Online Semantic Labeling of Deformable Tissues for Medical Applications

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

John Joseph O'Neill

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILISOPHY

TIMOTHY M. KOWALEWSKI, PhD

May, 2017

Acknowledgements

There are many people that have earned my gratitude for their contribution to my time in graduate school. I would like to personally thank the fellow members of my lab that have directly assisted with my research. These include, but are not limited to, Rodney Dockter II, Trevor Stephens, Anna French, Sachin Bijadi, Jason Lu, Reed Johnson, and Mark Brown. I would also like to acknowledge RONNY for his strength, perseverance, and ability to inspire me daily.

Dedication

To my loving wife.

Abstract

Surgery remains dangerous, and accurate knowledge of what is presented to the surgeon can be of great importance. One technique to automate this problem is non-rigid tracking of time-of-flight camera scans. This requires accurate sensors and prior information as well as an accurate non-rigid tracking algorithm. This thesis presents an evaluation of four algorithms for tracking and semantic labeling of deformable tissues for medical applications, as well as additional studies on a stretchable flexible smart skin and dynamic 3D bioprinting. The algorithms were developed and tested for this study, and were evaluated in terms of speed and accuracy. The algorithms tested were affine iterative closest point, nested iterative closest point, affine fast point feature histograms, and nested fast point feature histograms. The algorithms were tested against simulated data as well as direct scans. The nested iterative closest point algorithm provided the best balance of speed and accuracy while providing semantic labeling in both simulation as well as using directly scanned data. This shows that fast point feature histograms are not suitable for nonrigid tracking of geometric feature poor human tissues. Secondary experiments were also performed to show that the graphics processing unit provides enough speed to perform iterative closest point algorithms in real-time and that time of flight depth sensing works through an endoscope. Additional research was conducted on related topics, leading to the development of a novel stretchable flexible smart skin sensor and an active 3D bioprinting system for moving human anatomy.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis presents an evaluation of four algorithms for tracking and semantic labeling of deformable tissues for medical applications, as well as additional studies on a stretchable flexible smart skin and dynamic 3D bioprinting.

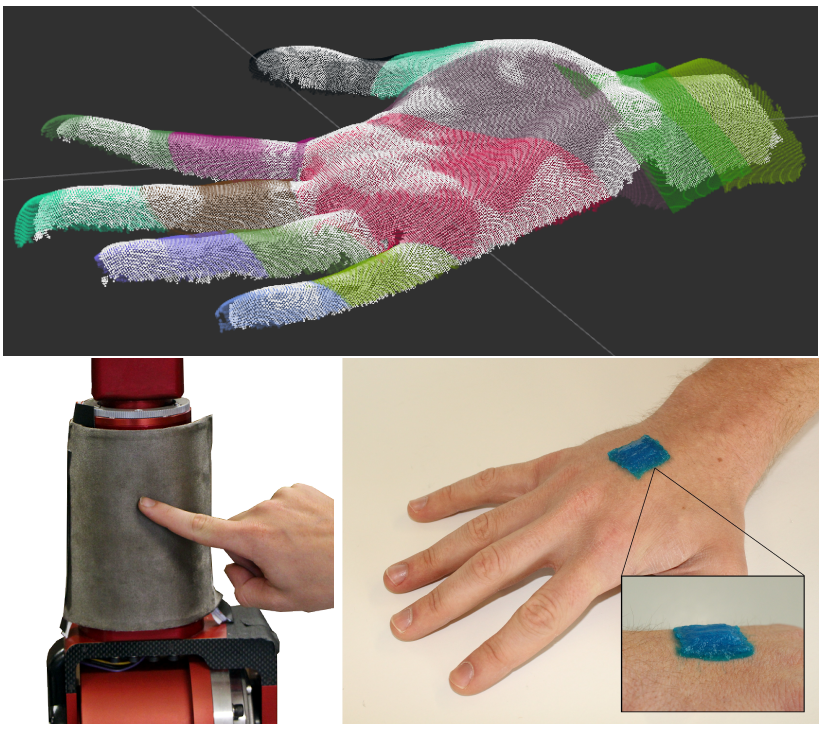


Figure 1.1: Example of scan with semantic labeling, a smart skin, and 3D bioprinting.

1.1 Overview

The long-term goal of this research was to improve the safety of surgery by increasing automation of minimally invasive surgery. The primary aim of this research was to demonstrate and quantify the limits of real time end effector tracking and registration of soft anatomy through the evaluation of four algorithms. Secondary aims were to validate core assumptions of the research and further evaluate the performance of the algorithms.

The four algorithms tested were an Affine Iterative Closest Point, a Nested Iterative Closest Point, an Affine Fast Point Feature Histograms, and a Nested Fast Point Feature Histograms. They were evaluated via the following three experiments:

- Experiment 1: Deform pliable synthetic hand to known geometry, then incrementally decrease resolution to evaluate limits of accuracy and speed for each candidate algorithm.
- Experiment 2: Deform pliable synthetic hand to known geometry, then incrementally increase artificial noise to evaluate limits of accuracy and speed for each candidate algorithm.
- Experiment 3: Collect real-time scan of synthetic hand with time of flight camera to evaluate limits of accuracy and speed for each candidate algorithm in a real-world situation.

The primary experiments showed that the Nested versions of each algorithm provided an improvement in Root Mean Squared Error at the expense of Framerate for all three experiments, for example reducing from $3.6mm$ to $2.0mm$ for Iterative Closest Point in experiment 2. The Fast Point Feature Histogram algorithms vastly underperformed the Iterative Closest Point algorithm in both Framerate and Root Mean Squared Error for all three experiments.

The secondary experiments showed that the Graphics Processing Unit provides enough speed to perform Iterative Closest Point in real-time ($> 10Hz$) and that time of flight depth sensing works through an endoscope. The secondary experiments also validated the theoretical algorithmic complexity and showed that reverse nested matching showed no improvements.

Two additional projects were also developed to provide alternative contributions to the field of real time robotic tracking with medical applications so as to make more substantial progress towards the ultimate goal. These include a stretchable and flexible Smart Skin for sensing position and force, which allows creating an synthetic organ that could track surgical interactions ratiometrically to allow stretching of the organ but retaining relative tool position measurements. Another study was a Dynamic 3D Bioprinting system for moving human anatomy where even though geometric surface tracking was feature poor, alternative approaches show the potential benefits of being able to track changing anatomy in real-time and bringing additive manufacturing to surgery.

1.2 Specific Contributions

The contributions of this these are summarized here and detailed throughout the thesis.

- Developed affine version of Fast Point Feature Histogram matching algorithm
- Developed modular nested rigid framework incorporating semantic labeling
- Validated GPU version of Iterative Closest Point
- Evaluated impact of closest point search direction on speed and accuracy
- Validated algorithmic complexity of fully implemented algorithms
- Validated time of flight functionality through endoscope
- Developed inverse polynomial fit for Smart Skin position sensing
- Created simple Smart Skin circuitry
- Invented Smart Skin force estimation from contact resistance
- Invented Smart Skin diagonal position estimation method
- Developed dynamic 3D Bioprinting on moving human anatomy
- Validated use of micro jetting system to perform 3D Bioprinting

1.3 Motivation

Despite medical advances, surgery remains high risk. Surgical errors have been estimated to account for at least 32,000 deaths per year in the United States, placing it among the top fifteen causes of death in the US [1, 2, 3]. Specifically, vascular injury accounts for one third of complications in laparoscopy [4]. It is the second highest cause of death within laparoscopic surgery, second only to anesthesia, with a mortality rate estimated at fifteen percent [5]. Surgery is also extremely common, currently surgeries are performed at a rate of 50 million per year, giving the average American an expected seven surgeries in their lifetime [6], while medical care becomes increasingly cost conscious [7].

Surgery requires real-time knowledge of the location of the tooltip in order to inform surgical decisions [8, 9] and many attempts have been made to track tissues including using stereoscopic [10, 11] and ultrasound [12, 13] sources. Since tissues are nonrigid this provides unique challenges that are not present in standard rigid tracking. The related requirement to tracking is registration and semantical labeling, where the results of the tracking is used to register what is being seen by the robot to a functional model or prior dataset such as high resolution MRI or CT data to make productive decisions about the scene.

There are many other robotic applications for nonrigid tracking [14] including facial recognition [15, 16], agriculture [17] and food service [18]. As robots become more and more integrated with human lives, they will have to become more adept at interpreting the world in a non-rigid manner, and using that knowledge to make more educated decisions regarding the objects they are interacting with.

Chapter 2

Background

2.1 Surgical Robotics

Minimally invasive surgery became prevalent beginning in 1986 when improvements in camera chips and lighting devices were used to give the surgeon a more flexible viewing arrangement [19]. The use of minimally invasive surgery surged in the 1990's, with nearly half of all general surgeries performed in a minimally invasive manner by 1999 [20]. Minimally invasive surgery has many benefits over traditional open surgery, including superior outcomes manifested as improved survival, fewer complications, and quicker return to functional health and productive life [20].

Figure 2.1: da Vinci robotic surgical system (image from of intuitivesurgical.com)

The next step forward in minimally invasive surgery was telerobotic surgery, or telepresence surgery, shown in Figure 2.1, where a robot with electrically actuated tools is controlled using a physically separate computer console [21]. This master-slave situation gives the operator an immersive stereoscopic 3D view and control over the movements of the robot with motion scaling, easy control of both the camera and the instrument, and an extra 'wrist' joint on the end of the instrument not available in conventional laparoscopic surgery [22]. However, the system does not add any intelligence to the situation, and the surgeon still has to perform the surgery [23].

2.2 Background on 3D Sensing

The oldest method of non-contact distance sensing is triangulation, made useful in 1787 by Jesse Ramsden's theodolite [24]. The origins of automatic non-contact depth sensing began in World War II, when it was used to learn the location of enemy planes [25, 26]. As electronics became more sophisticated, and digital photography allowed computer vision, these methods became more automated and more methods became possible such as moire, holographic interferometry, lens focus, and Fresnel diffraction [27]. There has been research into many different 3D sensing modalities for use in surgery [28, 29], including optical and visual.

2.3 Medical Diagnostics (MRI, CT)

Medical imaging technologies include 2D methods such as X-ray, as well as 3D methods such as Magnetic resonance imaging (MRI) and X-ray computed tomography (X-ray CT).

MRI scanners form images of the body (or other target) by using strong magnetic fields and detecting and measuring the radio frequency signal which is emitted by hydrogen atoms (such as in water molecules). Since these signals are related to the rate at which the excited atoms return to their equilibrium state, different tissues can be differentiated [30]. The MRI builds up volumetric data from a series of 2D 'slices' to form 3D voxels (volumetric pixels) in the sub-millimeter range [31].

X-ray computed tomography (X-ray CT) takes a series of many X-ray images from all angles of the target, and uses those with the help of a computer to give resultant tomographic images, or 2D 'slices' of the target. These slices can then be stacked to give 3D voxels in the sub-millimeter range [32, 33].

2.4 Medical Interventional (Flouro, US)

For real-time intra-procedural use, an essentially continuous video-like X-ray is called fluoroscopy. This can be scanned in 2D at up to video framerates, but is more limited by the radiation dosage received by the patient by so many continuous X-ray images [34, 35]. These images can also be reconstructed into 3D in a similar manner to conventional CT [36].

Ultrasonography (US) operates by sending a pulse of ultrasonic waves into a medium, and then measuring the delay and the intensity received, a knowledge of distance and material properties can be deduced. Ultrasonography operates in a similar manner to RADAR using ultrasonic waves instead of radio waves, therefore giving a much smaller working volume with higher spatial accuracy. 3D ultrasonography is also available by scanning at different angles and using many of the same techniques as CT [37, 38, 39].

2.5 General Surface Non-Contact

2.5.1 Structured Light

Structured light works by projecting a known, non-repeating pattern onto a scene, as shown in Figure 2.2. The camera then reads that pattern, and given the distance between the camera and the projector, the distance can be calculated for that point. These distances can then be turned into points in 3D space by tracing a ray from the camera aperture out at an angle that

is a function of the pixel u,v values, which could be a simple pinhole camera model, or a more complicated model depending upon the accuracy required and the lens characteristics [40, 41].

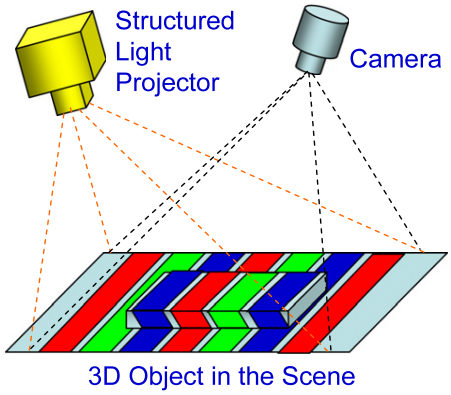


Figure 2.2: Structured light overview [40]

2.5.2 Time of Flight

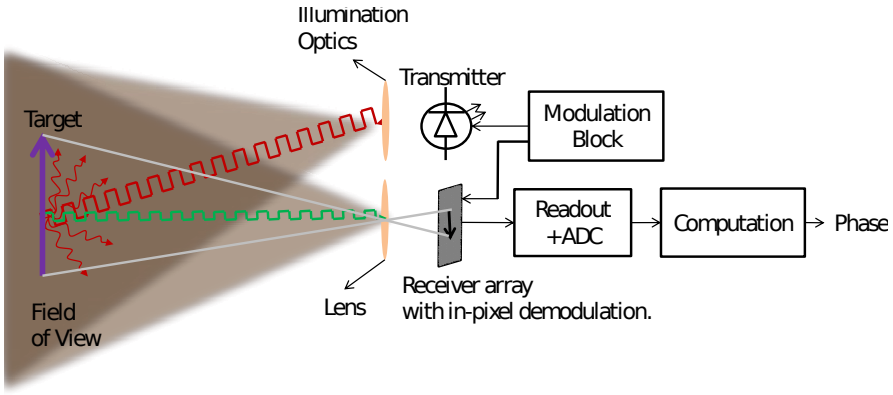


Figure 2.3: Time of flight overview [42]

Time of flight works by sending out a modulated signal and measuring the phase shift of the returned signal, as shown in Figure 2.3. This phase shift in time is then multiplied by the speed of light to get the distance to and from the object, e.g. if the signal was delayed one nanosecond, the flight distance would be 30cm, or the object would be 15cm away [43, 42].

It has been shown that a time of flight sensor can be transmitted through an endoscope [45], as shown in Figure 2.4. Using a commercial Time of Flight sensor, sub-millimeter resolution

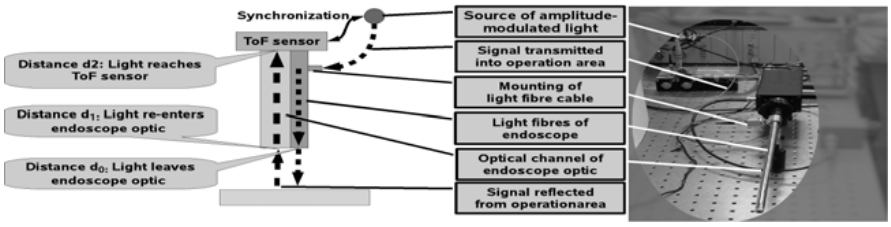


Figure 2.4: Time of flight through endoscope setup [44]

was achieved with a 64x48 pixel unit running at five frames per second. This has the benefit that the laser pulse signal can be diffuse, and therefore can be transferred through the optical fibers already present in current endoscopes. Additionally it has been shown that these surfaces can be registered to a prior scan, such as a CT scan, however the registration was rigid and performed with ICP from nine manually selected corresponding landmarks [44]. This has also been extended to incorporate RGB color data as well [46, 47, 48], however RGB color can be inconsistent in a minimally invasive surgery.

2.5.3 LIDAR

LIDAR is similar to time of flight, but instead of measuring the phase shift of a repeating signal with every pixel simultaneously, LIDAR usually measures only one pixel at a time, by pulsing a laser once and awaiting the return of the signal, then moves a mirror to aim the laser at the next pixel and measures the distance there. This can allow accurate measurements and long measurement distances (up to the km range), but at the cost of moving parts and slower refresh rates.

2.6 Computing

General purpose consumer grade computers provide several different processing cores which can be used for running algorithms.

2.6.1 Central Processing Unit

The Central Processing Unit, or CPU, is the chip in a personal computer responsible for executing instructions and performing calculations. In recent years CPU chipmakers have been turning to increasing the number of cores in lieu of increasing processing speed [53]. For example the AMD FX-6300 CPU has six $3.5GHz$ cores running in parallel. This provides an incentive to

	Temporal Resolution	Spatial Resolution	Surface Volumetric/	Notes
MRI	4-50 Hz [49]	sub-mm	Volumetric	Requires specialized equipment to be used intraoperatively
CT	6-12 Hz [50]	sub-mm	Volumetric	Overexposure can be hazardous [51]
Flouro	1-30 Hz	1mm	Shadow	Overexposure can be hazardous [52]
US	30 Hz	1mm	Shadow	Shallow depth of view
3D ToF	30 Hz	1mm	Surface	Requires air cavity

Table 2.1: Comparison of imaging modalities.

run code on more than one core in order make more efficient use of the computing resources available. This can be achieved by running multiple programs simultaneously, or by running one program which runs multiple threads, each processing a different subset of the data.

Programming for multicore processors has become easier with the OpenMP library for C++ [54]. This allows code to be converted to multithreaded with a minimal amount of work from the programmer, so long as the code was already written in an easily parallelizable manner. However gains from multithreading are limited by the serial section of the code, known as Amdahl's Law [55].

2.6.2 Graphics Processing Unit

The Graphics Processing Unit, or GPU, is the computational brain behind a graphics card. In normal personal computing the GPU is tasked with translating the 3D model of a scene into the rendered and shaded image that is outputted to the computer monitor. As multi core CPUs have become more common, and individual core speeds are no longer accelerating as rapidly as they were, the Graphics Processor Unit, or GPU has increasingly been leveraged for general purpose computing [56, 57].

Originally, custom shaders were written, hijacking the graphics pipeline to perform complex calculations instead of rendering graphics. The image, instead of being rendered to the screen, was copied elsewhere to be analyzed not as a color image, but as the result of the equations desired. However, a general purpose GPU computing structure was created by NVidia with their CUDA system [58]. This allows programmers to simply write the code they want copied to the GPU, executed and returned.

GPU computing can provide massive speed improvements, for example an NVidia GTX 960 (Maxwell architecture) has 1024 computing cores, at $1.1GHz$ each ideally totaling $1100Giga-flops$ (a Gigaflop is a unit of computing speed equal to one billion floating-point operations per second), compared to only 6 cores at $3.5GHz$ in an AMD FX-6300 CPU ideally totaling $14Gigaflops$. This should theoretically provide a $314x$ improvement over a single core process, or a $78x$ improvement over a multi-threaded application running on the CPU. However this is only true of what are referred to as embarrassingly parallel tasks [59], where the task can be trivially divided into many separate tasks that have no bearing upon each other. Even with an embarrassingly parallel task, the additional overhead can still add significant time cost [60]. Overhead to run code on the GPU is much larger than the overhead required to run code in parallel on several CPU cores. GPU acceleration has been shown to improve tracking for time of flight in surgical applications [61].

2.6.3 Robot Operating System

The Robot Operating System, or ROS, is an operating system for robotics research [62]. The system has been growing [63] and has been used for medical applications [64, 65] as well as for traditional robotics applications [66, 67]. The system consists of modular nodes, which can be written in C++ or python, communicating over the network locally or including other machines on a network. These nodes communicate with messages, which follow a defined structure, either from the list of standard messages, or custom messages designed for a specific need. This allows a mix and match system, where a combination of nodes can work together where some are custom made, some part of official ROS packages, and others downloaded from a third party.

The node framework of the ROS system also provides an inherent multithreading and caching capability, where different steps of a process are performed in different threads asynchronously, and each node can decide to discard missed messages, or cache them to analyze them all eventually. An example of rigid depth camera simultaneous localization and mapping in 3D is shown in Figure 2.5.

2.7 Algorithms for Registration

Any two scans of an object will likely contain a different subset of points on the surface of the object being scanned. The problem of understanding which points belong to the same real-world location is known as the 'correspondence problem' and has been studied for decades [68, 69]. Also, any 3D scanning technology will inherently contain measurement noise. These problems can be addressed by many algorithms in the literature [70, 71, 72], some of which are reviewed here.

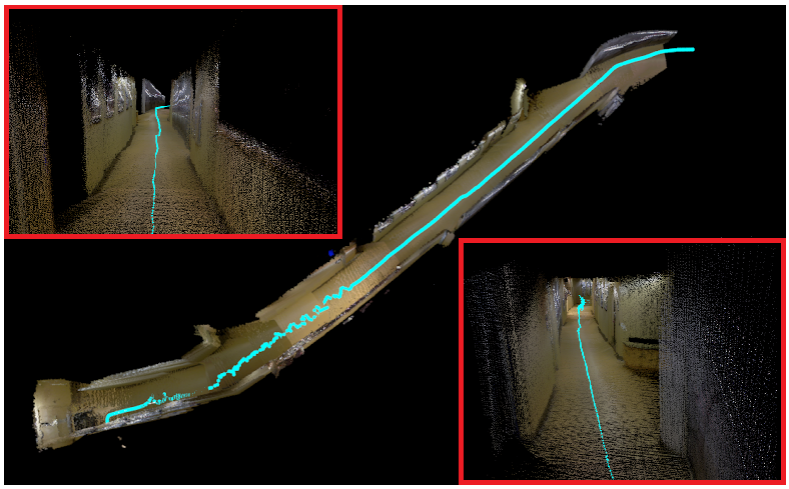


Figure 2.5: ROS Example Depth Camera Mapping.

2.7.1 Iterative Closest Point

First introduced in 1992, the iterative closest point (ICP) algorithm does not require any feature recognition (although it can also be performed on feature points). It simply looks to minimize the distance between each point in set A with with the distance to the closest point in set B, addressing the correspondence problem by assuming that the closest points correspond [73], as shown in Figure 2.6. This is then iterated, since the closest points may have changed, until a target distance or number of iterations is reached. This works best with two sets of points that are already close to being aligned, as otherwise local minima can be reached that do not represent an accurate registration. Therefore ICP is often used to fine tune registration that was initialized [74], or to register two time steps that are near enough that an acceptable first guess can be made [75].

