before flight, a test readiness review is required. The specific parts that were used are as follows 1 inch diameter steel flange for attachment of the pole to the platform. The pole used is metal for the parabolic flight and pvc for adjustable lengths on ground. Screws were used to attached the flange to the platform. 4 holes were drilled into the platform

Figure 3.3: Astro 3d imaging setup

for attachment to the plane floor using AN6 steel bolts. A tight plastic cap was attached the top opening of the pole to soften any physical impact. Threaded clamps were used to attach the sensors to the pole at adjustable positions.

3.6 Creating the ideal imaging setup on plane vs on ground

There are a few differences with the ideal imaging setup on plane vs on ground. One such being timing. As each parabolic arc is limited to only 8-15 seconds of microgravity. So we only could shoot for 8 seconds at max to be safe. While on the ground, time is only a concern due to funding for the subject's time and their availability. Thus when choosing the best parameters on ground. I tested various setups. This included sensor type, frame rate, sensor height, rotation speed, laser power, minimizing laser interference. All included in the table below.

3.7 body imaging on babies

Body imaging on babies is difficult because babies have trouble standing on their own. So typically they are imaged lying down. A result of this is the backside can not get imaged. The method we tried using to get a mesh of babies was smil. After getting it set up we got some preliminary results on fake babies that looked good. The original plan was to see how this could be used for this project too. As there are some cases where the entire body may not be able to be imgaed. This includes on an inversion table where the back is once again obscured from view of the sensors regardless of positioning. Another is the timing issue on the parabolic flights in which one could stitch together an incomplete view of the person if needed. With regards to the specifics of SMIL. We have rgb images of a baby. Depth images are in 16 and 24 bit, along with a segmented mask. The test images all have values between 791 and 957. While my images were between 0 and 255, so I added 791 to all the values. Since it was doing a cutoff at 65k, and treating the pixel values of 0 as missing. The table segmentation needs to be manually adjusted however as this affected the energy terms and subsequent mesh generation.

3.8 3d mesh modeling of humans

A few methods were tried out such as recent deep learning methods that take in an rgb image and produce a mesh Also the classic kinect fusion code was tried out paper. Finally a high level library of the kinect fusion code was used recfusion.

3.9 microgravity simulation

Below is one such example of microgravity simulation.

3.9.1 underwater

Some other ways besides parabolic flight testing to simulate microgravity include gravity boots, inversion tables, and underwater imaging. The protocols were developed for these but scanning of participants was halted due to coronavirus. Planning for underwater imaging was briefly explored. A 2016 paper by Disney research [4] used intel realsense cameras and a custom capture device. They demonstrated how calibration is different underwater along with the refraction model required. With such results, one could go on to perform 3d reconstruction from this mapping. Some things that have not been before include a complete scan of human bodies underwater which may require a scanning protocol and/or multiple sensors. These are a work in progress for this project.

3.9.2 inversion table

Different angles from -90 to about -45 degrees. Let's say capture at intervals x. Where x is 15 degrees. Each subject will be different on the amount of time they can handle. Let's say we want to capture 135 / x angles for y amount of minutes. We can say y is 5 minutes, where 10 minutes is extreme, while some may only be able to handle a few minutes. How much does it take for a subject to go back to baseline? Let's say y +- e minutes, where e is 0. So to capture one angle will take 10 minutes if it's 15 degrees, that's 9 takes. Which is about 90 minutes. Plus some time to get in and out. We can assume about 2 hours. So 30 athletes will take 60 hours. There are of course different considerations for the angle interval x, and time interval y. The main thing depends on time constraints and subject health. The protocol could be reversed too, to see if the order of inversion has an effect. The inversion table itself is 58" x 29" x 61". Currently one camera attached. Multiple cameras could be mounted.

3.10 Mesh similarity for ranking mesh quality

Various ways of ranking mesh quality. One such is using a human to rate the quality. With enough supervision, one could create a deep learning model to output mesh quality. This is quite time consuming. Another is to run each mesh through different pipelines and see how it performs on the end result for getting values such as body composition. This methodology is also good but our current methodology is not entirely automated. The method may not give the best accuracy but is completely automated in comparison to the previous approaches. There are many ways of defining similarity. One such way with text is to first perform some type of learning such as next word prediction in word2vec and then take the first feature set known as embeddings and use them as a dense representation. Upon which people have taken measures such as cosine angle similarity which can work better than euclidean distance sometimes in high dimesnional data. More advanced methods include using a neural network such as a siamese neural network which runs two pieces of text through the same network, obtains a final representation, and outputs a similarity score using a kernel function or something as previously mentioned like cosine similarity [3]. The need for this in text comes from the fact that there is no precise highly correlated numerical representation for text, rather it is highly context dependent. With images, as simple way is to take a simple l1 or l2 distance at each pixel and channel. Neural networks of course can also be used again. Other methods may include using neighborhood information which can reduce the impact of outliers such as through a gaussian distribution summation. Meshes are similar in the sense they are numerically represented like images. But they are different because there is no direct correlation between vertices like that of images which are aligned by x-y position. In this case, once again, one can use a neural network and get some intermediate representation and get the similarity from there. Due to time limitations, I perform a simple method for mesh similarity. This method is used

in often in the fusion of meshes and/or alighment of meshes. It is known as iterative closest point (icp).

One more well known distance metric worth mentioning is the Hausdorff distance which is a point estimate [10]:

$$d_h(X,Y) = maxsup(inf(d(x,y), sup(inf(d(x,y)), x \in X, y \in Y \tag{3.8}$$

Where sup is supremememum and inf is infimum. Iterative closest points is as follows:

$$E(R,t) = \frac{1}{N_p}\sum_{i=1}^{N_p} ||x_i - Rp_i - t||^2 \tag{3.9}$$

x_i and p_i are corresponding points. While in general, there are many performance things done for this algorithm. With t representing translation and R representing rotation. We do not have to consider these because our lack of dynamacy. Thus our problem is simplified. On top of this, I perform quadric edge collapse reduction to reduce the seach space as a form of sampling [9].