The iterative closest point algorithm has also been improved over the years with much emphasis on being more robust to worse initial guesses and to run faster with larger data sets and real-time processing [76, 77, 78, 79, 80].

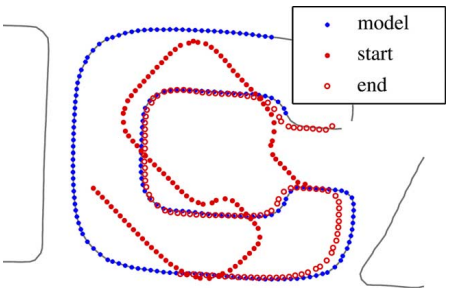


Figure 2.6: Iterative closest point 2D example [80]

2.7.2 Outlier Removal

In order to remove noisy data points, sparse outlier removal can be used by calculating the mean distance of each point to its neighbors, assuming that this follows a Gaussian distribution in which those with higher distances are outliers and therefore can be removed [81, 82, 83], an example of which can be seen in Figure 2.7.

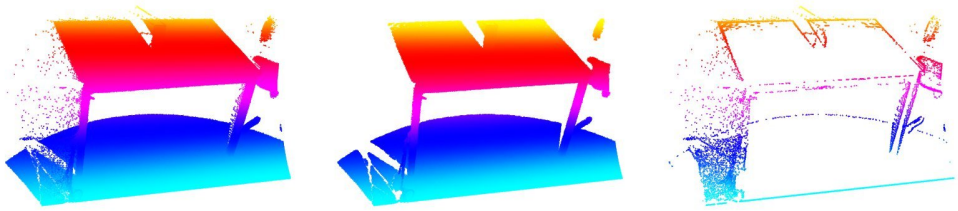


Figure 2.7: Example of outlier removal. Left: all points; middle: outliers removed; right: outliers isolated [82]

2.7.3 Moving Least Squares

The moving least squares algorithm works by fitting a polynomial function (which can be multidimensional) to the data by minimizing a least square cost function, which is either limited to points within a certain radius, or has decreasing cost with points further from the current point of interest. This allows a complicated geometry to be modelled using simple polynomials, and can be used to approximate a surface from a noisy data set or to interpolate an incomplete data set [84, 85]. The implementation in the Point Cloud Library is geared toward moving points from a noisy point cloud into alignment, but does not perform interpolation [82, 81].

2.7.4 3D Hough Transform

A 3D Hough transform works in a similar manner to a 2D Hough transform (fitting a 2D line to feature points), but instead of the equation for a line, the equation for a plane (or other 3D primitive) is used. Each parameter is binned into discrete bins, and then each point is evaluated as to whether or not it satisfies the plane equation at that point. If so, a vote is cast for that plane being a valid plane. Once all of the points are evaluated, the plane with the most votes is considered the most likely to represent a true plane. Since the plane equation has three parameters, each point must be calculated for the number of bins per parameter cubed which can be computationally expensive.

One way to speed up the process is if surface normals are available, the point+normal already defines a plane, so the vote can be cast for only one plane per point, reducing the calculations to one per point. However, surface normals are not inherently available from a depth camera, so those must be calculated for each point [86].

2.7.5 Gaussian Filter

In order to save computational time, denoising can be done in the depth map stage, prior to calculation of 3D position. This is possible because for a Time of Flight camera, the noise in the vertical and horizontal directions is negligible, since those errors are systemic and are affected by the camera lens calibration. Therefore, if there are many pixels, each with a Gaussian noise added to the depth, averaging adjacent pixels should give a more accurate depth to the object. This is called a Gaussian blur, which is shown in Figure 2.8, and can be thought of as a low pass filter with the parameter being how many neighbors are used to calculate each pixel, or the process can be iterated. However, it should be noted that this will blur sharp edges, and will not handle outliers well [87, 88].

2.7.6 Median Filter

A similar low-pass filter used on depth maps is the median filter, where instead of each pixel being assigned the average of it and its neighbors, it is assigned the median. This has a benefit of only using values already in the data set, which avoids smearing and reduces the worry of energy conservation, although if the true value is at the mean one can imagine only values above and below, and the median not providing an accurate value. This algorithm can also be iterated and the radius of neighbors used can be adjusted [89].

2.7.7 Bilateral Filter

A filter that is similar to a Gaussian filter is the bilateral filter [90, 91]. The bilateral filter however attempts to preserve edges more accurately by adjusting weights and looping through the pixels. The bilateral filter has been applied to depth images from structured light sensors [92].

2.7.8 Sobel Operator

In order to find the edges of an image, the simplest method is to take the derivative, which can be done with a Sobel operator [93]. It is a discrete differentiation operator, finding the gradient through a 3x3 kernel. The operation is fast, but does not find the most accurate gradient in the image.

Figure 2.8: Gaussian blur example in RGB photo. Left: original image; Center: five pixel radius blur; Right: twenty pixel radius blur

2.8 Non-Rigid Registration

While iterative closest point can find an optimum rigid body transformation between two sets of points, for deformable bodies the rigid body transformation may not adequately describe the registration of the sets. Situations that require non-rigid registration range from the relatively simple case of a patient with scans in different poses, to comparing anatomy between patients of different age and weight [94, 95]. One example of an algorithm for fitting points to a surface, without needing to know features is by trying to minimize a least squares error distance between the two sets of interest by fitting a transformation matrix, in order of increasing complexity: rigid, affine, trilinear and quadratic. The algorithm also minimizes the difference between the complex transformation and a simple rigid one, to keep the transformation as simple as possible while providing adequate registration. The algorithm also uses octrees (splitting each unit into eight subunits, two per dimension) to keep distance calculations cheap [96]. A non-rigid

registration can include local rigid registrations, down to various levels, or splines in order to keep transition of registrations smooth [94].

2.9 Metrics

The fit registers the scan to the model, and requires a metric to objectively evaluate the quality of the fit. Finding a valid measure of successful fitting has always been a concern for rigid 3D registration [97], and continues to be a concern in non rigid registration [98]. It is of particular concern in medical literature, where evaluation of a successful registration can impact patient outcomes [99, 100]. One metric for evaluating fit is Root Mean Squared Error, or RMSE, which is commonly used to evaluate the quality of a fit [101].

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}} \tag{2.1}$$

The squaring of each term in Equation 2.1 provides an emphasis on outliers, showing them more strongly than mean absolute error (MAE) would [102]. It also works equally well in a point to point method as it does in a point to plane or point to surface method, allowing extensibility.

Another common metric used to evaluate real-time algorithms is framerate [103, 104], which is the number of times the algorithm can run per second. This is therefore the reciprocal of the elapsed runtime of the algorithm, however framerate is more open used in literature due to its inherent readability and intuitive nature [105, 106]. A functional impact of runtime is the impact on latency, where every millisecond spent processing the data is an additional feedback delay to the system if it is meant to run in real time. Real time is assumed to be within average human reaction times, approximately $10Hz$ [107].

Chapter 3

Methods

3.1 Experimental Setup

In order to prevent surgical errors, such as laparascopic blood vessel injury, it is a prerequisite to track and semantically identify the target anatomy (e.g. blood vessel vs. kidney tissue). Therefore algorithms were developed and tested to provide an objective evaluation of their efficacy in providing a future solution to this problem.

While preliminary tests were performed using a kidney and surrounding vasculature, this model was found to be too feature poor to provide adequate features for the experimental algorithms to detect. Therefore, the organ to be tested was changed to a human hand, as the hand provides significant convexity and concavity and was thought to provide a more feature rich environment for testing the algorithms. Another benefit of the hand is that the segmentation was made more simple as each finger could provide a unique and easily identifiable subset of the whole.

The system is designed to mimic the setup shown in Figure 3.1, however since the concept of time of flight through an endoscope has been shown elsewhere, and is further validated in a secondary experiment, the primary experiments will be performed with line of sight to the target, which is shown in figure 3.2.

3.1.1 Silicone Model

The synthetic hand used was a commercially available Female Silicone Hand from Shenzhen Chengyida E-Commerce Co., Ltd, Shenzhen China, and can be seen in Figure 3.3. The hand has realistic skin texture, and was chosen for the similar visual appearance to maximize similarity to a human hand. The model includes analogs to internal bone structure made of wooden dowels

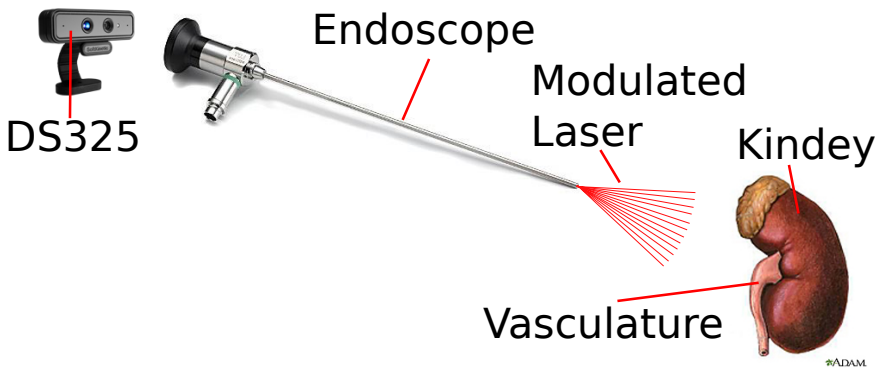


Figure 3.1: Preliminary Experiment Concept

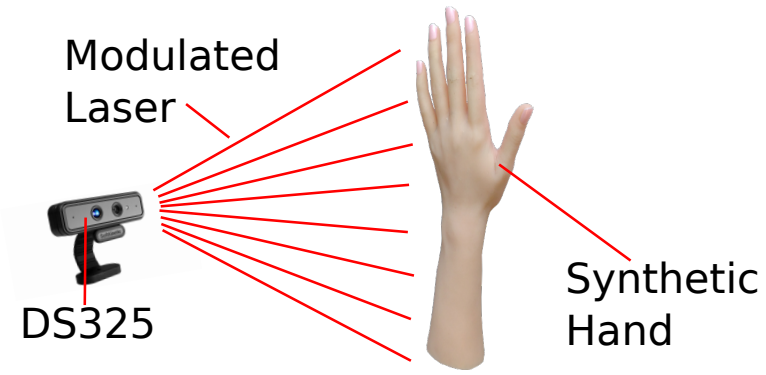


Figure 3.2: Final Experiment Concept

to provide a more rigid base to the softer silicone above. A model was used in lieu of an actual human test subject to provide reliability and the ability to perform slow accurate scans using a static scanner that takes over a minute to complete a scan, which would be an unrealistic amount to time for a human test subject to stay perfectly still.

3.1.2 Static 3D Scanner

The high resolution scanner used was the Artec Spider 3D scanning system (Artec 3D, Luxemburg) with a resolution of $G_s = 0.04mm$. This required a rigid body to be scanned slowly and carefully over a period of time, and then the individual frames were rigidly registered together using proprietary algorithms. This provides a high quality scan, with a resolution of

Figure 3.3: Silicone Model Hand Used

$G_s = 0.04mm$, but only works for fully stationary objects. The scanner can be seen in Figure 3.4. The software allows an interactive smoothing and outlier removal process, and allows exporting of the final scan into an ASCII point cloud file.

Figure 3.4: Artec Spider (image courtesy of artec3d.com)

3.1.3 Time of Flight Camera

The time of flight camera to be used is the DepthSense 325 (SoftKinetic, Ixelles, Belgium), which can be seen in Figure 3.5. This uses the Texas Instruments 3D-TOF chip set [42, 108, 109]. The depth sensor is a OPT8140 chip with a QVGA (320 x 240) resolution. The application programming interface (API) provided by Depthsense allows access to the raw phase data, so that any camera model can be applied to the depth data, which can be generated with standard

computer vision calibration [110].

Figure 3.5: DS325 (image courtesy of softkinetic.com)

The API also provides a 'confidence map' which is essentially the intensity of the modulated light, which provides the benefit of a grayscale image of the scene (in the infrared spectrum).

The device uses an infrared (IR) laser diode to illuminate the scene. This light can be captured and fed into the fiber optics of a typical endoscope. The light is intended to be diffuse, so the fiber optic channel will have no effect other than to change the length of the path of the light, which can be calibrated out with an offset determined by measuring a known plane a known distance from the endoscope tip.

The device also contains an RGB camera and audio microphones, which will not be used for this project.

In order to ascertain whether or not the optics of the depth camera can be modified to correct for the a wide field of view, and to determine if the laser diode could be moved off-board to more easily interface with the fiber optic bundle, the DS-325 was disassembled.

As seen in Fig.3.6, the depth camera has a M10 mounting bracket that is glued to the board, and a 74-degree field of view lens that screws in with an M10 thread. When removed, the CMOS chip is exposed, which means that any custom optics can be placed in front of it, and as long as they are properly aligned, there should be no issues with optics. If a standard off-the-shelf endoscope camera has a similarly sized chip, the optics may be able to be used directly, as long as the mounting is done properly. As seen in Fig.3.7, the laser diode is in a TO-18 package, with three through hole mounting points. This is a fairly standard package for laser diodes, e.g. in optical disk drives.

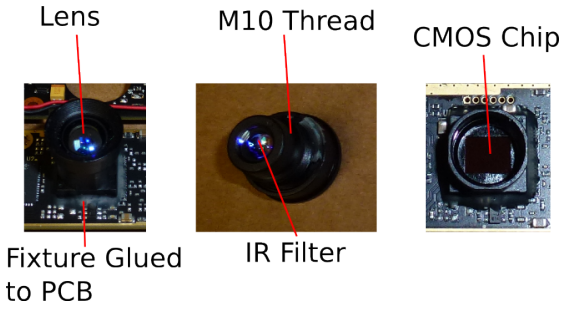


Figure 3.6: Detail of DS-325 depth camera

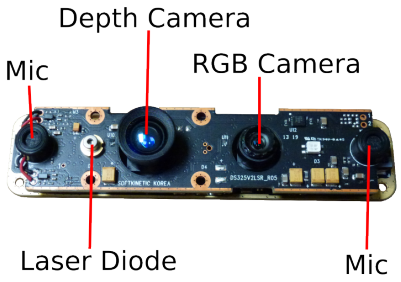


Figure 3.7: Photo of DS-325 without shell

3.2 Algorithms

For each of the Primary Experiments shown in Section 3.4, two transformation models and two alignment algorithms will be used:

Transformation	Iterative Closest Point	Fast Point Feature Histograms with 3D Hough Grouping
Global Affine	Affine ICP (Alg. 1)	Affine FPFH (Alg. 2)
Local Rigid	Nested ICP (Alg. 3)	Nested FPFH (Alg. 4)

Table 3.1: Candidate Algorithms.

3.2.1 Global Affine Transformation

Affine transformation is a combination of translation, rotation, scaling and shear. This allows non-rigid registration while preserving straight lines, which maintains a reasonable number of DOF: at most twelve, with three for translation and nine for the 3x3 matrix containing rotation, scaling and shear.

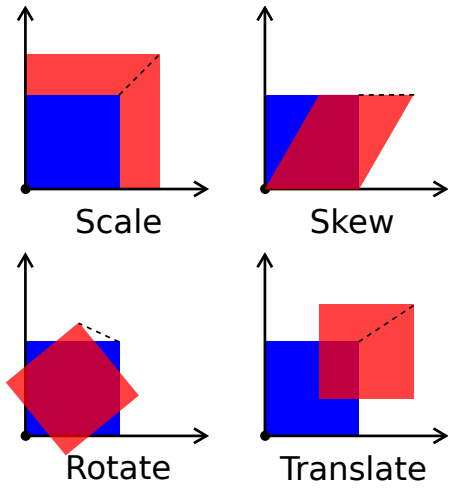


Figure 3.8: Different Affine Transformations

3.2.2 Local Rigid Transformation

Points are clustered into groups of similar transformation, allowing the total representation to vary up to a separate transformation for each point, but down to a small number of bins each with its own rigid transformation.

This can be modified by segmenting the prior data (e.g. the MRI data) during the same process as identifying the organs/regions. Then the prior data is matched to the current scan data, instead of the other way around, with local rigid transformations. The prior data can also be segmented in nested decreasing sizes, e.g. organ level, organ quadrant, etc. This will be performed for at least two levels: organ level, which for a liver would consist of the liver and the blood vessel as separate transformations, then at least one level of subdivision of each organ into 8 sections, being split along 3 orthogonal planes. For the hand, this will consist of each finger as a separate transform, along with a separate transform each for the palm and wrist. Each of those will then be bisected as well. These subdivisions will be determined by a human during the precalcuation stage. See Algorithms 3 and 4.

3.2.3 Affine Iterative Closest Point

Iterative closest point, while usually used for rigid transformations, can be implemented with affine transformations as well, provided there are enough data points [111, 112]. However Iterative Closest Point can be quite susceptible to local minima, so to avoid an exhaustive global search a good initial guess or a small time step is important to achieving good convergence. This algorithm has a computational complexity of $\mathcal{O}(n \times \log(n))$ due to the need for finding nearest neighbors, unless a KD-tree can be precalculated, in which case the algorithm has a computational complexity of $\mathcal{O}(n)$. See Algorithm 7.

3.2.4 GPU Accelerated Iterative Closest Point

Iterative closest point, can be greatly accelerated through the use of the Graphics Processing Unit [113]. The algorithm is ideally not any different than Algorithm 7, however due to being a separate implementation, the exact results may vary.

3.2.5 Fast Point Feature Histograms with 3D Hough Grouping

Point Feature Histograms (PFH) are pose-invariant descriptions of the local surface geometry near a point. This is calculated based on the points' 3D position and surface normal, as well as those of the k nearest points. However, this requires a computational complexity of $\mathcal{O}(nk^2)$, so for real-time applications a Fast Point Feature Histogram (FPFH) can be used, where a simpler weighed average of the surrounding k points are used, which reduces the accuracy of the algorithm but greatly increases the speed due to the reduced computational complexity of $\mathcal{O}(nk)$ [114]. This can then be used to generate a transformation with a 3D Hough grouping approach, with an initial alignment. See Algorithm 2.

When performed with the Local Rigid Transformations, this will be performed for each of the segments of the segmented MRI organ boundary data or hand, which will have their Fast Point Feature Histogram precalculated as part of the initial setup. Then each segmentation level will find a rigid transformation, with each level expected to have less change than the level above it. See Algorithm 8.

Algorithm 1 Affine ICP

```
DATA PRE-CALCULATIONS(Volumetric_MRI)
while System Active do
    ITERATIVE CLOSEST POINT(TargetPointCloud,ReferencePointCloud,Affine)
    POST-REGISTRATION CALCULATIONS(Transformation)
end while
```

Algorithm 2 Affine FPFH

```
DATA PRE-CALCULATIONS(Volumetric_MRI)
while System Active do
    FAST POINT FEATURE HISTOGRAM(Target,Reference,Affine)
    POST-REGISTRATION CALCULATIONS(Transformation)
end while
```

Algorithm 3 Nested ICP

```
DATA PRE-CALCULATIONS(Volumetric_MRI)
while System Active do
    for each level in the semantic tree do
        for each item in the level do
            Find a rigid transformation that is in addition to the rigid transformation of the parent
            ITERATIVE CLOSEST POINT(Target,Reference,Rigid)
        end for
    end for
    POST-REGISTRATION CALCULATIONS(Transformation)
end while
```

Algorithm 4 Nested FPFH

```
DATA PRE-CALCULATIONS(Volumetric_MRI)
while System Active do
    for each level in the semantic tree do
        for each item in the level do
            Find a rigid transformation that is in addition to the rigid transformation of the parent
            FAST POINT FEATURE HISTOGRAM(Target,Reference,Rigid)
        end for
    end for
    POST-REGISTRATION CALCULATIONS(Transformation)
end while
```

Algorithm 5 Data Pre-Calculations

function CONVERT MRI TO SURFACE(MRI)
Convert volumetric data to surface data based on gradients
Human confirms that surfaces are logical
end function
function CONVERT SURFACE TO POINT CLOUD($Surface$)
Surface is sampled at regular interval to create Point Cloud
end function
function CREATE SEMANTIC TREE($PointCloud$)
Human creates tree starting at organ level then subdividing each organ
Each branch is labeled with it's organ type, and whether or not it is touchable
end function
function CREATE K-D TREE($PointCloud$)
Subdivide until all points have a unique leaf
end function

Algorithm 6 Post-Registration Calculations

for each point in the source point cloud **do**
Find the closest point in the registered reference point cloud using K-D tree
NEAREST NEIGHBOR SEARCH($Point$,$ReferencePointCloud$)
$SematicLabel \leftarrow ClosestPointLabel$
end for
Find the closest 'dangerous' point in the target point cloud using K-D tree
NEAREST NEIGHBOR SEARCH($Point$,$TargetPointCloudDangerous$)
Evaluate end effector distance to nearest dangerous point
if $Distance < Threshold$ **then**
Emergency Stop
else
Continue
end if

Algorithm 7 Iterative Closest Point

function ITERATIVE CLOSEST POINT($Target, Reference, [Rigid/Affine]$)
  while $Change \leq Epsilon$ **do**
    for each $\overrightarrow{p}_i$ in $TargetPointCloud$ **do**
      find the closest point in the reference point cloud using K-D tree
      NEAREST NEIGHBOR SEARCH(p_i,$ReferencePointCloud$)
    end for
    for each $\overrightarrow{p}_i$ in $TargetPointCloud$ **do**
      if Rigid **then**
        Estimate the minimum cost rotation and translation
        $\min_T \sum_{i=1}^{N} \|T(\overrightarrow{p}_i) - \overrightarrow{m}_j\|_2^2$
        Where T is a rigid transformation matrix
      else if Affine **then**
        Estimate the minimum cost rotation and translation
        $\min_T \sum_{i=1}^{N} \|T(\overrightarrow{p}_i) - \overrightarrow{m}_j\|_2^2$
        Where T is an affine transformation matrix
      end if
    end for
    for each point in the source point cloud **do**
      Transform the source points using the obtained transformation
      $\overrightarrow{p}_i \leftarrow T(\overrightarrow{p}_i)$
    end for
  end while
  return $Transformation$
end function

Algorithm 8 Fast Point Feature Histogram

function FAST POINT FEATURE HISTOGRAM($Target$,$Reference$,[$Rigid/Affine$])
  Generate K-D tree
  for each $\overrightarrow{p}_i$ in $ReferencePointCloud$ **do**
    Find the neigbors in the target point cloud within radius r using K-D tree
    NEAREST NEIGHBOR SEARCH(p_i,$ReferencePointCloud$)
    for each j neighbor point in $TargetPointCloud$ **do**
      Calculate Simplified Point Feature (SPF), which
      relates to the feature used, such as the relative
      angles between the normals or the Euclidian distance
      $SPF(\overrightarrow{p}_i) \leftarrow f(\overrightarrow{p}_i, \overrightarrow{n}_i, \overrightarrow{p}_j, \overrightarrow{n}_j)$
    end for
  end for
  for each $\overrightarrow{p}_i$ in $ReferencePointCloud$ **do**
    Find the k closest points in the target point cloud using K-D tree
    NEAREST NEIGHBOR SEARCH(p_i,$ReferencePointCloud$)
    for each k neighbor point in $TargetPointCloud$ **do**
      Use the neighboring points to weight the histogram of point i
      $FPFH(\overrightarrow{p}_i) \leftarrow SPF(\overrightarrow{p}_i) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w_k}SPF(\overrightarrow{p}_k)$
    end for
  end for
  Use a 3D Hough grouping method that tries to maintain
  the same geometric relations of the correspondences of the points
  return $transformation$
end function

Algorithm 9 Nearest Neighbor Search

```
function NEAREST NEIGHBOR SEARCH(Point)
    Generate k-d tree
    for each level in the tree do
        Move down the tree recursively, going in the direction of the target point.
        if currentnode = leafnode then
            currentbest ← currentnode
        end if
    end for
    for each level in the tree do
        Move back up the tree recursively.
        if currentnode(closerthan)currentbest then
            currentbest ← currentnode
            if theotherchildwouldhavebeenbetter then
                Move back down the tree to the leaf
                currentbest ← currentnode
            end if
        end if
    end for
    return currentbest
end function
```

3.3 Setup

3.3.1 Scanning

For experiments 1 and 2, a high quality scan was taken of the silicone hand with the Artek 3D scanner. This was a rigid composite scan of the hand in a stationary pose, allowing maximum accuracy. The data was filtered to provide a smooth surface of the hand and remove outliers. The scan was then exported to a point cloud with 108,742 discrete points, to be used for the model.