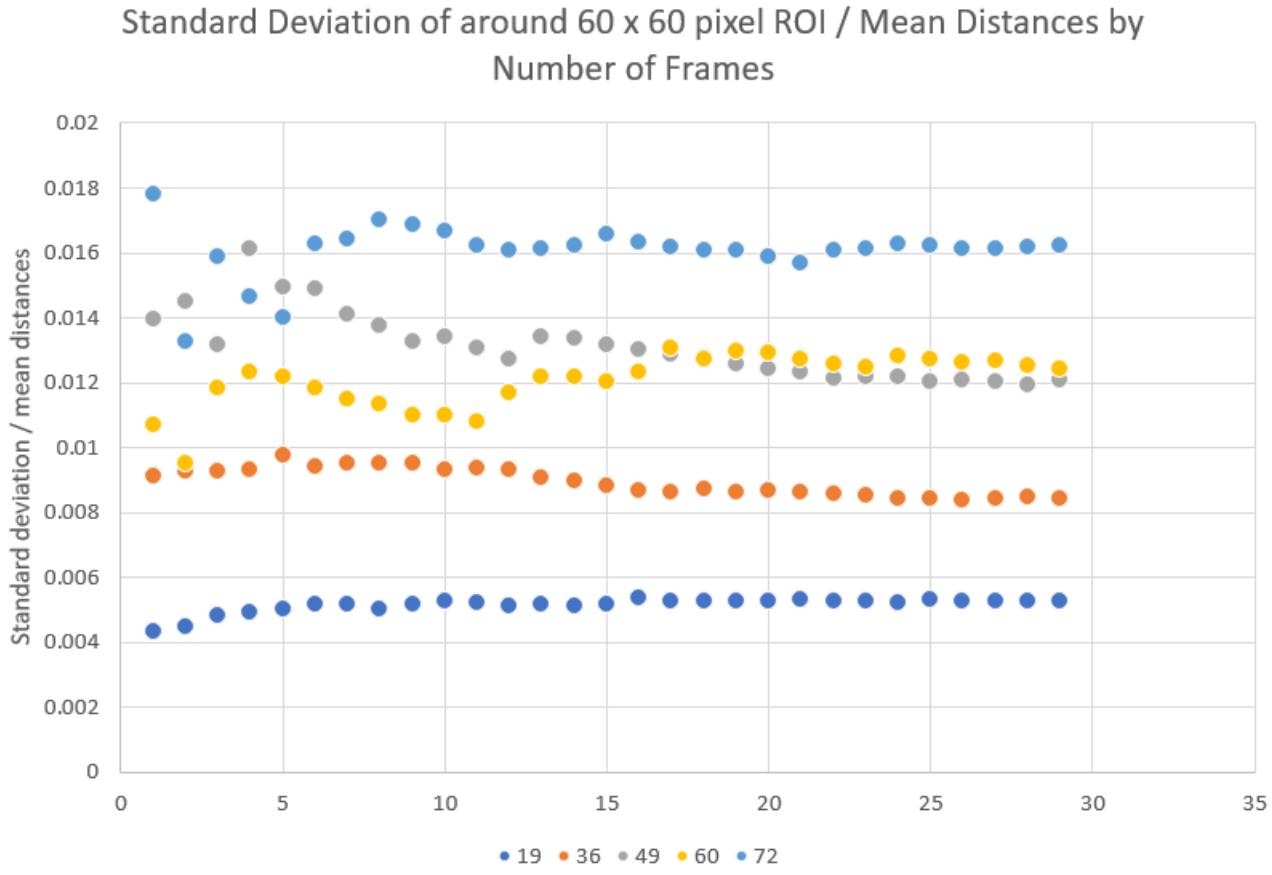
Figure 3.4: Inversion Table

CHAPTER 4
RESULTS

4.1 sensor testing results

The below figure shows the spatial resolution per frame of the d435. This was done to test if the spatial noise changes with the number of frames and the distance. The number of frames does not appear to have an impact; however, the spatial noise is increasing relative to the distance.

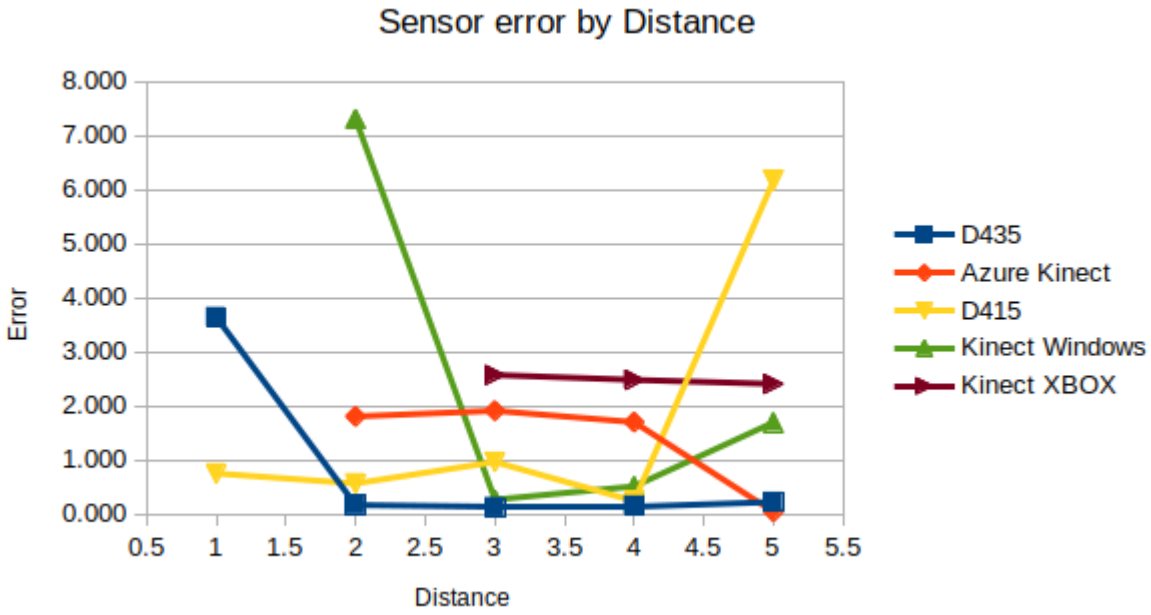
Figure 4.1: Spatial resolution per frame of d435



Here is the chart for sensor accuracy. The average errors by sensor are: 0.864 for the D435, 1.366 for the Azure Kinect, 1.739 for the D415, 2.446 for the Kinect Windows, and 2.489 for the Kinect XBOX.

Here is the chart for temporal noise. Whereas the D435 scored the best, the D415 scored the best in this category. This is partly explained by the d435 being like the d415 except having its

Figure 4.2: Sensor accuracy by distance



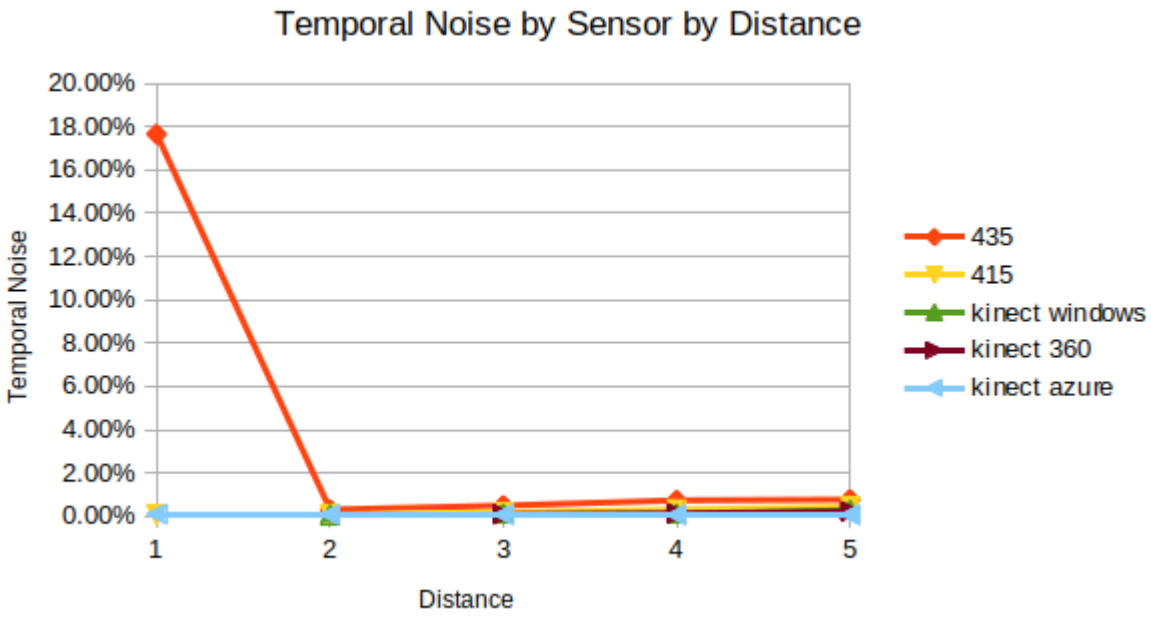
sensors more spread out. The winner in this category is the Kinect Azure with an average temporal noise of 0.07 percent, then comes the kinect for windows at 0.14, kinect 360 at 0.15, d415 at 0.22, and d435 and 3.99.

Here is the chart for fill rate. At first this metric may not seem that important but once you look at the actual depth images you can really see the difference between the sensors. If we exclude the kinect windows because it can't take data at 1 feet away and the kinect 360 which can't take data closer than 2 feet. The best sensors are d435 at 89.86 percent, d415 at 74.81, then Azure at 70.00. The Azure Kinect makes a very interesting design choice as the developers only keep pixels that the sensor is very confident in to maintain better accuracy. While the Kinect Windows and Kinect 360 trade this off by showing more valid pixels but have a reduced accuracy. With this judgement, it is even unclear whether the Azure kinect is even a better sensor in terms of specs as if one applied this same reduction in uncertain pixels in the kinect windows and kinect 360, those sensors may achieve comparable results.

4.1.1 sensor specs

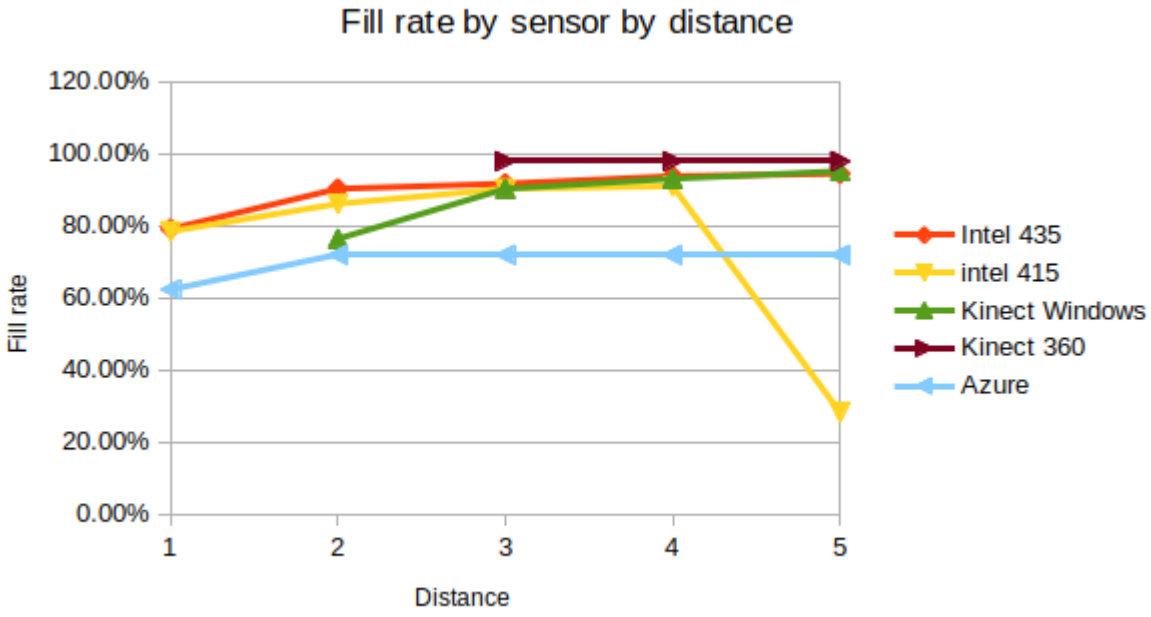
The d435 is capable of reaching up to 100 frames per second which for this particular imaging application is not that useful, but an application in like drones for example, it could be more useful. One thing to note is that 100 frames per second is a lot especailly when combined with multiple sensors and a high quality data connection and cpu is needed to transfer the data fast