The hand was then nonrigidly deformed manually, and scanned again with the Artek 3D scanner. The same composting filtering and outlier removal was performed, leading to a point cloud with 96,780 discrete points, to be used for the simulated scan. The undeformed and deformed states of the hand can be seen in Figure 3.9.

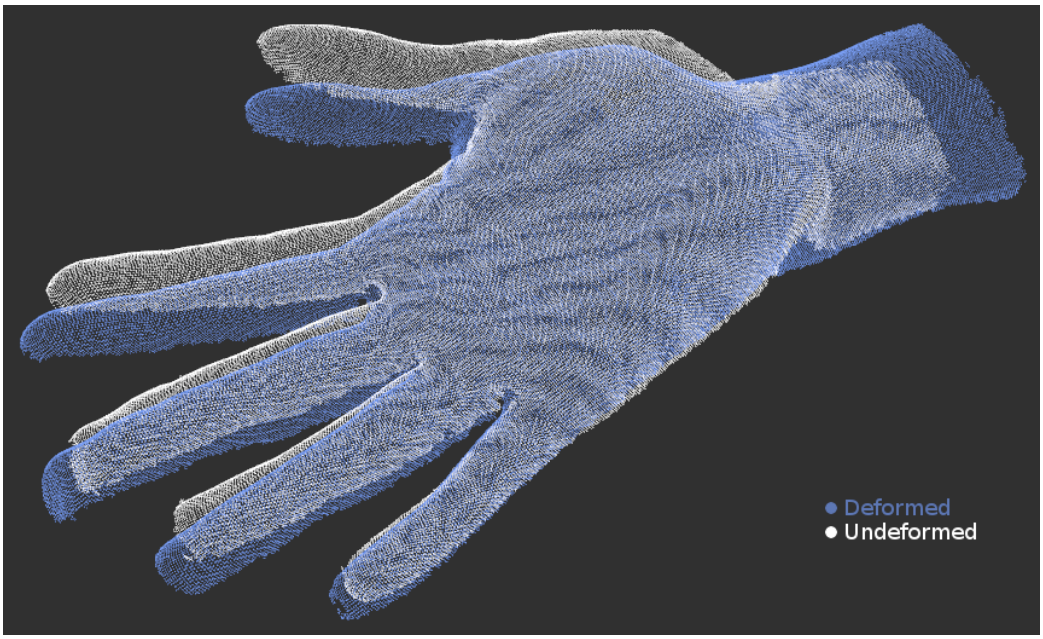


Figure 3.9: Scans of hand in undeformed and deformed states.

For experiment 3, a scan was collected with the SoftKinetic DS325 scanner in realtime to emulate time of flight scans through an endoscope. The scanner was stationary pointing at the hand on the scanning turntable. The scanner was turned on, collecting the raw depth scan data at 45Hz, as well as the confidence illumination data at 45Hz and color data from the RGB camera at 30Hz. The data was saved in a binary ROS bag file. This was collected for 59.3 seconds,

however at approximately the 500th depth frame the side of the hand that was collected for the model became occluded, as the hand had rotated approximately 90 degrees at that point. Therefore, only the first 500 frames were used for the experiment. The occlusion can be seen in the results section at the end of the sample.

3.3.2 Preprocessing

The raw scan data for the model was imported to MATLAB to manually segment into sub bodies. A custom user interface was developed to allow the user to draw lines around each subsegment. Since the object was a hand, the first level of subdivision was the five fingers, the palm, and the wrist. The user creating this first demarcation can be seen in Figure 3.10. These seven segments were further subdivided in half. For the fingers, they were divided roughly in half, between the first and second knuckles. The user creating this lower level demarcation can be seen in Figure 3.11. This provided sufficient resolution for the fingers, so no further subdivision was necessary. The palm was split diagonally, from the base of the pointer finger to the opposite corner, allowing the base of the thumb to move as a single piece. The wrist was divided roughly in half by dividing across the wrist, allowing sharper angles of the wrist.

These subdivided point clouds were saved to ASCII point cloud data files. Each layer was saved separately, such that each point was saved three times, once in the parent file containing all points, once in the second layer, and again in the third layer. The files were named such that the name encoded the nested nature, and additional metadata was not needed to determine the model makeup. The layers were one indexed, to allow zeros to denote an unused layer. This allowed a file search to proceed until the next expected file was not found.

The ASCII point cloud data files are human readable, but not the most efficient storage system. They also did not include estimated surface normals, which are needed by the FPFH algorithm. Therefore, code was written to look first for a binary file and associated normal, denoted by the `"_bin"` suffix and the `"_nrm"` suffix respectively. If the binary and/or normal file were not found, the code would generate them from the ASCII file and save them for future use. This allowed the faster binary load operation while keeping the human readable ASCII files for manual inspection if needed.

3.3.3 Software

Other than preprocessing, all code was written in C++ to run in the ROS framework. This allowed great flexibility in node connections, for example allowing real time demos as well as analyzing simulated data.

Details of the software structure and code, as well as example depth images can be found in

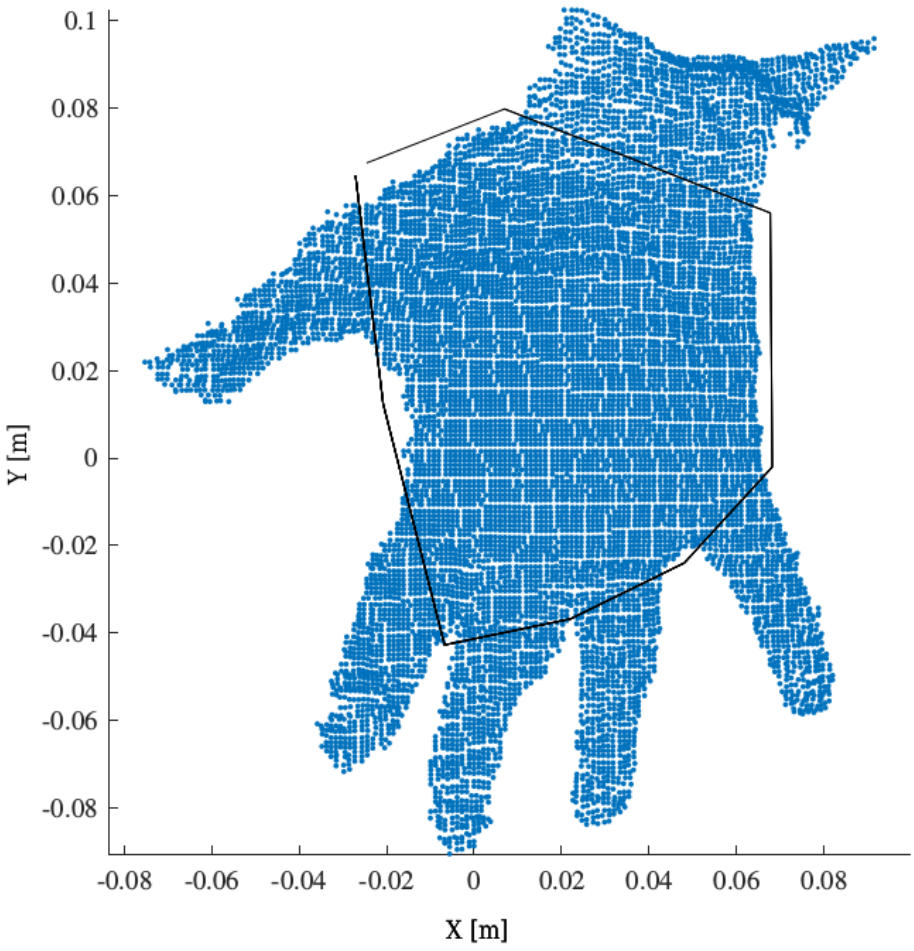


Figure 3.10: Screenshot of first step of MATLAB segmenting process.

Appendix B.

3.3.4 Simulation

For experiments 1 and 2 a single point cloud generated by the Point Cloud Simulation and modified by the Nonrigid Matching node to add Gaussian noise and/or subsample the cloud to simulate lower resolution or further distance from target. Since experiments 1 and 2 require changing these two variables independently, it was decided to nest the loops of the two variables,

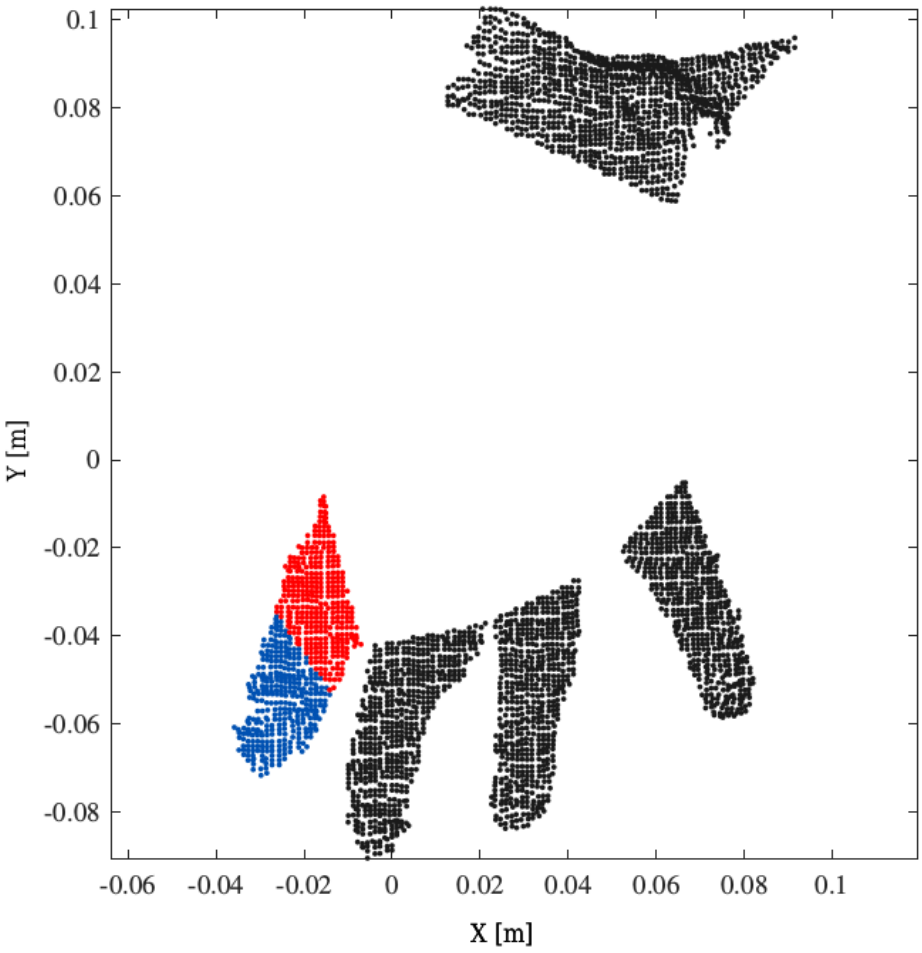


Figure 3.11: Screenshot of sub layer MATLAB segmenting process.

testing the whole 2D space of the combination of them. This provided additional information, which is explained in the Auxiliary Experiments section.

In order to convert the approximately uniformly distributed pseudo random numbers generated by the c library's `rand()` function into a Gaussian distribution, the BoxMuller transform was used [115, 116]. This sampling method for generates pairs of independent, standard and normally distributed random numbers from the source of uniformly distributed random numbers. The algorithm takes in the Standard Deviation in the units desired for the output variable, and a mean or offset in the units desired for the output, which was set to zero. The algorithm

gives a pair of numbers, but one of them is discarded, to allow the function to return one value. It should be noted that the C library's `rand()` function is not truly random, and only provides pseudorandomness [117], however it was decided that this was sufficient for these purposes as a sufficiently accurate representation of the sensor noise in the SoftKinetic. The noise in the SoftKinetic is shown to be primarily Gaussian in Figure 3.12, where the sensor was pointed perpendicularly at a flat plane for 60 seconds to evaluate noise.

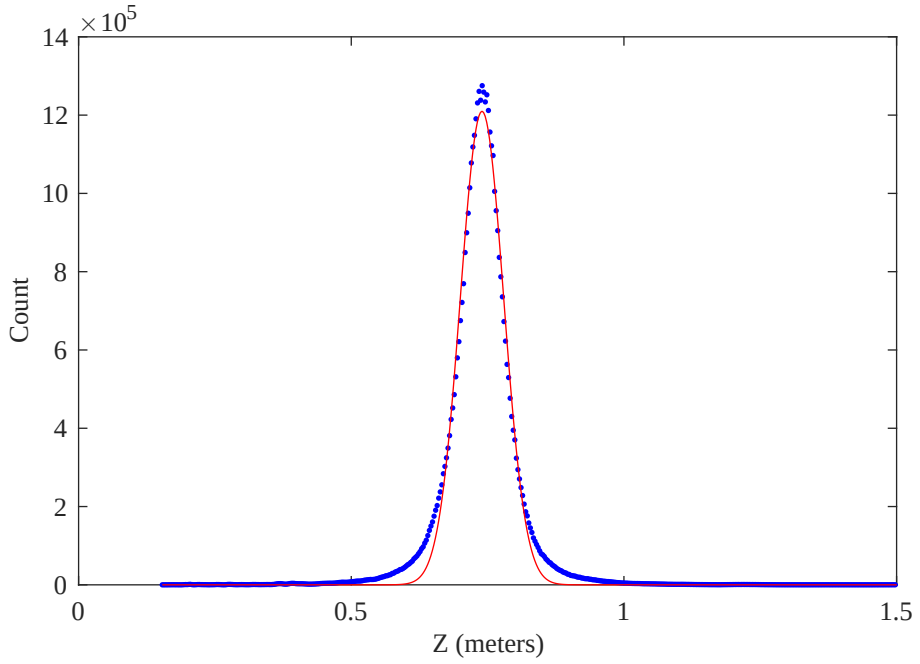


Figure 3.12: SoftKinetic noise is approximately Gaussian. Line shows fit of Gaussian distribution.

To simulate lower resolution, a voxel grid filter was used, which reduces all points within each regularly gridded voxel into a single point as the mean of those points. This allows a regular resolution of the model without relying on a subset of the points.

3.3.5 Evaluation

To evaluate the relative merits of the algorithms, metrics were established. For experiments 1 and 2 the transformation that resulted from the fit was applied to the unaltered cloud, and then run through the evaluation. The time required to calculate the RMSE was not included in the algorithm runtime.

The first metric is Framerate, which is measured in Hz, which is the reciprocal of Runtime, which is measured in Seconds. The Boost library's Posix Time class was used to evaluate the elapsed time of the algorithm in microseconds, which was then converted to seconds. This uses the computer's realtime clock, which is not actually accurate to the microsecond, but should be accurate to the millisecond, which should be appropriate for the durations measured, especially as the average of up to 500 frames will be used. The runtime is a useful metric, and was recorded in the log files. However in post processing, the reciprocal of the runtime was taken, to be used as framerate. This was done to allow a more human readable and understandable value for a real-time algorithm, as well as to allow easier comparisons to the literature, as well as comparisons to the capture rates described in the Background section.

The second metric is Root Mean Squared Error, or RMSE, which is a standard metric of fits. This was chosen in lieu of other metrics such as R^2 because it provides a value in real world units, and is not skewed by the number of points in the sample. The Root Mean Squared Error also emphasizes large outliers, which is useful for showing a poor fit. The RMSE was calculated by taking the normal distance to the plane defined by the nearest point and its normal. This avoids subsampling errors at the expense of slower runtime, however as the RMSE was calculated outside of the runtime there was no downside.

3.4 Experiments

Exp#	Description	Algorithms
1	Reduced Resolution	Affine ICP, Affine FPFH, Nested ICP, Nested FPFH
2	Gaussian Noise	Affine ICP, Affine FPFH, Nested ICP, Nested FPFH
3	Time of Flight Camera	Affine ICP, Affine FPFH, Nested ICP, Nested FPFH
A	GPU Acceleration	Affine ICP, Affine ICP GPU, Nested ICP, Nested ICP GPU
B	Reverse Matching	Nested ICP GPU, Nested ICP GPU Reverse
C	Algorithm Complexity	Affine ICP, Affine FPFH, Nested ICP, Nested FPFH
D	Size/Noise Sweep	Affine ICP, Affine FPFH, Nested ICP, Nested FPFH
E	TOF Through Endoscope	N/A

Table 3.2: Experiment Dependant Variables.

For each experiment, each algorithm in Table 3.2 was run on the same data. The algorithms were not run in real time, however, since the framerate is dependent upon the hardware used, and would have provided fewer data points for the slower algorithms. Therefore the algorithms were allowed as much time as needed to process each frame, and the time required was recorded.

3.4.1 Experiment 1: Reduced Resolution

Experiment 1 will be to take a high quality scan of a silicone hand. The high quality scan will come from the Artek scanner to provide an accurate ground truth. The high quality scan will be used for the model, manually segmented. The pixel size will be incrementally increased to determine algorithm robustness to lowered resolution or distance from target. The metrics evaluated will be RMSE and Framerate. The algorithms will be compared to a control of no matching, keeping the initial guess. The best algorithm will be considered the one with the lowest RMSE and the highest Framerate.

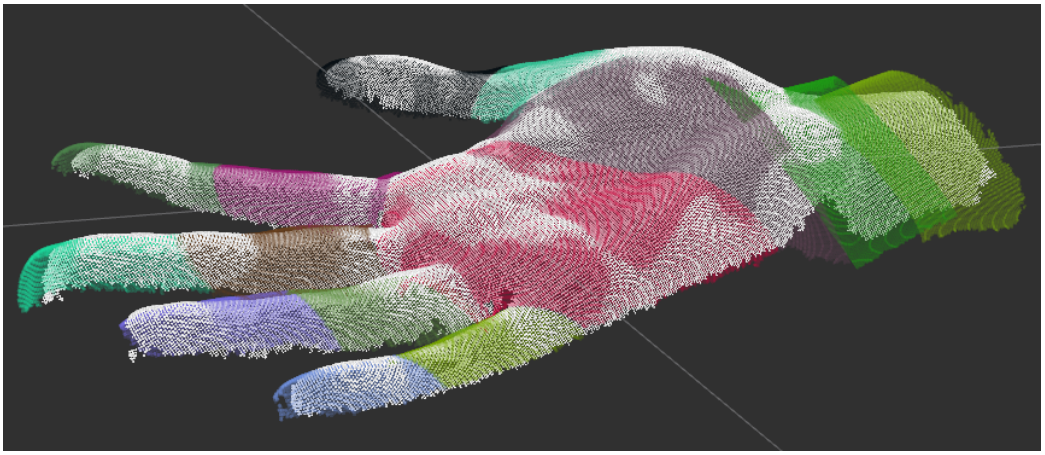


Figure 3.13: Example of scan and fitted model from Experiment 1 using nested iterative closest point.

3.4.2 Experiment 2: Gaussian Noise

Experiment 2 will be to take a high quality scan of a silicone hand. The high quality scan will come from the artek scanner to provide an accurate ground truth. The high quality scan will be used for the model, manually segmented. Gaussian noise will be incrementally added to determine algorithm robustness to sensor accuracy or external noise such as smoke or poor lighting. The metrics evaluated will be RMSE and Framerate. The algorithms will be compared to a control of no matching, keeping the initial guess. The best algorithm will be considered the one with the lowest RMSE and the highest Framerate.

3.4.3 Experiment 3: Time of Flight Camera

Experiment 3, which is the core experiment, will be to determine the accuracy of the algorithms with a realtime sensor. The same silicone hand will be used from Experiments 1 and 2. The same manually segmented model from Experiments 1 and 2 will be used, however the actual scan from a SoftKinetic DS325 time of flight scanner will be used to evaluate algorithms in a real-world setup as a surrogate for a time of flight used through an endoscope. The scans were recorded to disk to allow the same sensor stream to be run against each algorithm, as well as to allow the algorithms to run on each frame regardless of the algorithm speed. The metrics evaluated will be RMSE and Framerate. The algorithms will be compared to a control of no matching, which keeps the initial guess. The best algorithm will be considered the one with the lowest RMSE and the highest Framerate.

3.5 Auxiliary Experiments

In addition to the three primary experiments carried out as detailed above, additional experiments were performed due to interesting results found during the course of the project.

3.5.1 Experiment A: GPU Acceleration

Although implementation of rigid iterative closest point has already been shown, for example by [113], it was desired to implement it for the affine algorithm, as well as the nested rigid algorithm, in order to show that the algorithms are capable of being sped up through parallelization. Therefore, GPU implementations of the Affine ICP and Nested Rigid ICP were implemented, to show that they are capable of running in real-time. The FPFH algorithm could have benefited from GPU, but since parts of it were already running multi-threaded with OpenMP, and since preliminary results showed vastly inferior accuracy, it was decided not to implement them on the GPU. The metrics evaluated will be RMSE and Framerate. The algorithms will be compared to a control of the CPU version of ICP. The hypothesis is that RMSE will stay approximately the same, and the Framerate will increase by an order of magnitude.

3.5.2 Experiment B: Reverse Matching

A choice was made in terms of which cloud would have the KD-Tree made of it. This required a trade off. The standard option, or forward-matching as it will be called here, is to make a KD-Tree of the scanned cloud upon first collection. Then the algorithm will search through the model, using that tree to find the closest point in the scanned cloud. This required a KD-Tree the size of the scan.

The second option, or reverse-matching as it will be called here, is to make a KD-Tree of each layer of the model. The algorithm will search through the scan, using that tree to find the closest point in the current layer of the model. This requires more KD-Trees, but each can be smaller than in the forward-matching method, especially as the lower levels of the model are reached, leading to much faster searches, at the expense of more tree building.

This test was only implemented on the GPU version of the nested ICP algorithm, so the experiment will be to compare the GPU Nested ICP to the Reverse GPU Nested ICP algorithm. The metrics evaluated will be RMSE and Framerate. The algorithms will be compared to a control of the CPU version of ICP. The hypothesis is that RMSE will stay the same, and the Framerate will increase slightly for the Reverse version.

3.5.3 Experiment C: Algorithm Complexity

The theoretical algorithmic complexity of the algorithms are at least $\mathcal{O}(n)$, possibly up to $\mathcal{O}(n \times \log(n))$ due to the requirements of search. Experiment 1 allows an indirect testing of this theoretical complexity due to the fact that the number of points is changed by the downsample voxel size. Since the scan is a 3D surface, the downsampling should reduce the number of points inversely proportional to the voxel size squared. The hypothesis is that the rate of the algorithms will increase proportionally to the square of the voxel size.

3.5.4 Experiment D: Size/Noise Sweep

The combination of experiments 1 and 2 allowed a more complete analysis of the two variables, resolution and added noise, by testing all permutations of the two variables. For example, testing a combination of high noise and low resolution. A subset of this data is presented for experiments 1 and 2, however all data was recorded and is presented as an auxiliary experiment.

3.5.5 Experiment E: Time of Flight through Endoscope

Although the functionality and accuracy of time of flight distance sensing through an endoscope has been proven elsewhere [44] it was desired to perform an independent test of the feasibility of doing so. Therefore a full analysis of time of flight through an endoscope was not required, but a test was designed to show the successful determination of depth with the commercially available DS325 depth camera.

The first thing to test was transmission of the laser pulse through the endoscope's optical fibers. The time of flight camera only needs a pulsed flash to be visible, with no structure required, so therefore the laser light being transmitted through the optical fibers should be functionally identical to the laser light being transmitted in open air, other than the change in speed due to traveling through glass, traveling at approximately two thirds of the speed through the air due to index of refraction difference, causing objects to seem further away. The experiment was determined to be to channel the laser pulse from the laser through the fibers of an endoscope, and block as much as possible from going anywhere else. The pulse would be observed through the air, to only test transmission through the laser. The evaluation of success would be to look at the confidence/intensity image and determine if the light was in fact emerging from the end of the optical fibers.

The second test was receiving the signal through the rigid optics of the endoscope. To test this the laser pulse was allowed to emanate through an alternate optical fiber bundle, a SCHOTT 250-101 Flexible Light Guide (SCHOTT Lighting and Imaging, Mainz, Germany), since the input port to the endoscope's optical fibers is not aligned to the input of the rigid optics. The

return signal would be viewed through the endoscope's rigid optics. Since the endoscope requires a quite narrow field of view, the resultant image would take up a small portion of the image without corrective optics, but this was determined to be adequate to evaluate the functionality. The endoscope would be pointed at an inclined plane, and evaluation would be to look for a gradient in the raw depth image. Actual depth values would be ignored, since the index of refraction for the glass lenses will significantly alter the speed of light, and a new calibration would have to be performed if the system were to be fully implemented, as in [44]. The da Vinci endoscope was used because the placement of the laser diode was so close to the depth camera that a regular endoscope image was washed out by bleed-through. However, the da Vinci endoscope has two channels for stereoscopic vision, so the interface is much closer to the edge. The fiber optics used to transmit the laser diode light were not actually the fiber optics contained in the endoscope, since the relative placement of the endoscope's fiber bundle was not conducive to the setup. Therefore an off the shelf 'light pipe' for directing high power inspection lights was used in parallel to the endoscope to carry the modulated laser diode light.

Chapter 4

Results

4.1 Primary Results

4.1.1 Experiment 1: Reduced Resolution

This experiment was designed to determine the speed and accuracy trade-off for each algorithm, by reducing the resolution of the scan to speed up the algorithm. The average framerate for each algorithm is shown in Table 4.1, where the affine ICP shows the highest framerate value.

	ICP	FPFH
Global Affine Transformation	1.09	0.15
Local Rigid Transformation	0.38	0.01

Table 4.1: Experiment 1: Average Framerate [Hz].

The Rate for each algorithm is plotted in Figure 4.1.

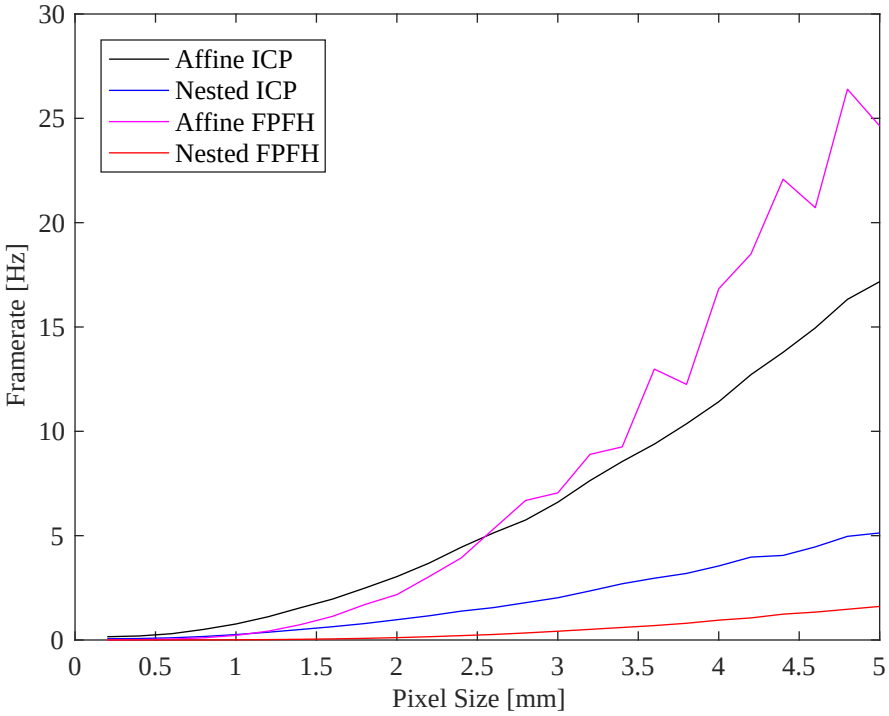


Figure 4.1: Experiment 1 Framerate Results Summary.

The accuracy of each algorithm was also recorded for each algorithm at the same range of pixel sizes. The Root Mean Squared Error for each algorithm is plotted in Figure 4.2. The ICP algorithms do not show a noticeable decrease in accuracy until above 3mm pixel size, and even then it is slight. The FPFH algorithms both converge to the a line of approximately 7.5mm RMSE at around 2mm pixel size.

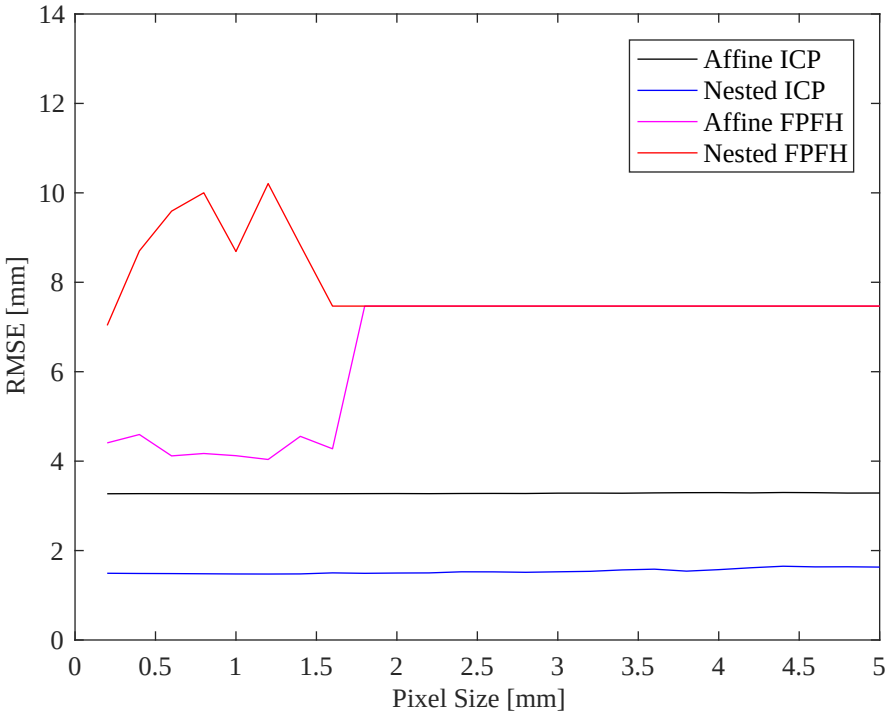


Figure 4.2: Experiment 1 RMSE Results Summary

4.1.2 Experiment 2: Gaussian Noise

This experiment was designed to determine algorithm robustness to sensor accuracy or external noise such as smoke or poor lighting, by adding Gaussian noise to the sensor readings for each algorithm. The average RMSE for each algorithm is shown in Table 4.2, where the nested ICP shows the lowest RMSE value.

	ICP	FPFH
Global Affine Transformation	3.6	12.7
Local Rigid Transformation	2.0	16.0

Table 4.2: Experiment 2: Average RMSE [mm].

The rate of the algorithms can be seen in Figure 4.3, where there is not a significant change

in the rate with respect to the standard deviation of the noise.

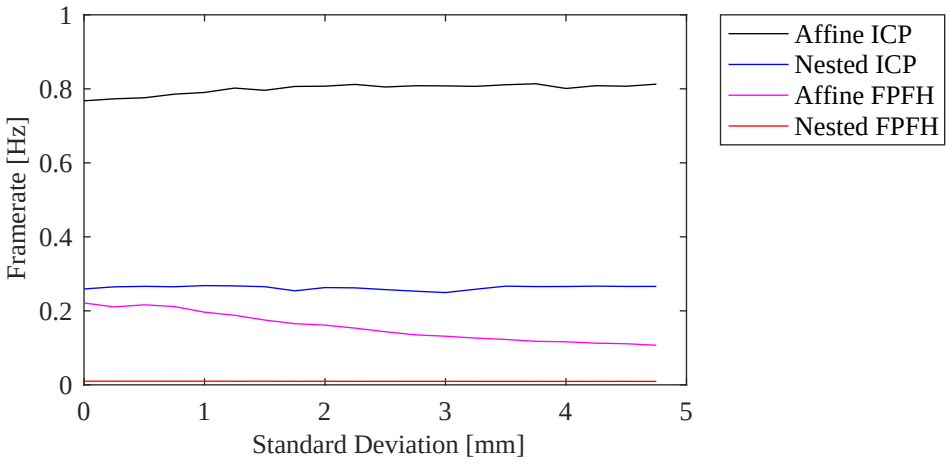


Figure 4.3: Experiment 2 Framerate Results

Since the standard deviation has little effect on the rate, as well as the wide variation between the rates of the different algorithms, it is more instructive to look at Figure 4.4, which shows the mean rate for each algorithm. This shows the significant impact that the nested strategy imposes upon the rate, although this could be made up for with differences in accuracy.

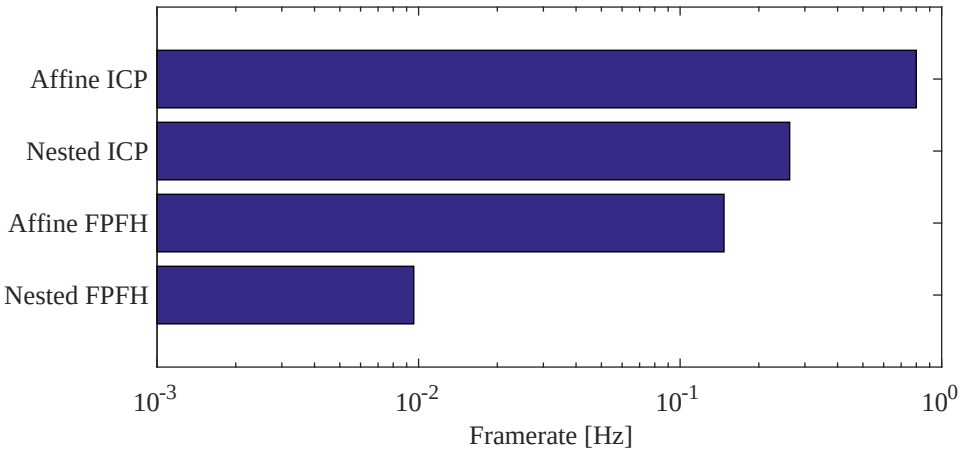


Figure 4.4: Experiment 2 Framerate Average

The accuracy of each algorithm was also recorded for each algorithm at the same range of

standard deviation of added noise. The Root Mean Squared Error for each algorithm is plotted in Figure 4.5. Both of the FPFH algorithms performed significantly worse than 7.5mm, which is where a control algorithm using only the prior fit would be, showing that while they converged to a solution they did not provide a reasonable fit to the model. The Affine version of the FPFH algorithm did show convergence at some points which provided a slight benefit compared to control, but it never outperformed any of the ICP algorithms. The nested versions of the ICP algorithms showed a significant benefit in terms of RMSE, but did come at a speed cost.

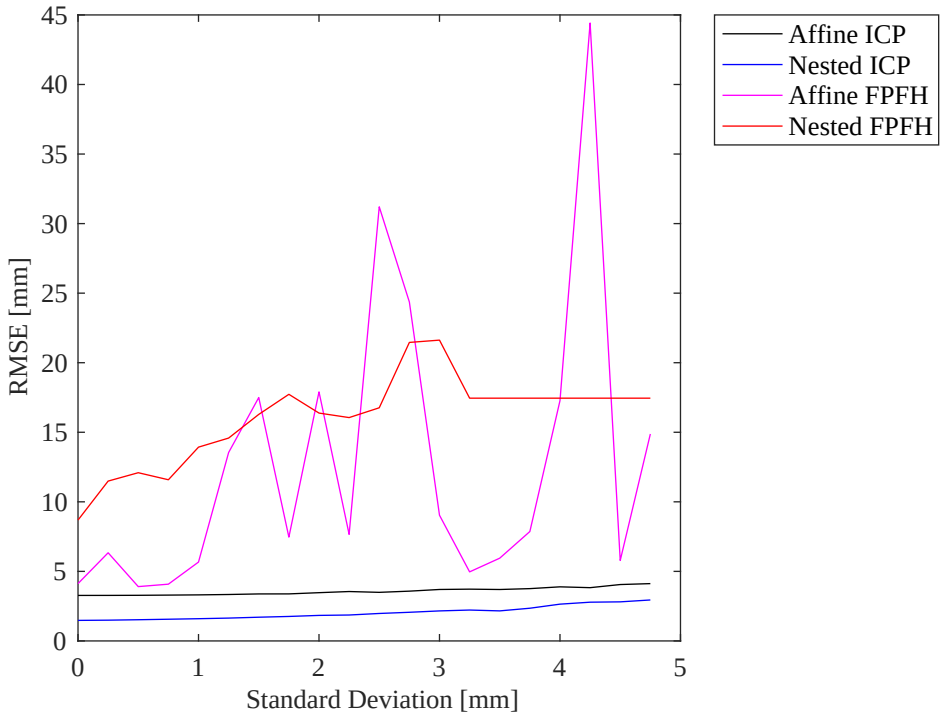


Figure 4.5: Experiment 2 RMSE Results with variable added noise

4.1.3 Experiment 3: Time of Flight Camera

This experiment was designed to test the real-world limits of the algorithms using a commercial off the shelf sensor to measure the actual object. The rate of the algorithms can be seen in Figure 4.6. This shows roughly the same relationship as in Figure 4.4, however the FPFH algorithms failed to converge, increasing their apparent speed at the expense of accuracy.

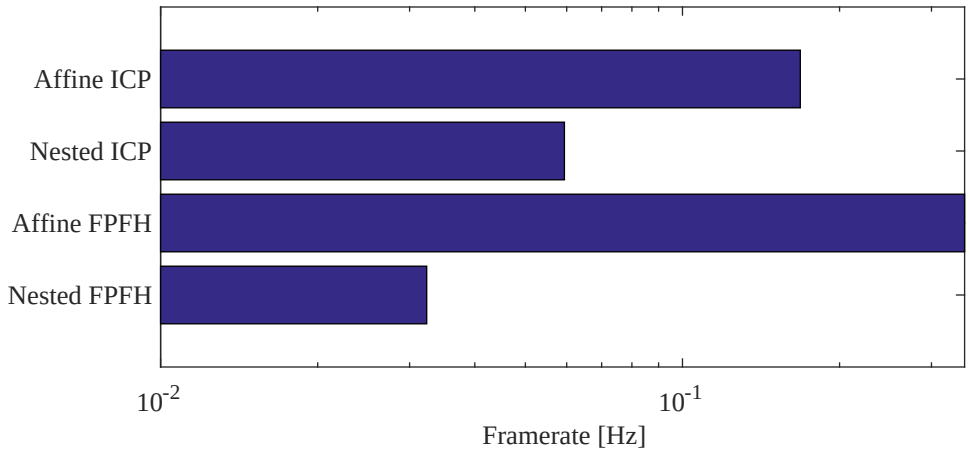


Figure 4.6: Experiment 3 Rate Results Summary

The accuracy of the algorithms was broadly the same as in the first two experiments, except that the real world data did not provide as accurate of an initial guess, leading to a more skewed RMSE for the FPFH data that did not converge. The nested ICP algorithms again provided an improvement in accuracy over the non nested variants.

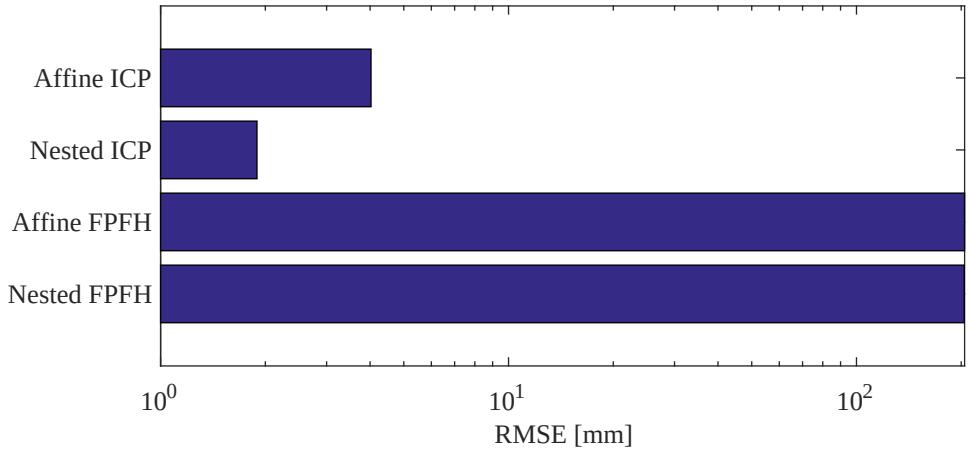


Figure 4.7: Experiment 3 RMSE Results Summary

The time series of Experiment 3 provides some interesting insights, and is shown in Figure 4.8. This shows the changes in the algorithms around the 200 frame mark, where the target

began moving. It also shows the point near the end, starting around the 400 frame mark, where the target becomes occluded. This shows that the nested algorithms are more robust to the partial occlusion at the beginning but still fail to compensate once the hand is fully occluded at the end.

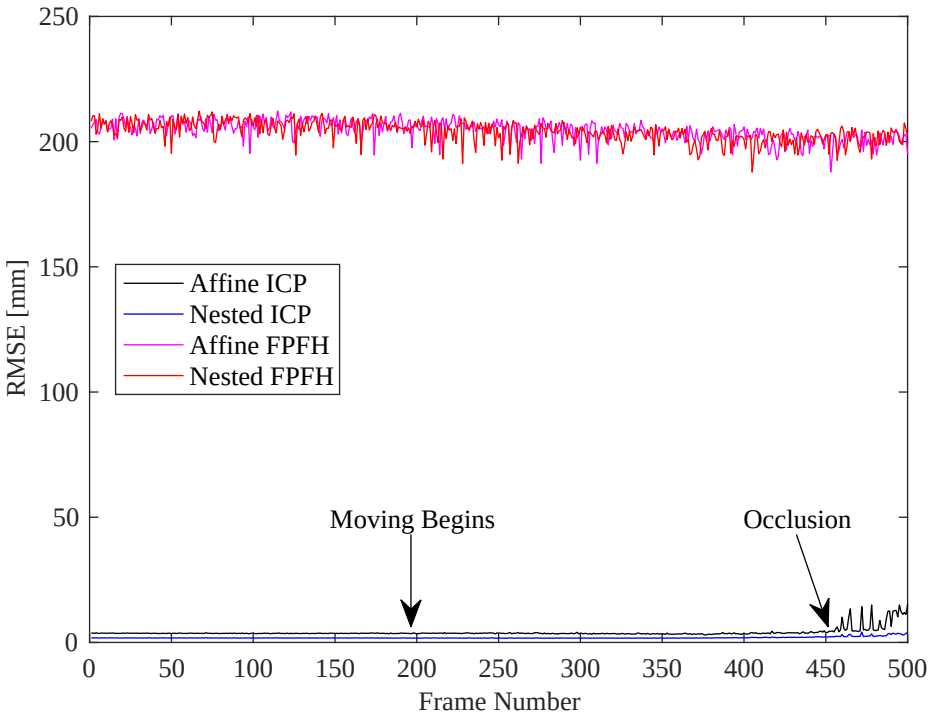


Figure 4.8: Experiment 3 RMSE Time Series

4.2 Secondary Results

4.2.1 Experiment A: GPU Acceleration

The benefits of GPU computing are clearly visible in Figure 4.9, where essentially the same algorithm shows an order of magnitude speedup by running on the GPU as opposed to running on a single core of the CPU.