Figure 4.3: Sensor temporal noise by distance



enough. One can refer to the specs for more details and I just outline some of the main findings. The strong point of the azure kinect is it's large field of view in certain settings of 120° by 120°. While the d435 has a max depth resolution of 1280 x 720. The depth field of view for the d435 is 85.2° horizontal and 58° vertical. At first it was unclear the advantage this would bring but after testing, I realized the field of view is very useful in allowing the sensor to get closer to the target and allow the target to remain in the field of the view of the sensor. As getting as close as possible is very helpful. The azure kinect is meant for closer range solution as the operating range varies between modes but the low end is 0.25m and the high end is 5.46m. While the d435 boasts a max range of 15m, it can only get 1m close. So just comparing this, the azure kinect is preferred for this use case. Next I do a quick review of the color sensor. When I first spec'd these out, I didn't know the impact but after working with sensors and methods, the current methods primarily rely on the depth sensor information. It is only in the case of calibration in which the sensor configuration is changing that the color sensor is used to locate the physical calibration boards using color features. But when building up the mesh of the scene, is the depth sensor that is being used and the color sensor has no use currently, although it could potentially as it provides information unavailable in the depth sensor.

Figure 4.4: Fill rate of sensor by distance



4.2 SMIL

Smil sample output:

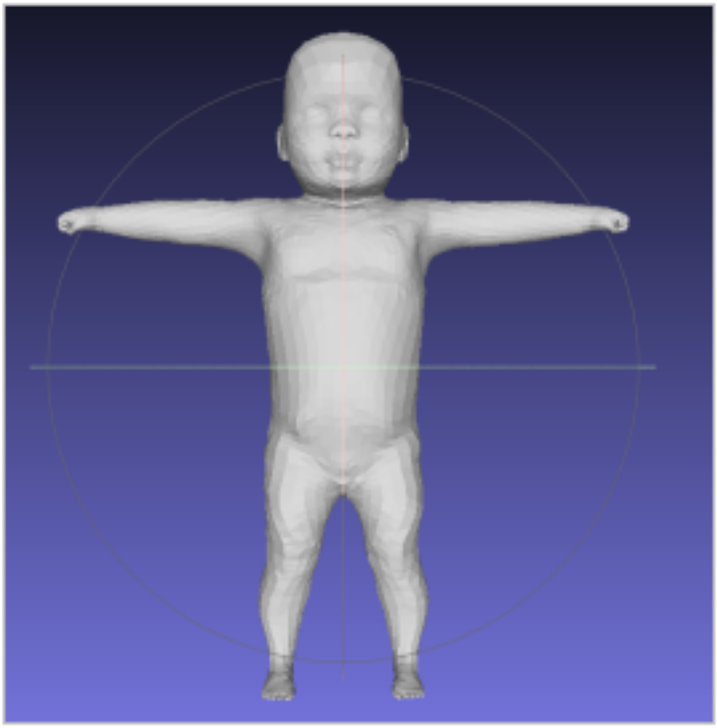
4.3 automated anthropometry results

We can see the training and validation errors quickly go down after about 10 epochs possibly from simply predicting the mean value of the output, then continues to go down very slowly. There is no sign of overfitting as the validation loss and training loss are similar to each other. Because only 1024 points are used, this greatly inhibits the capability of the netowrk to learn but was used for quick demonstration purposes. The number does not give me much intuition about the quality of the points. There are a couple ways to do this. One is to see how the final body shape composition results are and the other is to visualize the data. I decide to visualize the data because it could be helpful as a guide for humans if the points aren't place accurately by the algorithm I decide to visualize the data because it could be helpful as a guide for humans if the points aren't place accurately by the algorithm.

4.4 gradient boosting body composition results

First missing data was examined. Due to the low count, it was removed row-wise.

Figure 4.5: SMIL sample output

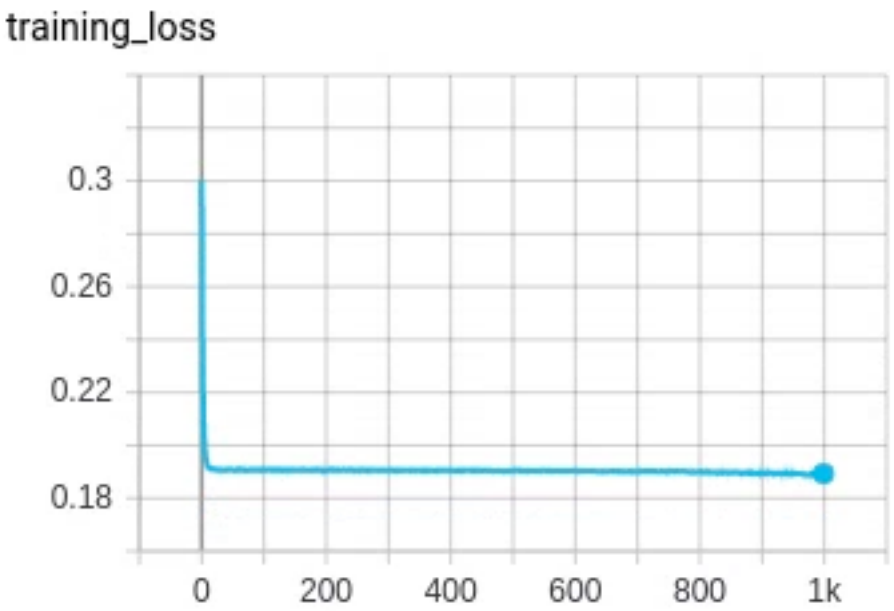


Distributions of the targets were created.

4.5 graph neural network body composition results

5000 scans, pca to body composition, training on this data -packageID from fit3d, compiled database in o drive, filename to subjectID and technology sex, height, weight, bmi, ethnicity -train to dxa values table 1 variables in bennets paper tied to filenames for all the scanners ask Ian. dxa and fit3d has 2 data points for each person, 2 values were averaged. Got a basic catboost model with the non-image features. can run graph neural network using 1024 vertices, scaled to unit sphere and augmented. Saved .obj files into .h5 files. .h5 file needs both data and label. numpy arrays float 32, and label as int 64. Output is regression, multiple outputs at once. MSE loss caused exploding gradient. file named matched to subject id and the output values. The data is mostly in one excel workbook, with different data coming from about 6 different sheets Subset of features used, like the 2019 shape up pca paper. The input data was combined. Some basic data exploration was done like counting of missing data. Distributions of the target variables were plotted, along with the features. Using catboost, after 999 iterations, I got a test RMSE mean of 2436, std of 791, train rmse mean of 1368, std of 109. The 0th iteration was 24,584 test rmse mean, 1346 std. 24,611 train rmse mean error, 336 std. In the most recent shape up paper, it was stated that the best results

Figure 4.6: Training of Automated Caeasr Point Placement



were 3070 for men, and 2630 for women using 3D PC + Anthro with 5 fold CV. I also tried a graph neural network. Where the data is .obj file with vertex and face information. I removed face information and sampled the vertex information. Next, I reformulated the classification architecture for regression. I did a quick test, trained only on 10 scans, after 50 epochs had about 8000 L1 loss. I also investigated sparse autoencoders, denoising, and contrastive for dimensionality reduction in comparison with principal component analysis.

Previous best results for RMSE on male were 3.07kg and 2.63kg for female while these results average across genders with a validation RMSE of 2.44kg. Demonstrating non-linearities in the dataset.

4.6 parabolic setup

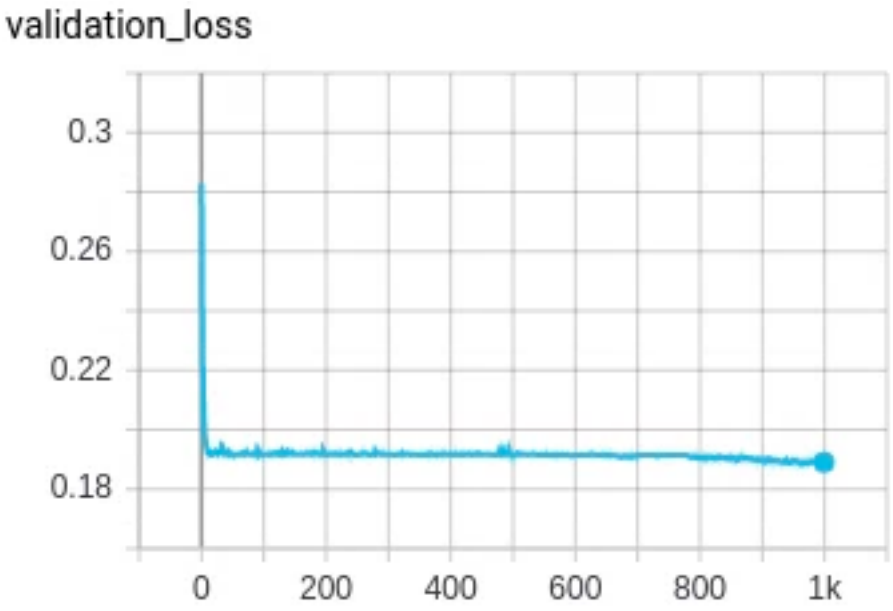
The fligh was postponed. Several tries were done. Originally a multi-post base was designed. This added support but also complexity. Which is why a single pole was chosen over it. Along with this, the single pole was quite stable as the weight of the sensors is light.

4.7 3d imaging testing results

Results were as follows and are provided in the figures.

We can see how at this setting, a higher frame rate helps with the imaging. There is more noise at 6 frames per second and it looks like the background is more part of the body because the sensor

Figure 4.7: Validation of Automated Caeasr Point Placement



has more difficulty tracking the larger change in positioning of the mannequin.