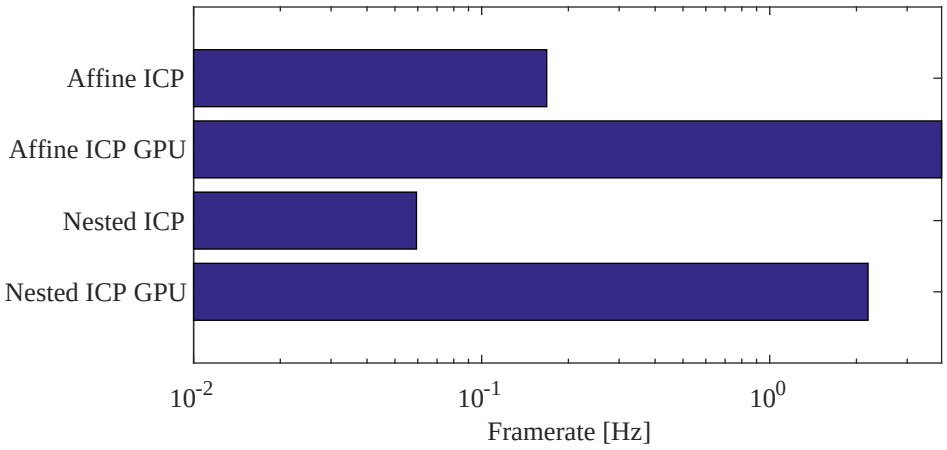


Figure 4.9: GPU Rate Comparison

The GPU algorithms did however show a slightly different RMSE than the equivalent CPU versions of the algorithm in Figure 4.10, showing the small differences in the coding of the algorithms, most likely reflecting the differences in the matching algorithms.

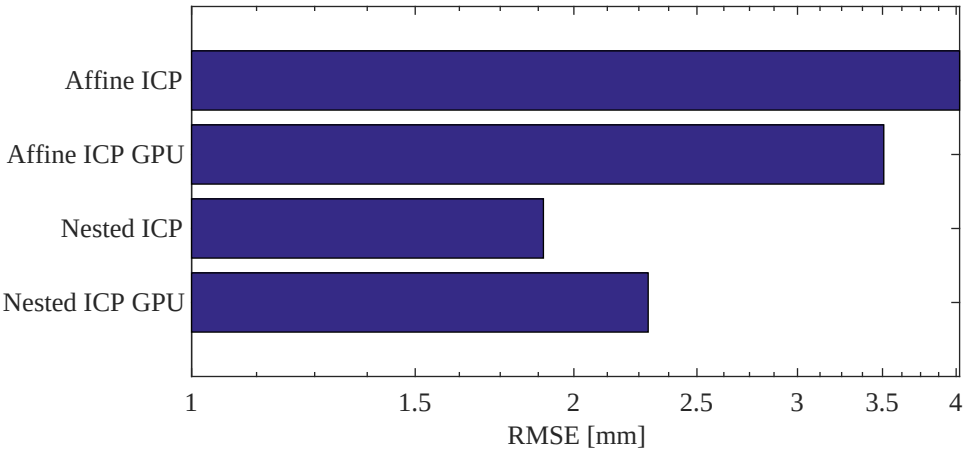


Figure 4.10: GPU RMSE Comparison

4.2.2 Experiment B: Reverse Matching

The reverse matching did not show a speed improvement over the forward matching, in fact as shown in Figure 4.11 the reverse algorithm ran significantly slower than the standard matching algorithm, for the Nested GPU variant. Also, the reverse GPU nested does not have any measurable difference to the forward nested GPU algorithm, since the direction of fit only changes the time taken, and does not change the solution found, therefore the RMSE was the same for the forward and reverse matching algorithms.

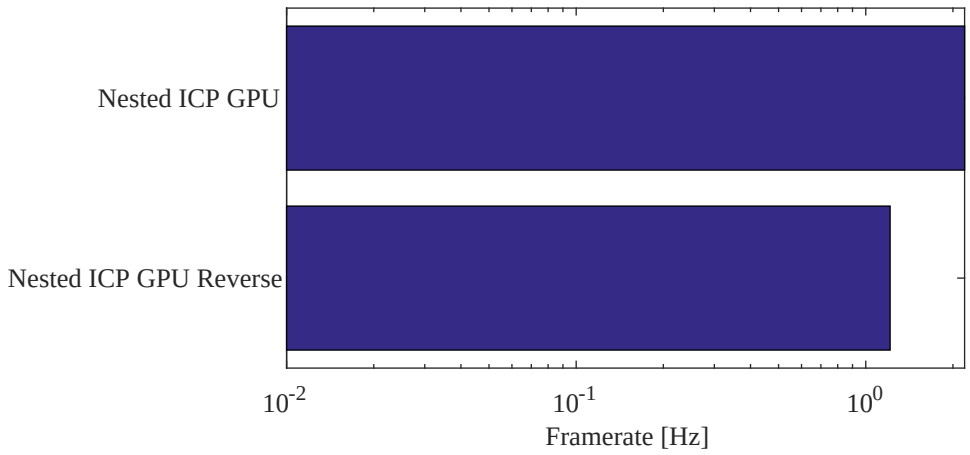


Figure 4.11: Reverse Match Framerate Comparison

4.2.3 Experiment C: Algorithm Complexity

The rate of the algorithms can be seen in Figure 4.12, where there is an increase in rate as the pixel size increases. This is because the algorithms are mostly $\mathcal{O}(n)$ where $n \propto 1/(PixelSize)^2$ so one would expect the lines in Figure 4.1 to increase $Rate \propto (PixelSize)^2$.

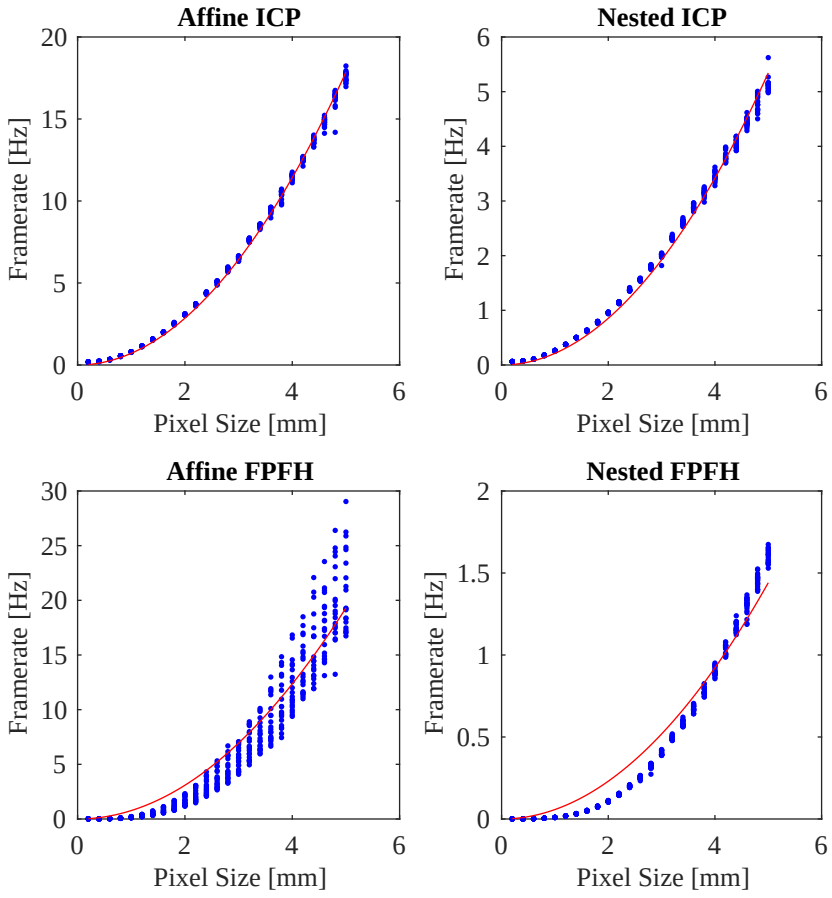


Figure 4.12: $\mathcal{O}(n)$ Complexity Validation. The dots represent data from Experiment 1, and the lines are a least squares fit assuming $\mathcal{O}(n)$

4.2.4 Experiment D: Size/Noise Sweep

In the interest of completeness, a combination of experiments 1 and 2 was performed where all permutations of the two variables were tested. As can be seen in Figure 4.13, for the case of the CPU ICP algorithm, no significant variations were seen in the full 2D sweep, with the metrics changing smoothly. The time metric appears constant across the RMSE dimension of the sweep.

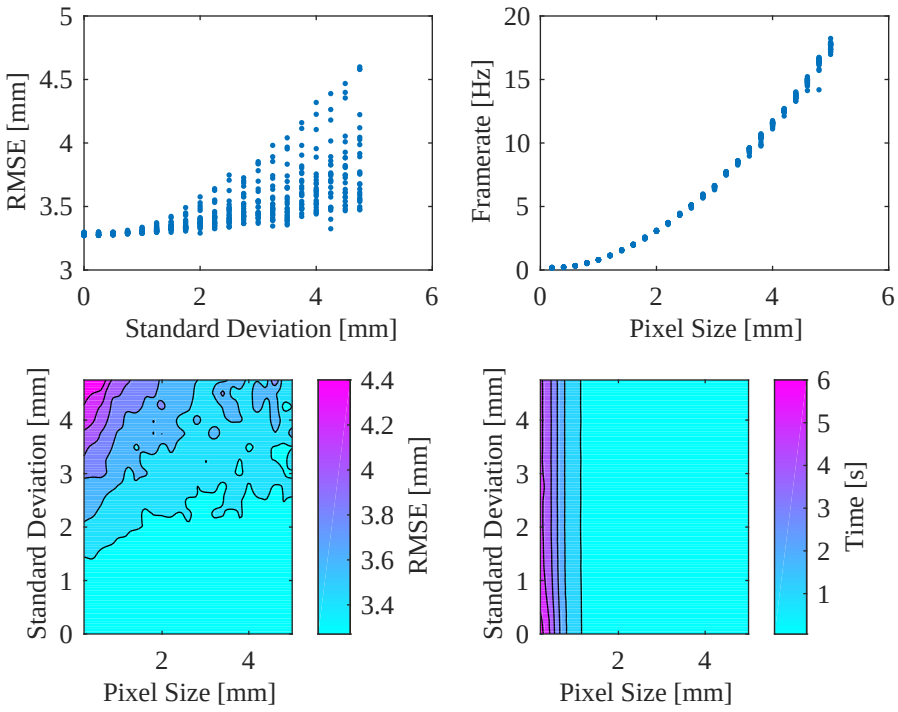


Figure 4.13: ICP Noise Resolution Study

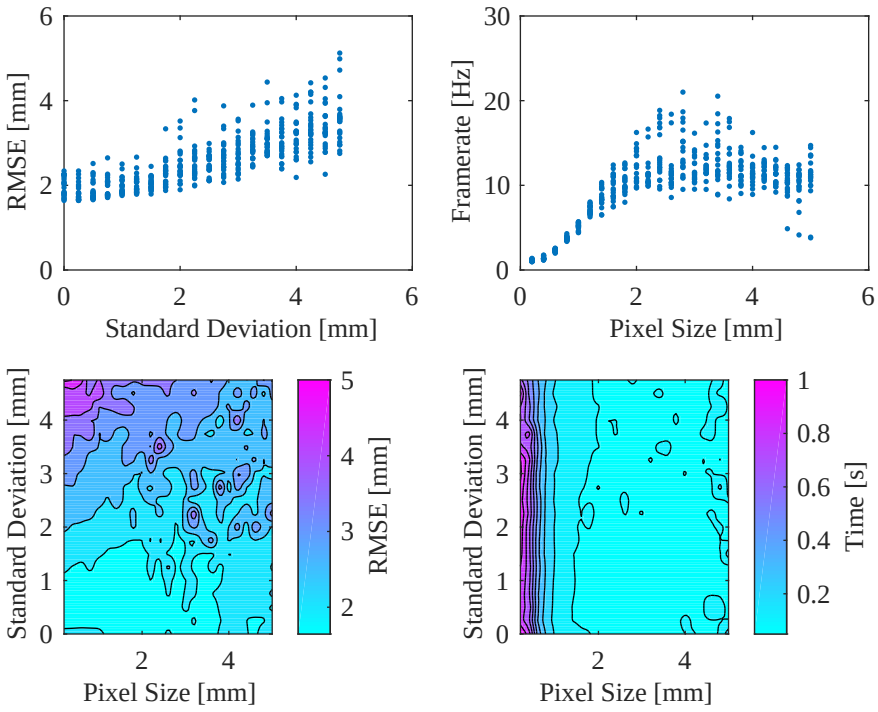


Figure 4.14: Nested GPU ICP Noise Resolution Study (Note difference in axis and colormap scales compared to Figure 4.13)

4.2.5 Experiment E: Time of Flight through Endoscope

The first test was to see if the modulated laser light could be sent through fiber optic lines. The modulated laser light was transmitted through the optical fibers, and the resulting glow was detected in the 'confidence map,' which shows the intensity of the light to be within the normal operating range, and the 'depth map,' which shows the depth being measured, as is shown in Fig.4.15.

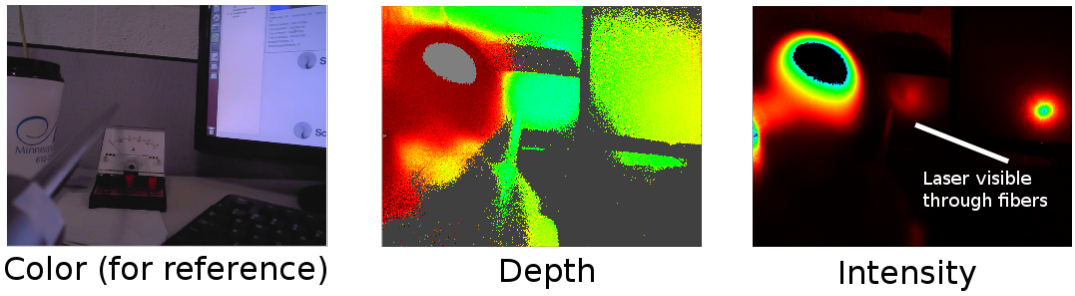


Figure 4.15: View of laser diode light being transmitted through endoscope optical fibers

The results of the test show the proper distances measured to within a few centimeters. The target plane was placed at an angle, so that the relative depth measurements at different points could be discerned, as is shown in Fig.4.16. The color gradient corresponds to different distances, showing that the endoscope optics and the optical fibers did not inhibit the system's ability to measure distances by time of flight. The standard optics of the depth camera were used, which led to a much larger field of view than was necessary, which is visible in the figure as the small circle in the center of the image.

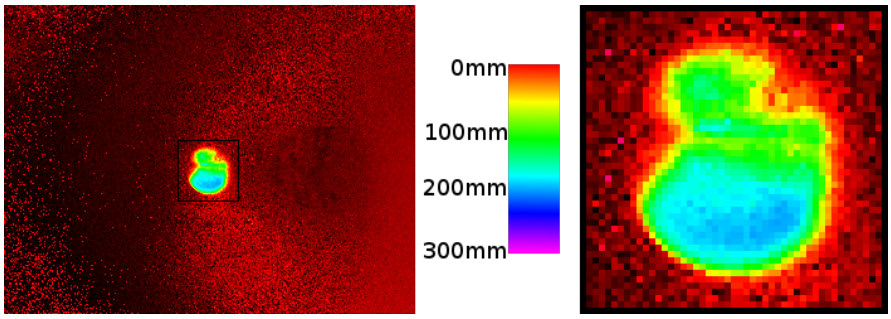


Figure 4.16: View of laser diode light being transmitted through endoscope optics and optical fibers. Color gradient corresponds to inclined plane being observed

Chapter 5

Discussion

5.1 Primary Experiment Analysis

All algorithms from all experiments can be seen in Figure 5.1. The lower left hand corner represents the ideal algorithm, while the upper and rightmost points represent the worst performing algorithms. Visually, there is a clear left-right separation between the ICP algorithms and the FPFH algorithms, where all ICP algorithms outperformed all FPFH algorithms in RMSE. In Runtime it is less clear, with the FPFH algorithms not always slower than the ICP algorithms. The algorithm that performed the best in terms of speed was the Affine ICP. The algorithm that performed the best in terms of accuracy was the Nested ICP. The margins in each metric are within an order of magnitude of each other, so therefore there is no clear winner, as the algorithm will be chosen based on the application. However, due to the unrelenting force of Moore's Law [118, 119], the accuracy metric could be given priority, as the calculations involved are embarrassingly parallel, so sufficient speed only requires sufficient computing power. Also the nested ICP can provided better semantic labeling, leading to an improved situational awareness and better knowledge of the boundary areas between semantically labeled sections.

The Fast Point Feature Histograms were inferior to ICP in every contest. In many cases, there were not enough reliable features in the data to provide a match, and although this provided a speed boost to the FPFH algorithm, it performed badly in RMSE. This is likely due in some part to the human shapes being used being feature poor. Additionally, the low quality and resolution of the real-time scanner and the reduced resolution and increased noise rapidly made the already poorly performing algorithm fail, as can be seen clearly in Figure 4.5, where below 1.5mm pixel size the FPFH algorithm is at least improving on the control, but still not doing as well as ICP at a slower rate. Beyond 1.5mm, insufficient features were found and

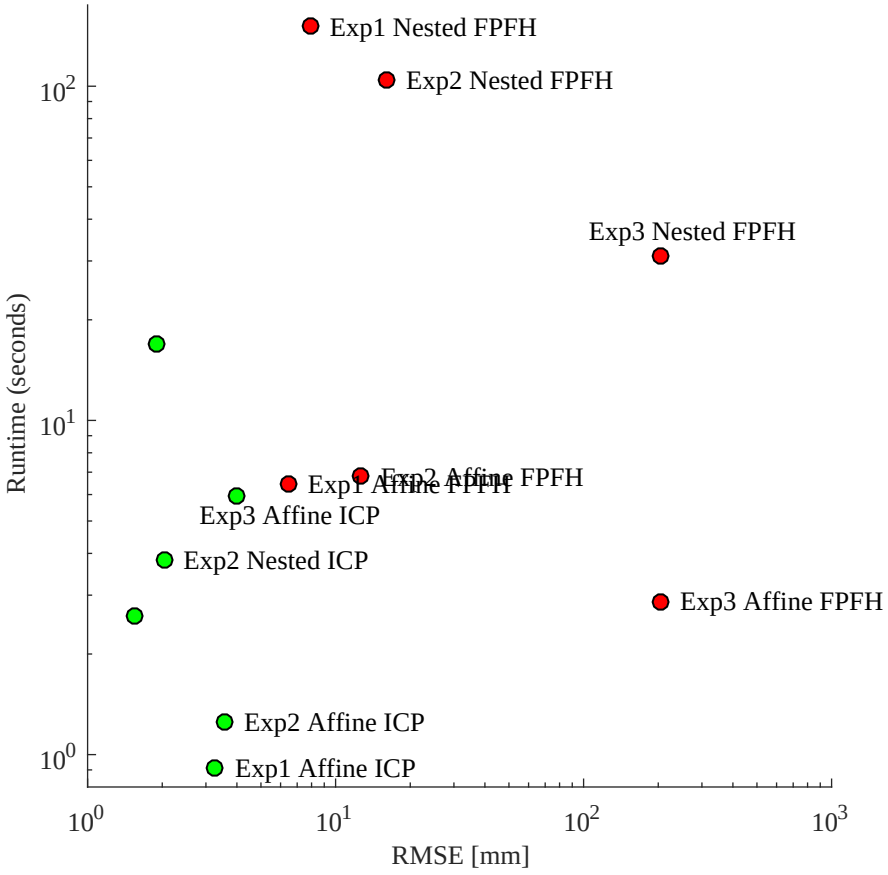


Figure 5.1: Exp1: Reduced Resolution; Exp2: Gaussian Noise; Exp3: Time of Flight Camera

the default transformation remains the one used, which is equivalent to an RMSE of 7.5mm. Therefore the points above 8mm RMSE for the nested FPFH actually show a worse fit than a control with no fit whatsoever. Insufficient features are even more of a problem in the nested variant, as each nesting level takes a subset of the model, linearly reducing the number of features available. Therefore this algorithm is not recommended for nonrigid tracking of human anatomy with current hardware. Future time-of-flight scanners may be able to create more feature rich models, and computing power may improve to allow real-time use of this algorithm, but at the current time the algorithm fails to provide any benefit. Also there simply may not be enough geometric features in human anatomy to facilitate tracking based on features alone. Future work

will likely need to include high resolution RGB textures to extract additional features from the color space to be combined with geometric data.

Semantic labeling provides significant advantages in terms of providing context for performing desired actions safely and smoothly. While the Affine versions of the algorithms could be used to provide some semantic labeling, the nested rigid transformations provided a better labeling of the different preregistered regions. It also provided a more clear transition area, including showing whether the transition area between two different labeled regions was an overlapping transition, or whether there was an unidentified area between the regions. This could provide additional confidence feedback, for example too much overlap or too much unidentified area between the regions could be interpreted as a sign that the model was failing to adequately describe the scene.

The nested rigid transformation, as implemented here, provided no limits to drift of lower level nested regions. At its most extreme, there were cases during testing where a finger match jumped over to be fit to an adjacent finger. This was rare, due to the use of a prior, but could be further avoided by providing a penalty for translating or rotating too far beyond the parent. This could be implemented as a hard stop or as a scaling cost to be incorporated along with the other fit parameters.