4.7.1 mannequin imaging

Figure 4.8: Training of Automated Caeasr Point Placement without beginning

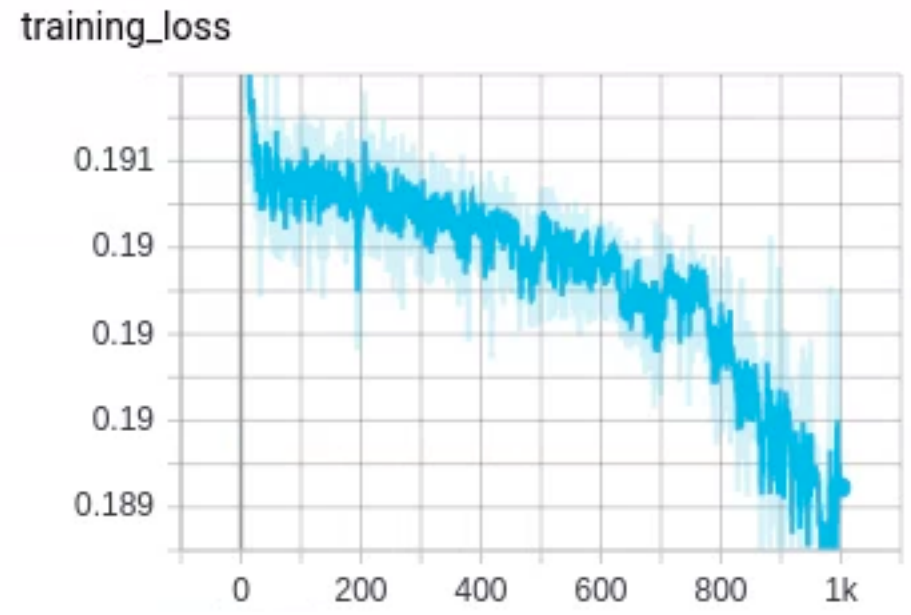


Figure 4.9: Validation of Automated Caeasr Point Placement without beginning

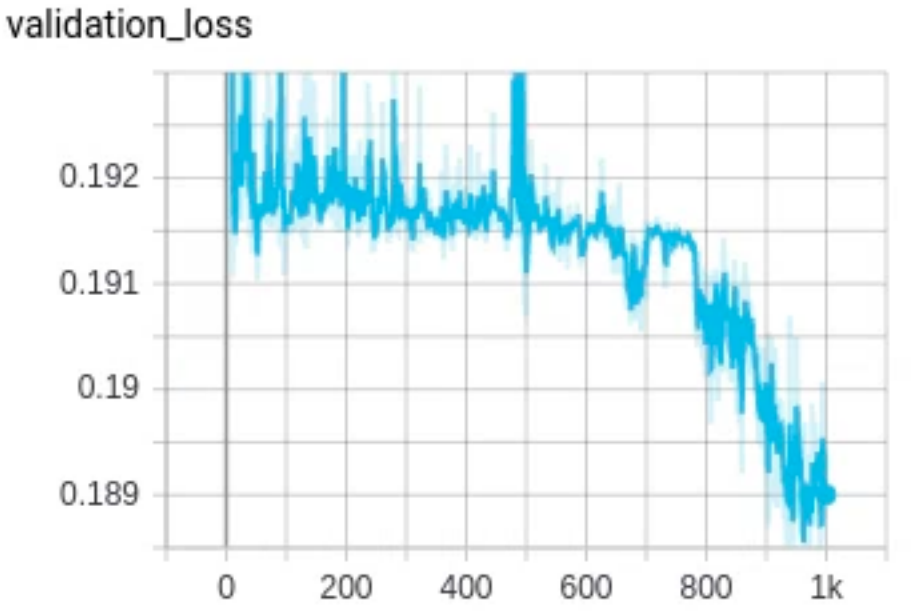


Figure 4.10: Missing data

```
SubjectID                 0
Race                      0
Age                       0
Percent_HBA1C             0
Manual_Height             0
Manual_Weight             0
Manual_BMI                0
WBTOT_FAT                 0
VFAT_MASS                 0
waist_natural_girth       5
hips_at_width_max_girth   6
biceps_right_girth        0
thigh_right_max_girth     0
waist_hip_ratio           6
waist_height_ratio        5
dtype: int64
```

Figure 4.11: HBA1C Distribution

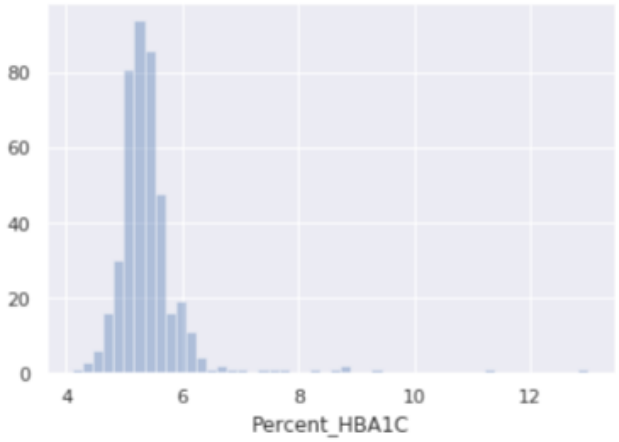


Figure 4.12: WBTOT FAT Distribution

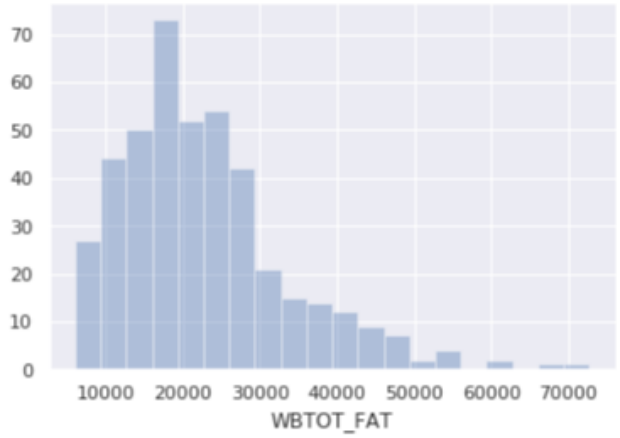


Figure 4.13: VFAT MASS Distribution

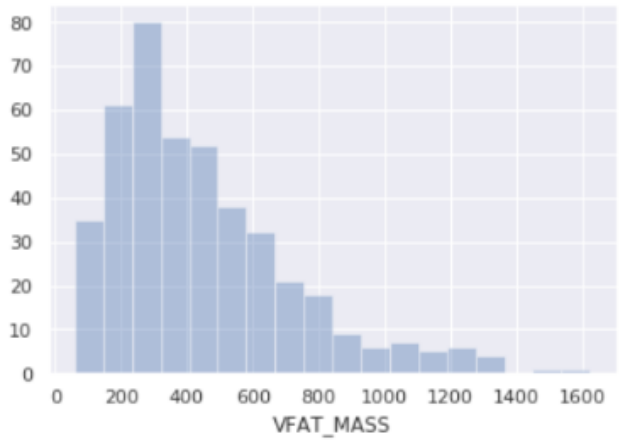


Figure 4.14: Race Ratios Distribution

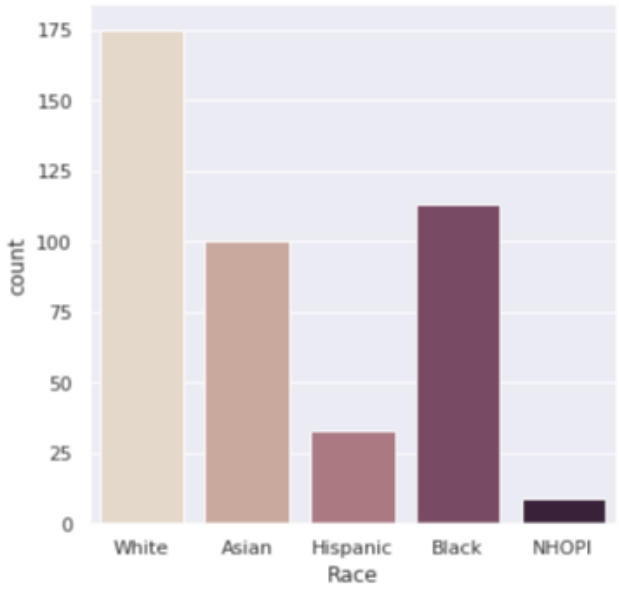


Figure 4.15: Fat Mass Training Table

	iterations	test-RMSE-mean	test-RMSE-std	train-RMSE-mean	train-RMSE-std
0	0	24583.988336	1346.137968	24611.077377	335.515052
1	1	23954.612166	1341.481161	23977.641413	320.422257
2	2	23332.202447	1336.817610	23360.064226	318.256459
3	3	22778.629031	1346.884986	22769.748008	306.131040
4	4	22185.000104	1330.592870	22174.481380	295.515801
...	...	...	...	...	...
995	995	2435.762452	790.463304	1368.520825	108.975341
996	996	2435.756712	790.465094	1368.454910	109.008941
997	997	2435.712977	790.657594	1368.204666	109.079069
998	998	2435.699738	790.638334	1368.092496	109.120926
999	999	2435.749463	790.611023	1367.707004	109.137616

1000 rows × 5 columns

Figure 4.16: Mannequin imaged at 90 frames per second

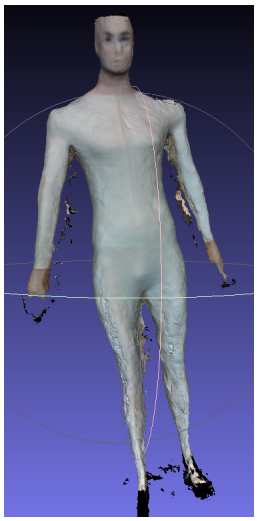


Figure 4.17: Mannequin imaged at 6 frames per second

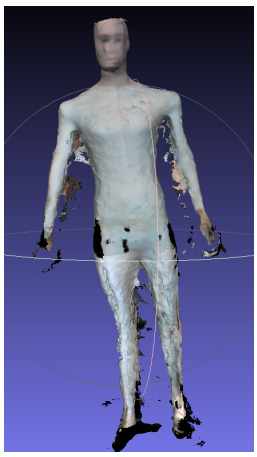


Figure 4.18: Inversion Table Imaging 0°, 30°, 60°, 90°

Figure 4.19: Inversion Table Imaging 0°, 30°, 60°, 90°

CHAPTER 5
CONCLUSION

I did a comparison of various 3d imaging sensors. I found that the closer the sensor to the object, the less spatial noise; however there is a certain minimum distance for each sensor that it can image.So in general one wants to get the sensor as close as possible beyond this minimum distance. The D435 had the lowest error when using a stationary target. The temporal noise for the sensors were all extremely low. Fill rate had interesting results with the Kinect 360 having the highest fill rate. While the azure kinect had the lowest fill rate. One can clearly see the poor fill rate when looking at the depth image of the azure kinect. It would seem like one of the reasons they did this was to boost the accuracy of the sensor on pixels it was sure of. While the previous kinects have these edge problems, they still reported a value regardless of its accuracy.

The graph neural network is the first method to use deep learning to automatically pick caesar points. The performance could be greatly improved with more data. Currently only about 1500 scans were used. This is because each scan also has to be human labelled with 75 points which is time consuming.