5.2 Secondary Analysis

5.2.1 GPU Acceleration

For the point cloud sizes used, GPU computing provides a clear advantage with minimal downsides. It provides an order of magnitude increase in framerate, for example going from $0.8Hz$ to $11Hz$ in Experiment 2 Figure 4.4, while providing essentially the same accuracy, other than differences due to the programming of the two algorithms. The only cautionary note, is that the overhead can quickly dominate, diluting the gains from the GPU computing. This includes overhead which could be parallelized, such as preparing the point cloud or testing the matches, as well as fixed overhead that is inherent to the system, such as the overhead of transferring the data to and from the GPU, and waiting for stray cores to finish. These diminishing returns can clearly be seen in Figure 4.14 (top right) where the code is unable to run faster than $20Hz$. GPU computing requires more work and is difficult to debug, but it is clearly worth pursuing in this scenario.

5.2.2 Reverse Matching

The reverse matching scheme was not successful at speeding up the algorithm, and had no noticeable effect on fit quality, therefore it is not recommended that this method be pursued further unless orders of magnitude changes in cloud size are encountered, at which time it may be worth re-evaluating this method.

5.2.3 Algorithm Complexity

The theoretical algorithmic complexity of the algorithms was broadly confirmed by the experiments as shown in Figure 4.12. The fits were not perfect, but were close enough to imply that the order of the algorithmic complexity theoretically calculated was true in practice. For the GPU algorithms, the constant time portions of the algorithm rapidly dominated, not allowing the code to run faster than approximately $20Hz$, implying that the constant portions of the code have significant room for improvement.

5.2.4 Size/Noise Sweep

The complete sweep of resolution and noise did not provide much further insight when compared to the experiments as run separately, as they showed relatively predictable and linear extrapolations from the experimental data. The additional time and effort to run the complete sweep was minor, and showed that there were no surprises in the rest of the space validating the choices of the experiment 1 and 2 subsets used.

5.2.5 TOF Through Endoscope

The test of the functionality of a time of flight camera through an endoscope provided a good initial validation of the idea of using a time of flight camera through an endoscope, which has been more deeply investigated elsewhere [45, 44].

Chapter 6

Smart Skin

This chapter is a reproduction of an article submitted to the IEEE Transactions on Robotics journal. As such it was designed to be able to stand alone, and contains its own abstract, background, results, and interpretation. The relevance of this work to the wider thesis goal and body is discussed in the Chapter 8 Conclusion.

6.1 Abstract

Safe, intuitive human-robot interaction requires robots to intelligently interface with their environments. We introduce a stretchable smart skin sensor that provides this function. Stretchability allows it to conform to nearly arbitrary surfaces. It senses contact over nearly the entire surface, localizes contact position with sub-centimeter accuracy, and provides an estimate of the contact force. Our approach exclusively employs stretchable, flexible materials resulting in accurate position sensing when stretched up to 133%. We exploit novel carbon nanotube elastomers to create a two-dimensional potentiometer surface. Finite element simulations validate two independent simplified polynomial surface models to enable real-time processing on a basic microcontroller with no supporting electronics. The skin can be scaled up to arbitrary sizes with only five electrodes. We designed, implemented, calibrated, and tested a prototype smart skin as a tactile sensor on a robot for sensing unexpected physical interactions. We experimentally demonstrate its utility in collaborative robotic applications by showing its potential to enable safer, more intuitive human-robot interaction.

6.2 Introduction

Robotic systems in industrial, home, and medical applications have increasingly required the ability to work safely in collaborative environments with humans. In order to safely work around humans, multiple sensing modalities and joint configurations have been utilized with varying levels of success [120]. The primary need in this field is an affordable, modular sensing system—such as a skin with minimal wiring—that could be applied to any robotic link regardless of geometry and material. Ideally, like human skin, this system should detect physical contact, localize its position on the link, and assess the magnitude of the force [121].

By covering each link of a robotic arm with a tactile sensing skin, robots could gain increased awareness of their physical interactions with unpredictable environments or human activity. This would enable humans and robots to collaborate more safely and intuitively in a variety of situations. For example, in robotic surgery, human arms and robotic manipulators must compete for access to a highly constrained workspace. A surgeon and scrub nurse can communicate the need for space through touch by letting their arms collide and remain in contact. Unlike a traditional robot, the surgeon can intuitively push and the nurse will temporarily move away to allow access without the need for explicit verbal communication.

Several industrial robots have utilized 'soft' joint configurations to avoid harmful collisions. The Baxter Robot (Rethink Robotics, Boston, MA) employs motor-driven spring joints that require added complexity and specialized designs. These specialized joints also complicate controls algorithms. Other methods have included computer vision algorithms for collision avoidance, however computer vision algorithms typically suffer from line-of-sight and accuracy limitations.

Prior art has also considered smart skins: flexible materials embedded with force or position sensors that can be molded to different geometries. Several attempts at smart skins have utilized arrays of rigid force sensors embedded in flexible media such as a flexible printed circuit board or polydimethylsiloxane (PDMS) silicone. Force sensors include standard strain gauges [122], piezoresistive sensors [123], capacitive force sensors [124], and polyvinylidene fluoride (PVDF) sensors [125]. The issue with the use of rigid sensors in flexible media is that the device is still not stretchable. A second issue with sensor arrays is that contact position and force data are limited to tactile pixels (taxels): finite locations separated by deadzones. Even as sensors get smaller, sensing is limited to discrete positions in the 2D plane. Moreover there is a linear increase in the number of electrodes ($2n$) for increases in sensing area ($n \times n$), even if multiplexed.

Given the shortcomings in rigid sensor arrays, research groups have instead focused on developing flexible sensor pads for smart skin sensing applications. Choi et al. designed a tactile sensor based on $0.5mm^2$ PVDF sensor arrays embedded in polyester film [126]. While these small PVDF sensors allow more flexibility than the larger sensor arrays, they still suffer from large numbers of electrodes. Papakostas et al. also developed a flexible tactile sensor comprised

of polyester sheets embedded with a grid of silver-polymer conductive traces [127]. The grid contacts form a piezoresisitive force sensor. The grid of conductive traces allows for flexibility in both the sensors and medium. However, this solution still requires n^2 electrodes for an $n \times n$ sensor array. Park et al. developed an interlocked micro-structure skin formed from PDMS designed to mimic human-skin [128]. This approach also provides an estimate of shear force via resistance sensing.

Another novel approach has been the use conductive liquids (Eutectic Gallium-Indium) embedded in microchannels within PDMS [129]. The use of conductive liquids does permit truly stretchable materials. In this approach the sensing apparatus required specific geometric designs and therefore was limited to discrete (non-continuous) sensing pads. In order to cover the smoothly curving, swept geometry of robotic arms such as the Kuka LBR (KUKA Roboter GmbH, Augsburg, Germany) or the Baxter, a smart skin needs to be both flexible and stretchable. A stretchable two-dimensional skin that conforms to arbitrary geometries is also easier to manufacture than 3-dimensional skins specifically made to fit a particular robot geometry.

Pugach et al. have proposed a method involving conductive rubber sheets created by mixing carbon into the material [130]. Using electrical impedance tomography, a series of many electrodes situated around the perimeter of the conductive surface are used to sense and localize tactile force. While this group was able to achieve successful calibration and sensing using this method, this design suffers drawbacks such as the requirement that forces be applied by conductive materials which is not the case in typical collaborative environments. This approach does not scale well with large areas while maintaining spatial and temporal resolution. The electrodes situated around the perimeter require the number of interconnected wires to scale by $4n$ for n^2 discrete taxels. Additionally it is unclear whether this design can endure the necessary stretching for conforming to varied robotic link geometry.

Lacasse et al. demonstrated a flexible skin sensor based on carbon black (CB) filled silicone and conductive fabric [131]. The CB silicone and conductive fabric is then cut into individual strips and woven to form a sensing grid of discrete taxels. The resistance in each discrete sensor in the grid is then independently measured using an inverting amplifier. A considerable merit of this approach is that it is stretchable. However, the use of discrete sensors remains a drawback in terms of discrete position sensing and potential for dead zones. The discrete sensors again require a large number of wire interconnects ($2n$ wires for n^2 discrete taxels). Additionally the use of silicone PDMS carbon black as a force sensor requires system identification and dynamic calibration due to the visco-elastic properties of silicone that confound force measurement with time. This negatively impacts the variety of geometry available, ease of manufacturing, and complexity of electronics and processing required.

To date, there exists no inexpensive, stretchable skin sensor that is simultaneously 1) scalable

(e.g. constant electrode count for increased surface area without decrease in temporal resolution or relative spatial position resolution); 2) provides continuous force and position sensing (without dead zones or limited taxels); 3) can be easily manufactured; 4) requires minimal electronics and processing, and 5) can be easily adapted to fit modern robot link surfaces. In prior work we proposed a solution: a flexible, stretchable smart skin sensor [132]. We exploit a novel carbon nanotube (CNT) doped PDMS elastomer sheet to meet these needs by implementing a 2D potentiometer similar to the work initially proposed by Walz et al. as a low-cost tool-tissue tracking method for surgical simulation [133]. In this work we characterize the accuracy, flexibility, and limitations of this sensor and demonstrate its utility in collaborative robotic applications.

A preliminary version of some of this work was presented in [134], where we presented the linear position method and a cubic force fit, as well as emergency stop and evasive action experiments. This paper additionally provides an alternative and independent diagonal position method, as well as an improved neural network model for estimating force including differentiation between low and high forces. Importantly, this paper now also evaluates the quality of the positional sensing over stretches up to 133% and the piezoresistive effects of stretch on bulk resistance, investigating and substantiating the claim of skin stretchability which was unsubstantiated in the original work. Additionally this paper shows a more detailed, confirmatory analysis of the microstructure of the skin and uses Akaike Information Criterion to rigorously evaluate the polynomial fit order.

6.3 Methods

The smart skin development consists of four primary components: the physical sensor, the electronics and algorithms, an optional offline finite element simulation, and the calibration routine. These components are outlined in the following section.

6.3.1 Sensor Design

The proposed smart skin sensor consists of three layers of flexible, stretchable materials (Fig. 6.1).

The top layer (Ag-nylon) is silver-coated conductive fabric (Medtex, Statex Productions, Bremen, Germany). The middle layer is non-conductive perforated cloth (Powermesh Fabric, 99% polyester, 1% spandex). The bottom layer (See Fig. 6.2b for detail) consists of a $1.75mm$ PDMS substrate with a uniform, bonded $100\mu m$ CNT-PDMS coating (7-SIGMA Inc., Minneapolis MN). The layers are cut into the desired pad dimensions ($14.7x14.7cm$, Fig. 6.2a) then bonded via non-conductive PDMS adhesive (7-SIGMA).

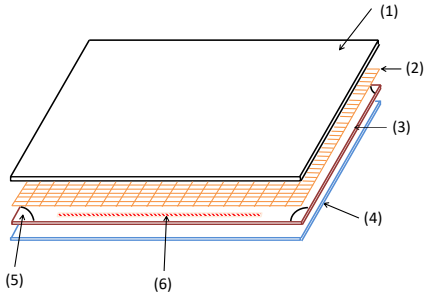


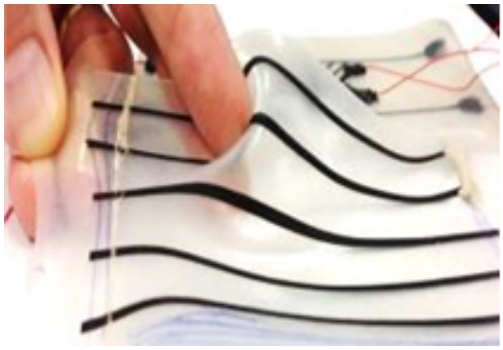
Figure 6.1: Smart skin exploded view with layers. (1) Ag. Nylon (2) Perforated Layer (3) PDMS-CNT Layer ($\approx 50\mu m$ thick) (4) PDMS Substrate (1.5 mm thick) (5) Stretchable Contact (Ag Nylon + PDMS-CNT) (6) Stretchable Adhesive

The electrodes are cut into $1cm$-radius quarter circles from silver conductive cloth infused with the CNT-PDMS resin material and heat cured onto the bottom layer (Fig. 6.2). A single wire is then conductively bonded to each cloth electrode. A contact resistance of about 500Ω is achieved while maintaining stretchable and flexible mechanical properties at the electrode locations. In this embodiment, the bottom layer exhibits an overall resistance of approximately $1.85k\Omega$ between corner electrodes.

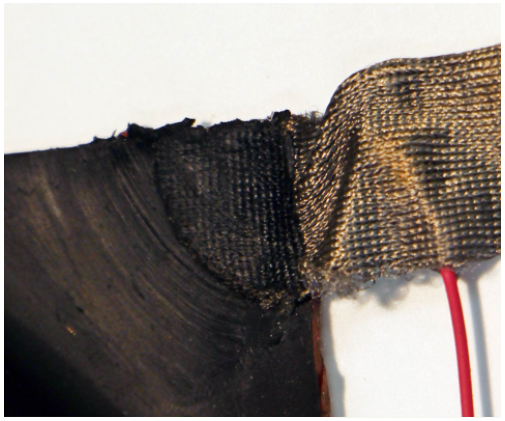
The CNT-PDMS was imaged with an SU8230 Ultra-high Resolution cold field emission Scanning Electron Microscope (Hitachi Ltd., Tokyo Japan) to evaluate the consistency and protrusion of the CNT's within the PDMS-CNT sheet. A small sample was prepared for SEM that exhibited both a mechanically torn-off and sheared-off region to investigate both cross sectional and surface distributions. No observable differences were noted between cross sectional regions obtained by tearing (shown, Fig. 6.5) or shearing (not shown). An overview image was first taken of the transition region between surface and cross-sectional edge achieved via tearing, Fig. 6.3 (left) and a close up of the edge (right). A typical region of interest on both the torn edge and sheet surface was identified for enhanced magnification to detail the CNT behavior at the sheet surface (Fig. 6.4) and within the edge (Fig. 6.5). Note that the higher voltage used in (Fig. 6.4) should provide more penetration further beyond past the surface boundary to reveal deeper structures. From Fig. 6.4, it appears that exposed CNT fibers occur within a typical spacing of roughly $10\mu m$. The fact that the CNT protrudes in a hairlike fashion from the surface allows the contact resistance to vary sufficiently for forces estimation, as shown later.

To quantitatively analyze human interaction a synthetic replica of an adult male index finger was created by casting A35 durometer silicone rubber (PlatSil 71-35, Polytek Development,

(a)



(b)



(c)

Figure 6.2: (a) The assembled smart skin with bonded zippers for easier attachment, (b) A close-up of the PDMS substrate with CNT stripes to indicate stretch, and (c) A close-up of CNT PDMS conductive cloth contact detail.

Easton, PA) to emulate typical human tactile interaction. A $6.5mm$ diameter wooden dowel was inserted in the finger for internal skeletal structure and repeatable attachment to load cells.

6.3.2 Electronics and Algorithm Design

In order to achieve a high degree of accuracy using the least amount of wires, four conductive nodes were attached to the smart skin, one at each corner. The four corner electrodes (nodes) and top fabric layer were wired directly to programmable GPIO pins of a simple microcontroller (ATmega328, Atmel Corporation, San Jose CA). The microcontroller allows changing each node to either V_{dd}, ground, high impedance or analog input. For position measurements,

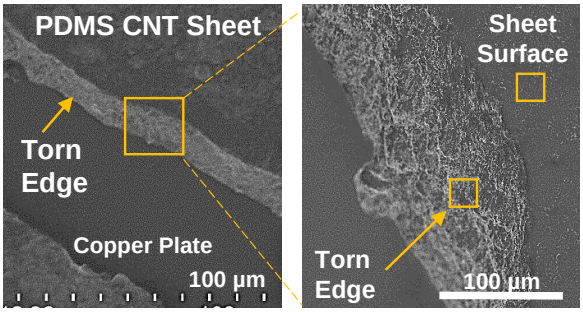


Figure 6.3: SEM overview of carbon nantoube-doped silicone skin sample (left) with a torn edge at 0.6kV x450 magnification in 1.9 mm LM(UL) mode. The detail image (right) shows a close up view of the torn edge with an apparent higher concentration of nanotube ends visible there than at the sheet surface at 0.6kV x3.50k magnification in 2.0mm LA0(U) mode; the yellow boxes indicate the approximate region shown it the detail images immediately below.

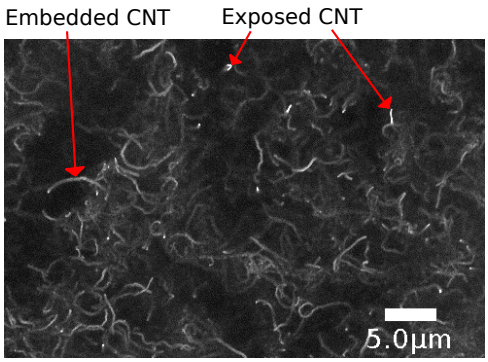


Figure 6.4: SEM detail image of sheet surface region showing carbon nanotube matrix embedded within silicone and exposed CNT ends protruding from surface (15.0kV x10.0k magnification in 5.1mm SE(U) mode).

the microcontroller was programmed such that the top fabric layer acts as a high impedance analog-to-digital converter and the CNT skin is rapidly pulsed in alternating configurations. Two adjacent terminals were set to V_{dd}, while the other two were set to ground. The conductive fabric voltage was measured, then the voltage values were rotated clockwise and the voltage measured again. The final two iterations were simply the opposite of the first two, providing another data point to average with the first. These two primary direction voltages are seen as the first two rows in Table 6.1 and the node/pin assignments in Fig. 6.6.

For force estimation, the top fabric was set to either V_{dd} or ground. One of the nodes was used

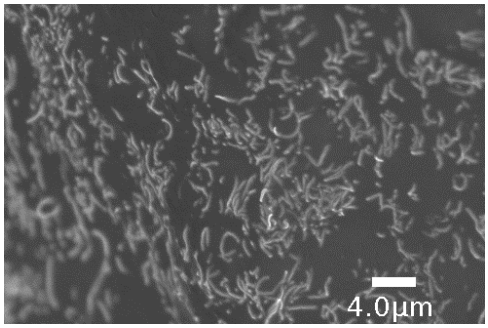


Figure 6.5: SEM detail image of nanotubes embedded within torn edge confirms significantly higher CNT density within the PDMS-CNT material than that expressed near the sheet surface (0.6kV x13.0k magnification in 2mm LA0(U) mode).

as an analog-to-digital converter in order to measure the ratio of contact resistance to the bulk resistance of the pad. The contact resistance is proportional to the contact force: higher forces result in lower contact resistance for a constant contact area but more finger force also allows more perforated holes to make contact, increasing the effective contact area. The setup can be rotated in a similar manner to the directional setup, and additionally the node configuration can be flipped. The fabric can be either V_{dd} or ground, yielding a total of 16 permutations (of which eight are merely opposites). A combination of the eight is utilized in the force estimation, in order to utilize the maximum dynamic range across the pad.

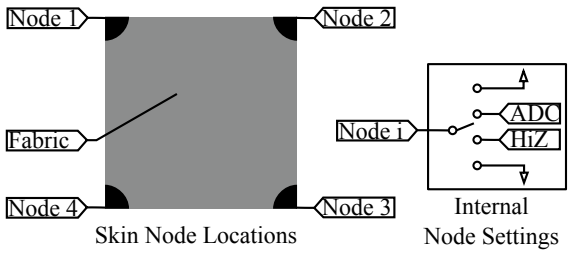


Figure 6.6: Smart skin electronic node schematic.

Since the force estimation and position estimation algorithms require different node setups, each estimation cannot be taken simultaneously. However by electronically switching the electrode values, position and force can be read in an alternating fashion. This allows a minimum setup that can be very small, for example if an ATTiny85 (Atmel Corporation, San Jose CA) were to be utilized, the setup shown in Fig. 6.7 would be sufficient, with only 5 wires and one

microprocessor. The switching and reading for each mode requires approximately $1ms$. This allows minimal force and position information readings of $3ms$.

Node #	1	2	3	4	Fabric
V_{vert}	Gnd	Gnd	V_{dd}	V_{dd}	ADC
V_{horiz}	Gnd	V_{dd}	V_{dd}	Gnd	ADC
$V_{dd} - V_{vert}$	V_{dd}	V_{dd}	Gnd	Gnd	ADC
$V_{dd} - V_{horiz}$	V_{dd}	Gnd	Gnd	V_{dd}	ADC
V_{diag1}	Gnd	HiZ	V_{dd}	HiZ	ADC
V_{diag2}	HiZ	Gnd	HiZ	V_{dd}	ADC
$V_{dd} - V_{diag1}$	V_{dd}	HiZ	Gnd	HiZ	ADC
$V_{dd} - V_{diag2}$	HiZ	V_{dd}	HiZ	Gnd	ADC

Table 6.1: Permutations used for position. Top four rows use linear method, bottom four rows use diagonal method.

Node #	1	2	3	4	Fabric
V_{f1}	Gnd	Gnd	ADC	HiZ	V_{dd}
V_{f2}	Gnd	Gnd	HiZ	ADC	V_{dd}
V_{f3}	ADC	HiZ	Gnd	Gnd	V_{dd}
V_{f4}	HiZ	ADC	Gnd	Gnd	V_{dd}
$V_{dd} - V_{f1}$	V_{dd}	V_{dd}	ADC	HiZ	Gnd
$V_{dd} - V_{f2}$	V_{dd}	V_{dd}	HiZ	ADC	Gnd
$V_{dd} - V_{f3}$	ADC	HiZ	V_{dd}	V_{dd}	Gnd
$V_{dd} - V_{f4}$	HiZ	ADC	V_{dd}	V_{dd}	Gnd

Table 6.2: Permutations used for force.

The system was assumed third-order, since it was adequately described by a two by three polynomial surface with voltage as a function of positions in section 6.3.3. The inverse mapping, with position as a function of voltages is thus expected to also follow a third order polynomial, and the equation for the position is a two by three polynomial surface with cross terms, where the third order term goes along the direction in question (Eq. 6.1). For example, to calculate

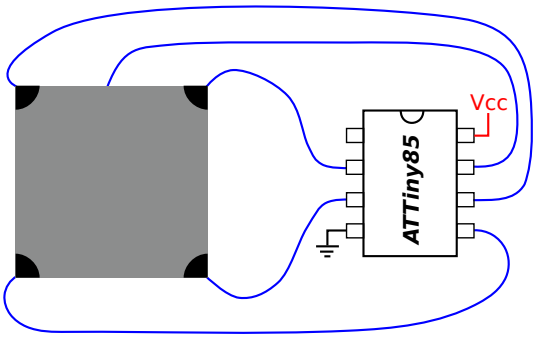


Figure 6.7: Minimal smart skin setup requires supporting interface electronics between microcontroller and skin pad.

X, the V_{horiz} that increases is used as the third-order term:

$$X = \begin{bmatrix} V_h^3 & V_h^2 V_v & V_h V_v^2 & V_h^2 & V_v^2 & V_h & V_v & 1 \end{bmatrix} \cdot \begin{bmatrix} a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \end{bmatrix}^T \tag{6.1}$$

This equation was fitted to the calibration data, then implemented in the microcontroller to provide real-time position data to the robot controller for the collaborative interaction experiment.

The relation between node voltages and applied force was not found to have parametric model. Instead we utilized a Neural Network (NN) regression to relate the node voltage to the applied force. For this approach we used the eight permutations found in Table 6.2. Using a known applied force we then fit the NN regression weights in a three layer NN setup. Hidden layer 1 consisted of ten nodes, hidden layer 2 consisted of five nodes and the output consisted of one node. The two hidden layers utilized a sigmoid activation function while the output node used a pure linear activation function. The calibration data for force was collected and used in a standard back-propagation scheme to train the model. The feedforward model was then implemented in the microcontroller.

In many robotic cases, estimation of continuous force values is not necessary, instead it is often preferable to provide alerts when force values exceed a threshold or assume certain discrete force levels. We additionally used the force estimation from the NN regression to estimate three discrete force levels: Low Force ($2.5N < F < 7.5N$), High Force ($12.5N < F < 17.5N$), and No Force ($F < 0.5N$). This information is then either sent to a relay on the robot's emergency stop circuit for the emergency stop experiment, or used to verify the validity of touch for the

collaborative interaction experiment.

6.3.3 Finite Element Modeling and Simulation

The position sensing algorithm was based on a finite element model of conduction diffusion. We modified the approach of [133] to create a finite element model of the voltage as a function of two-dimensional position with a rectangular $14.7cm \times 14.7cm$ Neumann boundary and quarter-circle pads ($1cm$ radius) as Dirichlet boundaries. MATLAB's (MathWorks Natick, MA) Partial Differential Equation toolbox numerically solved the DC conduction diffusion problem: $-\nabla \cdot (\sigma \nabla(V)) = q, E = \nabla(V)$, with $V = 5volts$ and a mesh of $n = 10417$ nodes. The simulation results are shown in Fig. 6.8 for a single linear permutation of corner electrode voltages, and in Fig. 6.9 for a single diagonal permutation.

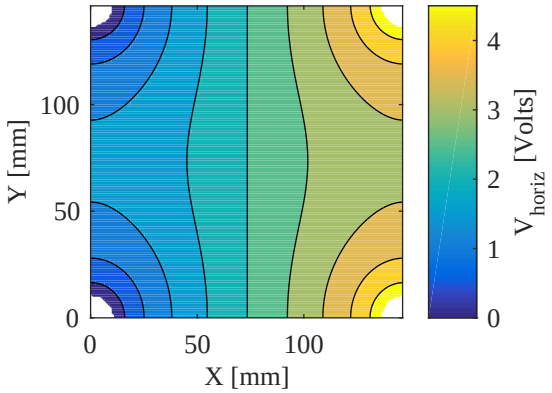


Figure 6.8: Finite element simulation with linear sweep.

To enable real-time processing on a microcontroller we employed a simplified polynomial surface model using least squares regression with bisquare robustness correction to fit coefficients. Given the odd and even symmetry displayed in Fig. 6.8 we chose cubic and quadratic terms. Because of this lowest-possible model complexity (polynomial order) for the observed curvature, we adopted this polynomial model to represent the complexity of our mapping between two-dimensional measured voltages and positions.

While a fit of the predicted voltage at a given point, as [133] showed, is useful, for the skin application the inverse solution was what was actually needed, so that for a given observed voltage a position estimate could be given. This was achieved by simply fitting the X and Y dimensions each to a 3rd order polynomial of the voltages V_1 and V_2, which are explained further in Table 6.1. The fitting of the polynomial for X is shown in Fig. 6.10 where the white area

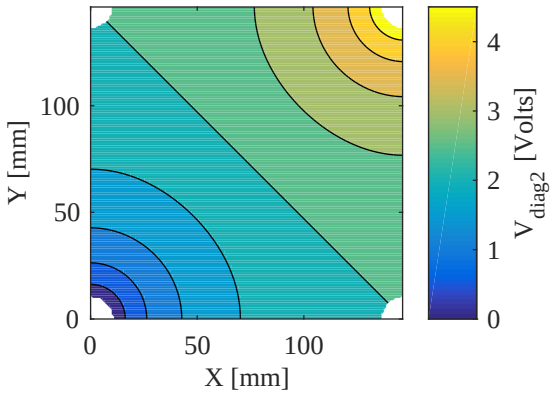


Figure 6.9: Finite element simulation with diagonal sweep.

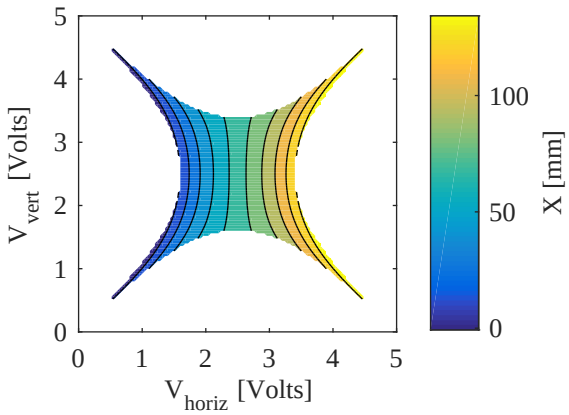


Figure 6.10: Inverse mapping of linear voltages to position (white area does not map to workspace, i.e. it indicates voltage combinations that were never observed).

represents voltage combinations not present the simulation.

To evaluate the level of overfitting we ran an Akaike Information Criterion (AIC) analysis to determine which terms from the full 3rd order polynomial with crossterms were required and which were extraneous (Eq. 6.2). We used the residual sum of squares (RSS) to evaluate a models maximum likelihood estimate. Fig. 6.12 shows the diminishing returns beyond eight terms, corresponding to a 3rd order by 2nd order polynomial with cross terms.

$$AIC = 2k + n\ ln(\frac{RSS}{n}) \tag{6.2}$$

In addition to the linear sweep method from [133] we found that a diagonal sweep, holding

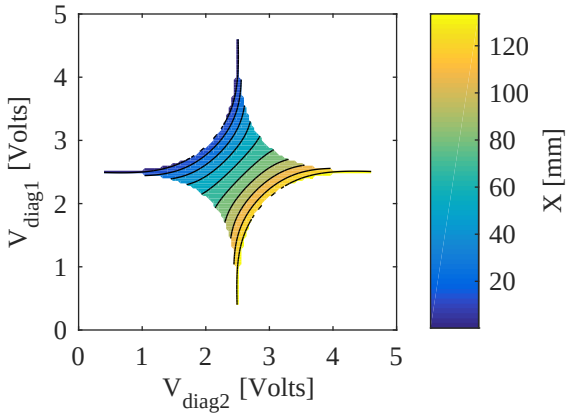


Figure 6.11: Inverse mapping of diagonal voltages to position (white area does not map to workspace, i.e. it indicates voltage combinations that were never observed).

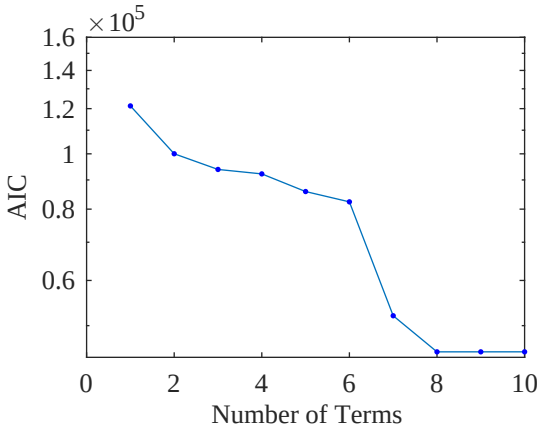


Figure 6.12: Akaike information criterion (AIC) of best inverse mapping polynomial fit to linear FEM.

the other two corners at high impedance, provided a similar level of quality to the linear method, with the same polynomial order.

To evaluate the level of overfitting we again ran an AIC analysis to determine which terms from the polynomial were required and which were extraneous. Fig. 6.13 shows the diminishing returns beyond nine terms, corresponding to a 3rd order by 2nd order polynomial with cross terms.

These two methods can also be combined into a single least squares fit, which provides a very small benefit in simulation, but can be used in the physical system to reduce Gaussian error by

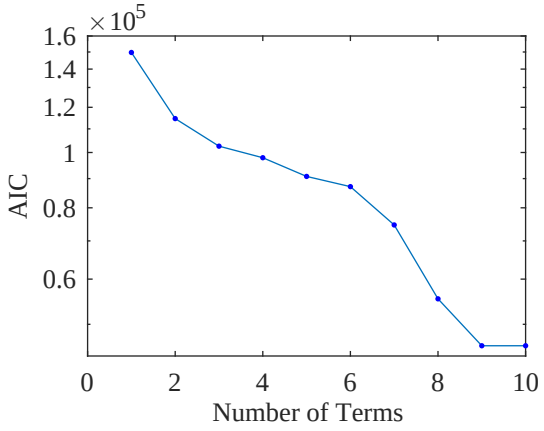


Figure 6.13: Akaike information criterion (AIC) of best inverse mapping polynomial fit to diagonal FEM.

using four separate readings instead of just two.

6.3.4 Experimental Design

The proposed smart skin was evaluated through a series of experiments to validate our inverse kinematic model and to assess utility with regards to collaborative robotics. A variety of calibration routines were required to validate this inverse model. The first calibration routine was performed for the 2D position of force application. The second calibration routine fit the parameters for the force magnitude. The third calibration routine validated the inverse kinematic model with the skin in multiple stretched states.

Experiment 1 In order to calibrate the 2D position sensing, a routine was established wherein a fixed force was applied to the skin in known locations along a regular 2D grid. The known x-y locations and the measured voltage potentials were recorded for each point on the grid. This data was then applied to a BiSquare least squares fit in MATLAB in order to fit to the 3rd order polynomial described in Section 6.3.2.

In order to apply a known force in a known x-y grid, a custom robotic arm, CORVUS, was used [135]. The CORVUS arm was outfitted with an end effector comprised of the conformable replicate finger mold (Fig. 6.14) described in section 6.3.1. The force was controlled by maintaining a constant compression of the replicate finger through position control. Force was also monitored through a load cell to ensure a stable force was being applied. The x-y grid used a $1cm$ separation in each dimension on a sample skin area of $144cm^2$.

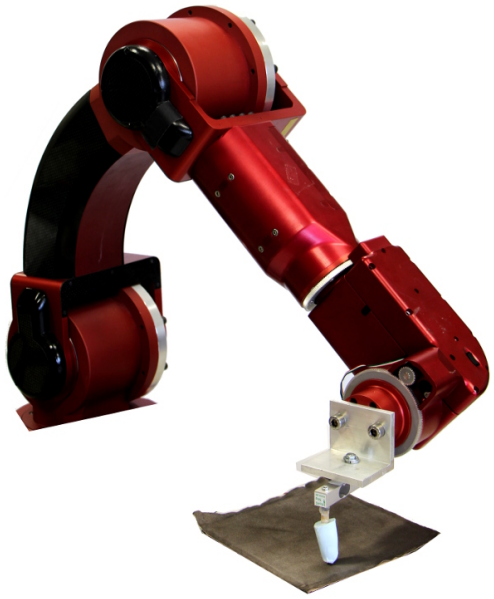


Figure 6.14: A custom robotic arm was used to calibrate position sensing in typical human tactile interactions by applying known forces in a known 2D grid via a human finger replica.

Experiment 2 To calibrate force, a load cell on the synthetic finger was used to measure the interaction force, while all of the voltage permutations were measured. Both sensors recorded at $30Hz$ throughout the touch. This test was performed at the center of the pad.

The load cell data was synchronously recorded along with the voltage values. This data was then used to the Neural Network regression model where the input variables consisted of the eight node voltage permutations as described in section 6.3.2. The results of this regression can be found in section 6.4.

Experiment 3 To evaluate accuracy in a stretched state, the skin was placed under an XYZ linear stage which automatically actuated forces in a regular grid. The skin sample was placed in a micrometer stage on a stationary table. On the Z stage a conductive end effector followed a downward trajectory onto the skin. The skin was stretched by clamping the left and right 5mm strips in a vice clamp attached to a micrometer. The calibration grid was set to a constant 5mm spacing in X and Y for each stretch state. The grid location and sensor data for each touch was logged for 1 second with an average taken for each node permutation. The skin was tested at a neutral 100% stretch state (90 mm between clamps), a 106% stretch state (95 mm between clamps), a 111% stretch state (100 mm between clamps), a 122% stretch state (110 mm between clamps), and a 133% stretch state (120 mm between clamps). The skin was tested without a top

fabric layer in order to avoid extraneous contact artifacts between clamped layers, instead the force actuating end effector on the Z stage was covered with the conductive fabric node. The true position and conductance data was then fit to the proposed polynomial surface model for each stretch level to assess the degree to which the polynomial fit method works across a range of stretch levels.

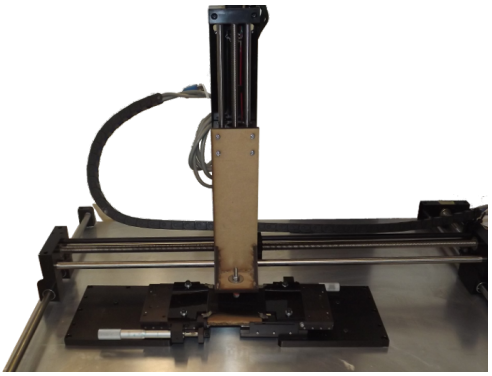


Figure 6.15: Stretch evaluation setup.

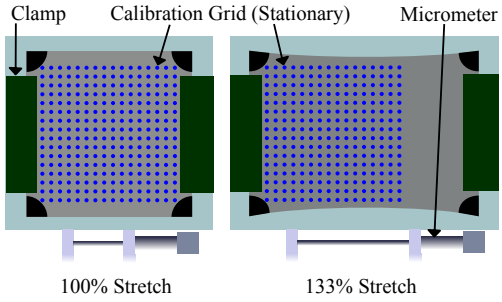


Figure 6.16: Schematic of stretch evaluation setup.

Additionally, during the stretch test the bulk resistance of the pad was also measured by adding a shunt resistor to each node and measuring the voltage differential across the resistor to estimate the current flow into or out of the pad. This sensing modality requires yet another permutation, where the fabric and two nodes are high impedance, with one node grounded and one node at V_{dd}. This was logged for each press at each stretch level. This additional circuitry would only be necessary in practice if the skin was to be stretched or unstretched actively during use.

Experiment 4 Multiple experiments were designed to assess the utility of the smart skin in a collaborative robotic setting. To evaluate the functional response of the smart skin, the skin was mounted to conform to the surface of a distal link of the CORVUS robot (Fig. 6.17). The skin was connected to the emergency stop circuit by means of a solid state relay. The skin's microcontroller was programmed to sample at $150Hz$ and respond to any touch with non-zero force by enabling the robot's emergency stop circuit. The robot was then commanded a trajectory which ran at $3mm/s$ and intercepted the synthetic finger attached to a load cell (Fig. 6.17). During the experiment, emergency stop state and force data was collected. This experiment was repeated 20 times to test repeatability.

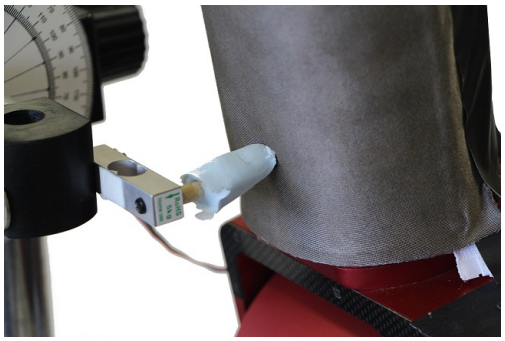
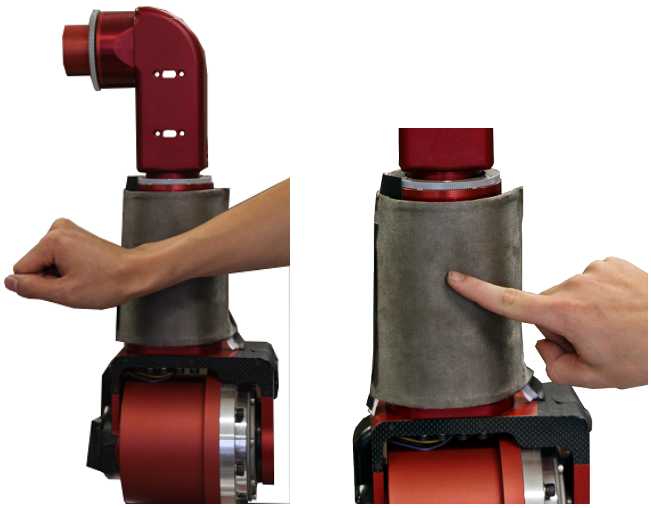


Figure 6.17: Known forces were applied to the skin mounted on a link via the silicone cast finger attached to a load cell.

In order to further evaluate the emergency stop functionality of the smart skin, the test was run with the synthetic finger replaced by a human test subject's arm (Fig. 6.18a). The skin was programmed to trigger an emergency stop in a similar manner to the above experiment, and again the robot was commanded to take an unsafe trajectory that intercepted the human. The experiment was repeated 20 times to test for repeatability.

Experiment 5 In order to functionally verify the skin's positional response in human-robot interactions, the skin was placed on the CORVUS robot as above but a human operator pushed on the link with an index finger. Then, the microcontroller was set to determine if a valid touch had occurred, and if so, output the position of a touch. The robot was programmed to respond to touches by moving in the opposite direction. The location of contact provided by the skin determined this direction based on the geometry of the link. The direction was assumed to be normal to the skin, so the robot would move directly away from the finger. The synthetic finger attached to a load cell was moved toward the robot while maintaining a constant velocity, with the robot either set to take evasive action or remain still.



(a) (b)

Figure 6.18: An arm (a) and a finger (b) were placed in the path of the robot.

6.4 Results

The construction of the smart skin resulted in a highly stretchable and flexible surface. The electrical properties of the skin such as conductivity did not change significantly with the repeated application of stress. As shown in Fig. 6.19, the skin is capable of stretch to approximately 150% of the original size without tearing.

The skin was successfully applied to the irregular surface of the CORVUS robot arm, and functioned on the arm. The skin installed on the robot arm can be seen in Fig. 6.18b.

Figure 6.19: Evidence of stretching near 150%.

An overall outline of the results concerning model validation and collaborative experiments

can be found in Table 6.3.

Experiment 1 Using the X-Y calibration data of potential, force and location, the least squares fit was run to determine the coefficients for the polynomial. The coefficients were computed with R^2 values of 0.9934 for the X-Fit and 0.9978 for Y-Fit. The mean absolute error was $3.32mm$ and RMSE was $7.02mm$. 90% of all error was below $5.7mm$, and the highest error was at the corners and boundaries. The 3D plot of the polynomial in X Cartesian space plotted alongside the calibration points is shown in Fig. 6.20, and the polynomial in Y Cartesian space is shown in Fig. 6.21.

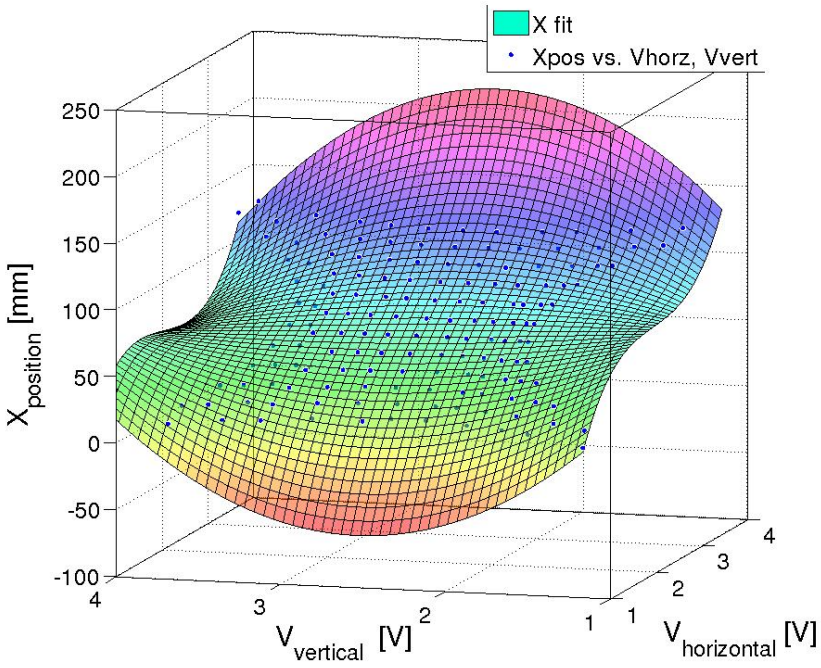


Figure 6.20: Polynomial fit in X ($R^2 = 0.993$).

Experiment 2 The force measured by the load cell is plotted against a single node voltages for 144 separate touch incidents located at the center of the pad (Fig. 6.22). A Neural Network (NN) regression model relating node voltage to true load cell force was fit with an R^2 value of 0.875.