5.1 Widgets

5.1.1 Sub-Widgets

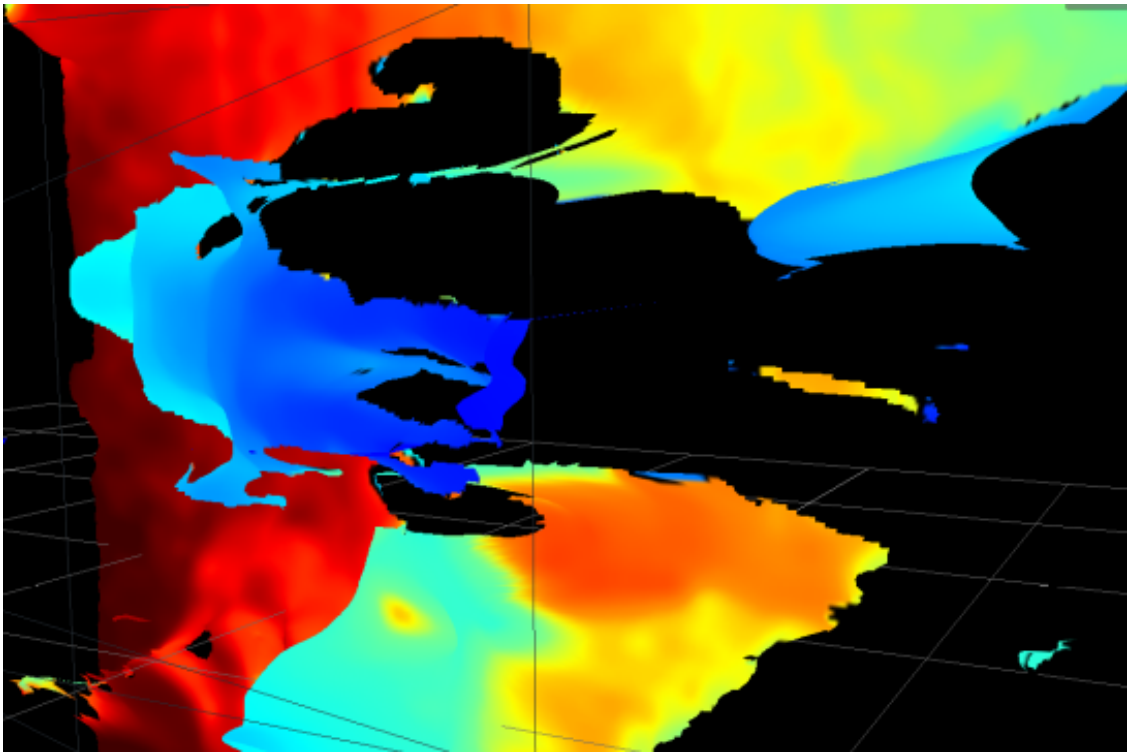
Sub-Sub-Widgets

Para-Widgets

Sub-Para-Widgets

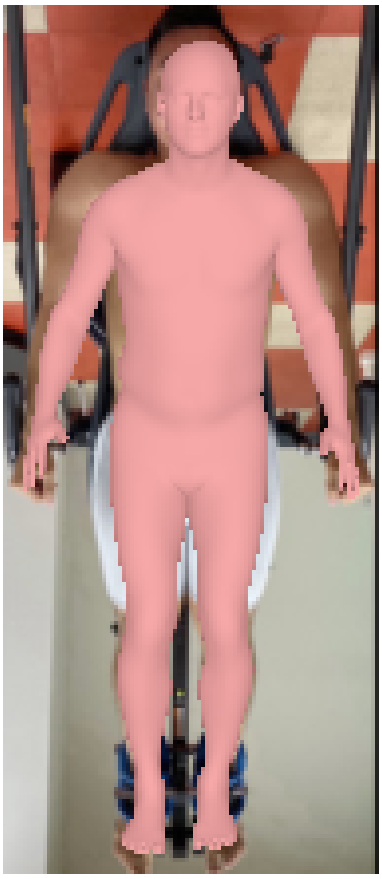
APPENDIX A
SOME ANCILLARY STUFF

Figure A.1: Inversion table depth image from the intel d435 in color



Experiment #	Description	Finished	Sensor	Score	Depth Format		Image Re
1		Yes	435		90fps		
2		Yes	435		6fps		
3		Yes	435		90fps		
4		Yes	435		90fps		
5		Yes	435		90fps		
6		Yes	435		90fps		
7		Yes	435		90fps		
8		Yes	435		90fps		
9		Yes	435		90fps		
10		Yes	415		90fps		
11		Yes	435		90fps		
12		Yes	435		90fps		
13		Yes	435		30fps		
14		Yes	435		90fps		
15		Yes	435		90fps		
16		Yes	435		90fps		
17	Frame rate	Yes	435	2	90fps		
18		Yes	435	2	60fps		
19		Yes	435	1.5	30fps		
20		Yes	435	1	15fps		
21		Yes	435	1	6fps		
22		Yes	415		6fps		
23	Rotation Speed	Yes	435		90fps		
24		Yes	435		90fps		
25		Yes	435		90fps		
26		Yes	435		90fps		
27		Yes	435		90fps		
28	Offline reconstruction	Yes	435		90fps		
29	Depth Resolution	No	435		30	320 x 180 6fps	1280 x 72
30	Depth Resolution	No	435		30	320 x 180 6fps	848 x 480
31	Depth Resolution	No	435		30	320 x 180 6fps	640 x 480
32	Depth Resolution	No	435		30	320 x 180 6fps	640 x 360
33	Depth Resolution	No	435		30	320 x 180 6fps	480 x 270
34	Single sensor moving	Yes	435		30fps	1280 x 720	1280 x 72

Figure A.2: SPIN 3d reconstruction on single 2d rgb image



BIBLIOGRAPHY

[1] Carlina V Albanese, Evelyn Diessel, and Harry K Genant. Clinical applications of body composition measurements using dxa. *Journal of Clinical Densitometry*, 6(2):75–85, 2003.

[2] Brett Allen. *Learning body shape models from real-world data*. Citeseer, 2005.

[3] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[4] Sundara Tejaswi Digumarti, Gaurav Chaurasia, Aparna Taneja, Roland Siegwart, Amber Thomas, and Paul Beardsley. Underwater 3d capture using a low-cost commercial depth camera. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016.

[5] Anna Veronika Dorogush, Andrey Gulin, Gleb Gusev, Nikita Kazeev, Liudmila Ostroumova Prokhorenkova, and Aleksandr Vorobev. Fighting biases with dynamic boosting. *CoRR*, abs/1706.09516, 2017.

[6] NJ Fuller, SA Jebb, MA Laskey, WA Coward, and M Elia. Four-component model for the assessment of body composition in humans: comparison with alternative methods, and evaluation of the density and hydration of fat-free mass. *Clinical Science*, 82(6):687–693, 1992.

[7] Xian-Feng Han, Hamid Laga, and Mohammed Bennamoun. Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. *CoRR*, abs/1906.06543, 2019.

[8] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[9] Mohammad Hussain, Yoshihiro Okada, and Koichi Niijima. Efficient and feature-preserving triangular mesh decimation. 2004.

[10] Daniel P Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.

[11] Intel. Intel realsense camera depth testing methodology. Technical report, 2018.

[12] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of*

the 24th annual ACM symposium on User interface software and technology, pages 559–568, 2011.

[13] Michel Y Jaffrin and Hélène Morel. Body fluid volumes measurements by impedance: A review of bioimpedance spectroscopy (bis) and bioimpedance analysis (bia) methods. *Medical engineering & physics*, 30(10):1257–1269, 2008.

[14] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[15] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2252–2261, 2019.

[16] Benjamin Langmann, Klaus Hartmann, and Otmar Loffeld. Depth camera technology comparison and performance evaluation. In *ICPRAM (2)*, pages 438–444. Citeseer, 2012.

[17] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1886–1895, 2018.

[18] Bennett Ng, Markus Sommer, Michael Wong, Ian Pagano, Yilin Nie, Bo Fan, Samantha Kennedy, Brianna Bourgeois, Nisa Kelly, Yong Liu, Phoenix Hwaung, Andrea Garber, Dominic Chow, Christian Vaisse, Brian Curless, Steven Heymsfield, and John Shepherd. Detailed 3-dimensional body shape features predict body composition, blood metabolites, and functional strength: the shape up! studies. *The American journal of clinical nutrition*, 110, 09 2019.

[19] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[20] Wojciech Sankowski, M Włodarczyk, Damian Kacperski, and Kamil Grabowski. Estimation of measurement uncertainty in stereo vision system. *Image and Vision Computing*, 61:70–81, 2017.

[21] Ali Sophian, Wahju Sediono, Muhammad Ridzwan Salahudin, Mohd Saatarie Mohd Shamsuli, and Dayang Qurratu'aini Awang Zaaba. Evaluation of 3d-distance measurement accuracy of stereo-vision systems. *Int. J. Appl. Eng. Res.*, 12(16):5946–5951, 2017.

[22] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018.

[23] Yueh-Ling Lin and M. J. Wang. Constructing 3d human model from 2d images. In *2010 IEEE 17Th International Conference on Industrial Engineering and Engineering Management*, pages 1902–1906, 2010.