Additionally a force magnitude classification was performed using the NN regression wherein the true force was used to segment the data into two categories: Low Force ($2.5N < F < 7.5N$) and High Force ($12.5N < F < 17.5N$). Using the voltage data from these two categories the

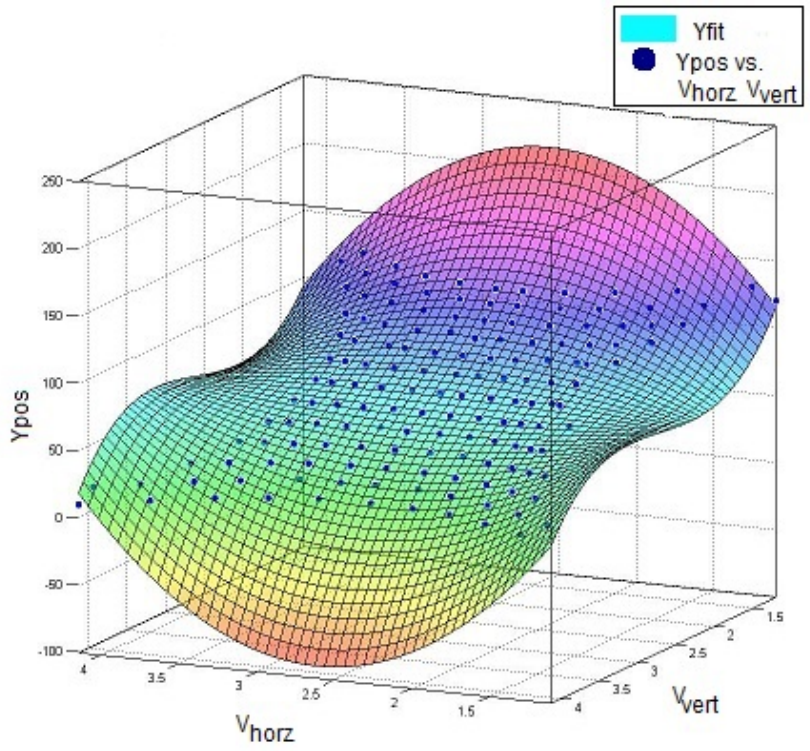


Figure 6.21: Polynomial fit in Y ($R^2 = 0.998$).

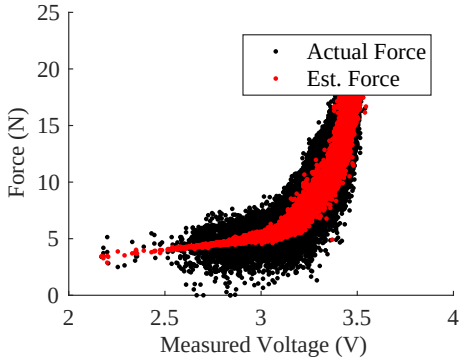


Figure 6.22: Neural Network Model Relating Node Voltage to Force ($R^2 = 0.875$).

output of the NN regression was plotted versus the two ranges of force magnitude (Fig 6.23). The results indicate clear separation between the low and high force ranges.

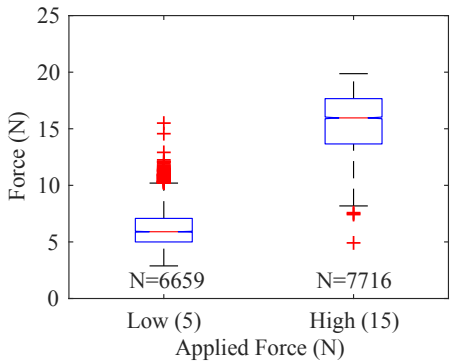


Figure 6.23: Estimated Force Versus Force Magnitude Category.

Experiment 3 The stretched skin provided good polynomial fits in X and Y (Fig. 6.24), with no systemic differences between the unstretched and 133% stretch cases.

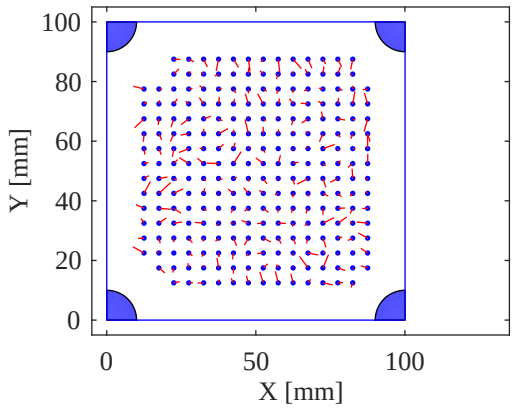
The results of the stretch test show that the error does not increase as the skin is stretched (Fig. 6.25), staying at approximately $2.5mm$ with a standard deviation of $1.5mm$. This allows the skin to be calibrated at any number of stretch levels.

The results of the stretch test also show that the bulk resistance of the pad (as measured between electrode pairs along the top and bottom of the pad using shunt resistor current) varies with stretch (Fig. 6.26). Each stretch level is more than 2 standard deviations from the next allowing estimation of stretch level online by measuring the current. Then a calibration for that stretch level could be loaded to provide adaptable stretch functionality.

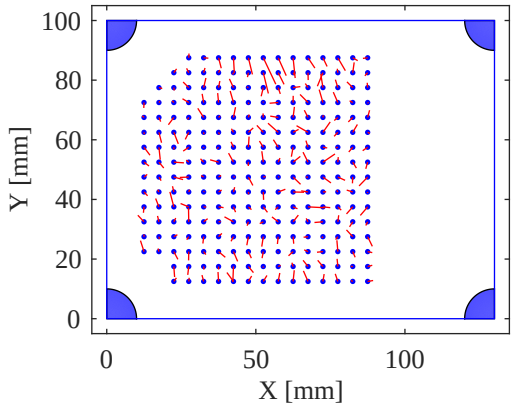
Experiment 4 The smart skin's response to a finger touch is shown in Fig. 6.27. The vertical line shows the skin's triggering of an emergency stop event. The green horizontal line shows the force at which the skin triggers that a touch has occurred, which was at 0.5 N. Beyond the point where the emergency stop was triggered, the forces represent the dynamics of the robot, in this case the brakes take approximately 100ms to fully stop the robot, and oscillations continue beyond that point. The skin successfully detected the touch and brought the robot to a halt for all 20 iterations of the experiment.

In the second emergency stop experiment with the human subject, the interaction force was not quantitatively measured. But, no significant force on or movement of their arm during the tests was observed or reported. The skin successfully detected the touch and brought the robot to a halt for all 20 iterations of the experiment.

Experiment 5 The result of the collaborative control experiment is shown in Fig. 6.28. The control test without evasive action rapidly reaches the limit of the load cell and does not decrease,



(a) Skin 2D fit with no stretch.



(b) Skin 2D fit at 133% stretch.

Figure 6.24: The skin unstretched (a) and at 133% stretch (b) shown with the actual point pressed shown as the blue circles, with the red line drawing to the point estimated by the inverse mapping polynomials to indicate error.

as the target position is never achieved. However, with evasive action triggered by the contact data from the smart skin sensor the force is maintained at approximately $11N$, which is below an example critical force of $12.5N$, until the target position is reached.

6.5 Discussion

The X-Y calibration provided an accuracy of below one centimeter, which is sufficient for many bulk sensing applications, such as those performed by human skin. This would provide sufficient

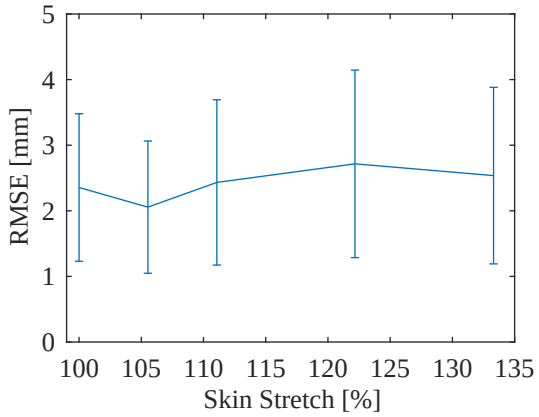


Figure 6.25: Accuracy of polynomial fit over various stretch levels. Error bars show (+/−) one standard deviation.

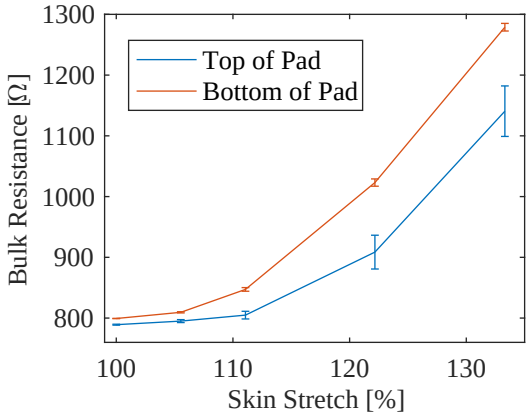


Figure 6.26: Bulk resistance of pad between electrode pairs along the top and bottom of the pad for various stretch levels. Error bars show (+/−) one standard deviation.

accuracy to allow situational awareness to a robot with a soft and durable continuous skin. A limitation of this experiment, however, is that multitouch was not tested, and robust multitouch does not seem possible with the specific methods described here. Future work should involve using node currents as well as node voltages to detect that a multitouch is occurring and possibly estimate the multiple touch locations.

The force measurement provided the ability to differentiate between low and and high forces, but discrete force estimates do not provide enough repeatability to allow continuous force measurements. The continuous force estimate had an R^2 value below 0.9 and therefore may not

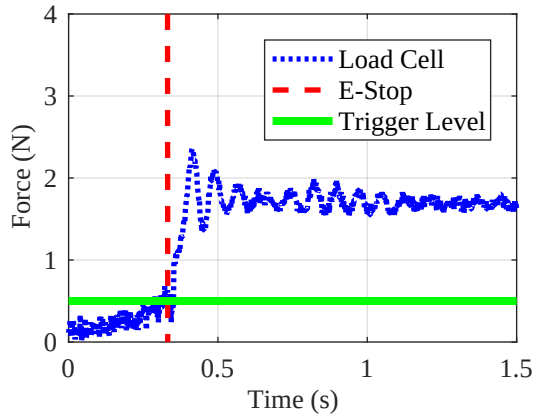


Figure 6.27: Force on finger with emergency stop.

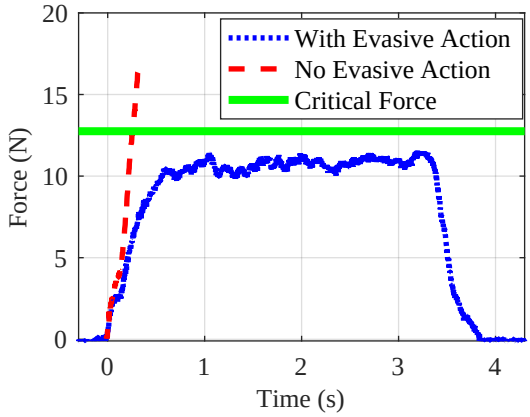


Figure 6.28: Force on finger with evasive action.

provide sufficient accuracy for some applications. A limitation of the experiment is that the force measurement was performed with a synthetic finger, and due to the contact resistance estimation method the force estimate is highly dependent upon the geometry and hardness of the contacting object. Therefore future work should focus on further characterizing the force estimation, especially focusing on contacting objects specific to the desired use case.

The stretch calibration showed that accuracy does not diminish as the skin is stretched, although the change in calibration for each stretch level did not seem to follow an expected pattern so a separate calibration should be done for each stretch level expected for the desired use case. If the use case involves the skin to be in a rigid state, such as stretching and permanently affixing to a rigid robot link, the calibration should be done in place to provide maximum

Table 6.3: Summary Overview of Experimental Results

	Experiment	Primary Result
1	Stationary Calibration	RMSE = $7.02mm$
2	Force Calibration	$R^2 = 0.875$
3	Stretch Calibration	RMSE = $2.5 \pm 1.5mm$
4	Emergency Stop	Threshold = $0.5N$
5	Collaborative Control	Collaboration Force = $10N$

accuracy while automatically calibrating to a functional coordinate frame. Calibration can be completed with approximately 25 known locations in a relatively simple procedure. If the use case involves dynamic motion, such as around an elbow of a robotic joint, the skins should be calibrated to the full range of stretches expected, and the bulk resistance of the pad should be measured in real time to determine stretch level and choose the appropriate calibration. Further work should determine the effect of repeated stretch on functionality, and evaluate stretching modalities other than the uniaxial mode shown here.

The emergency stop experiment showed the usefulness of this sensor in a robotic system. The low threshold force of just half a Newton is roughly equivalent to the weight of a tennis ball, and the test was performed with a human arm and finger, without causing any discomfort. A limitation of this experiment was that it was performed in a high speed mode, where the position calculations were not performed and only emergency stop decisions were made. Future work should continue to evaluate emergency stop detection as other features, such as multitouch are added, to ensure that reaction times stay fast enough for the application.

While the force experiment showed that discrete force estimations are quite noisy, the collaborative control experiment showed that the force could be used in a feedback loop to roughly maintain an interaction force. The quality of the interaction will be dependent both upon the robot used and the contacting object, however in a collaborative environment with humans. This could be acceptable as larger contact objects would correspond to faster reactions, which would likely be appropriate.

Qualitative analysis was performed on the SEM scans of the skin material. The roughly $10\mu m$ spacing of carbon nanotubes protruding at the surface for viable contact places a theoretical lower bound on the positional resolution accuracy achievable with this method of doping PDMS with CNT. However, given the relatively low expression of CNT's near the surface of the sheet when compared to within the bulk body (see Fig. 6.5), it is likely that the effective electrical

contact area of the nanotubes can be increased. Also it should be noted that due to electrical constraints in practice (such as analog to digital conversion), the resolution is much lower.

The capability of sensing contact on the whole body of a robot is an important step for enabling safe and more advanced, intuitive human-robot interactions. In this paper a design of a flexible, stretchable skin that can detect location within a typical $5.7mm$ resolution was presented. A 2D potentiometer concept for position and demonstrated that the same system can also provide a crude estimate of contact force. By design, the sensor can be scaled up in size due to its ratiometric measurement method for positional accuracy. It can also be scaled down. Given the finite CNT distribution evident in the SEM images, there is a lower limit with this method.

A significant limitation of the skin as shown in this paper is that the outer surface is a conductive node, and therefore if it is contacted by anything electrically active the sensor could be damaged. Future work should address this by investigating using another PDMS sheet with CNT as the top layer, potentially with a different loading of CNT, while leaving a waterproof PDMS layer exposed on the outside of the sensor. This top layer could also have four nodes, leading to a full eight nodes, with many more permutations available for features such as multitouch.

Repeatedly stretching this sensor does not influence its localization error significantly, and additionally localization accuracy is not affected by stretching. By design, the skin's durability is determined primarily by the PDMS substrate. PDMS is among the most durable stretchable elastomers available for typical manufacturing implying that the skin sensor exhibits favorable durability. The minimum threshold force and sensitivity are tunable via mesh size, elastomer, and fabric mechanical properties. Multiple coarse force thresholds are possible via multiple layers. In this work, we demonstrated a single layer version with a detection threshold of 0.5N for contact sensing which exceeds the sensitivity of typical in-joint torque sensors or estimates (e.g. the KUKA or UR5 robots). Adding such a skin could effectively expand the safe operating speed range of robots with internal joint torque measurements. Alternately, it could provide inexpensive tactile sensing for existing robots without incurring the costs of internal joint torque sensors.

While the skin was demonstrated in both an automatic emergency stop system as well as interactive robotic environments, the application of this design is not limited to robotics and machinery, but can also be applied to medical devices, specifically prosthetics. Prosthetics are very similar to robotics as they are a robotic extension of an appendage without the human sense of touch [122]. With this design, localized feedback to the operator may be possible (as per neural interface methods in [136]) but at a substantially lower cost and manufacturing complexity then current sensor designs. Another application could be in highly constrained collaborative environments such as surgery where interactions are inevitable, such as the example shown in

Figure 6.29: Example collaborative use case.

Fig. 6.29.

This skin provides an affordable and electrically simple solution to the widespread problem of giving robots a sense of touch, and the stretchable and flexible design allows greater adaptability while still providing accurate positional sensing and broad force range detection.

Chapter 7

Dynamic Bioprinting

This chapter is a reproduction of two papers. A paper submitted to the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, and a paper submitted to the 2017 Hamlyn Symposium on Medical Robotics. As such each paper was designed to be able to stand alone, and contains its own abstract, background, results, and interpretation. The relevance of this work to the wider thesis goal and body is discussed in the Chapter 8 Conclusion.

7.1 3DBioprinting Directly onto Moving Human Anatomy

This section is a reproduction of a paper submitted to the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems.

7.1.1 Abstract

This paper establishes the feasibility of robotically 3D printing biomaterials such as alginate hydrogels onto moving human anatomy and a stationary plane. The alginate hydrogels used are *in-vivo* compatible and a proven biomaterial for tissue scaffolds. We developed a control scheme for precision material deposition via piezo microjetting while tracking in real-time to continuously sense anatomy location and deposits material in a predefined trajectory derived from two pre-selected target geometries. We show that multilayer 3D structures can be created on a moving human hand with 1.6 mm average error and 87.8 % overall accuracy.

7.1.2 Introduction

Additive manufacturing has become a ubiquitous technology that allows for rapid prototyping, personalized design, and small-scale production. A variety of additive manufacturing methods

exist, including fused-deposition modeling, selective laser sintering, and stereolithography. These methods utilize build materials such as plastics and metals. Materials are typically deposited onto a static, planar build surface and the object is built up layer by layer.

Recently, bioprinting technology has advanced in the field of tissue engineering via additive manufacturing techniques [137, 138]. Potential applications include tissue or organ regeneration, creation of biometric multi-layered skin tissue, and burn wound treatment [139]. One bioprinting approach has been to deposit living cells onto a surface using an inkjet system [140] noting several advantages over the more popular extrusion-based systems due to an inkjet's high-speed control and non-contact interface. In another approach, stem cells were embedded in a hydrogel solution and then deposited via pressure-driven nozzles onto skin wounds with the benefit of laser-based position sensing [141]. In this case the bioprinted stem cells provided better wound-closure rates than the manually applied gels. In [142] synthetic materials designed to mimic human skin were 3D-printed through pressure controlled channels on a linear 3-axis robotic stage. This study showed more accurate cell localization and 3D architecture of the reconstructed epidermis when compared with manual methods.

Prior art has demonstrated significant benefits of bioprinting for tissue engineering, however its scope has been constrained to depositing materials onto stationary targets. It traditionally emphasized planar substrates with open loop deposition trajectories [138]. In laboratory settings this has required printing onto phantom culture disks or printing onto sedated animal subjects. A case for printing directly onto anatomy has been made [141, 143]. In some clinical settings however, anatomy may be free-moving (such as the unfixtured hand of a burn patient that must move during therapy to maintain range of motion for skin grafts) or exhibit quasi-cyclic motion (such as a beating heart, breath-induced thorasic cavity motion, or vascular pulsatile throbbing of artery-proximate brain tissues). Therefore to increase the applicability of bioprinting, particularly in human-in-the-loop contexts, additive manufacturing techniques need to be augmented to allow for the deposition of material onto moving 3D surfaces. Previous attempts at tracking and drawing on a hand include either direct contact with the hand [144] or tracking the hand only in two degrees of freedom [145]. The gap in prior art has been the demonstration of an additive manufacturing technique capable of depositing 3D geometries of viable biomaterials directly onto unconstrained, non-planar, moving anatomy.

The objective of this paper is to demonstrate the feasibility of robotically depositing bioprinting-compatible materials directly onto unconstrained, moving human anatomy. Specifically, we demonstrate the 3D printing of alginate hydrogels onto (i) a non-stationary human hand (Figure 7.1) and (ii) a stationary plane to serve as a baseline. Alginate hydrogels are *in vivo*-compatible and a proven biomaterial for tissue scaffolds [146, 147]. A temporal coarse-fine approach controls precision material deposition via microjetting (a generalization of inkjetting

to situations where the deposited material is not strictly an ink). Our system employs a real-time tracking algorithm to continuously sense anatomy location and deposit material onto it in a predefined trajectory derived from two pre-selected target geometries: a 2D block 'M' logo and a 3D stepped pyramid.

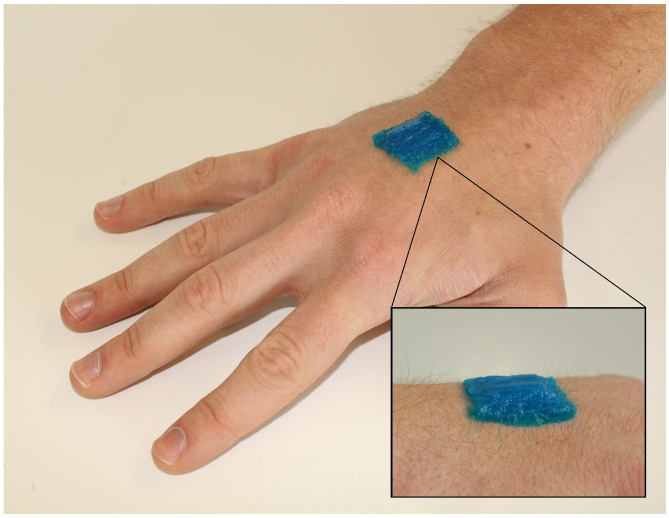


Figure 7.1: Concept of 3D printing *in vivo* compatible bio-materials directly onto unconstrained, free-moving anatomy.

7.1.3 Methods

7.1.4 Hardware

Our custom additive manufacturing platform consisted of three primary components: a 3 DOF robotic platform, a material deposition system, and a sensing apparatus (Figures 7.2-7.3). The robotic platform used was an XYZ gantry system (Newmark Systems Inc, Rancho Santa Margarita, CA). This gantry stage has an XY travel distance of $600mm$ and a Z travel distance of $300mm$. Each dimension is actuated by a stepper motor, which is controlled with a DRV8825 stepper motor driver. The stepper motor drivers are in turn controlled by an ARM Cortex microcontroller (Teensy 3.2, PJRC Sherwood, OR).

Camera-in-hand velocity control was implemented with a proportional controller, where the velocity commanded to the steppers was proportional to distance error in X, Y, or Z measured between desired deposition location and actual measured location of the target. The gantry axes were aligned with the sensors to allow a one-to-one mapping. The proportional command was capped to between $2mm/s$ and $50mm/s$ to avoid unrealistic demands of the steppers as well

as to keep the velocity from dropping near zero, as the drop-on-demand nature of the material deposition system is such that the robot does not need to stop at each point, but merely needs to glide over the point. The acceleration of the steppers was also capped to $312mm/s^2$ avoid undue stress on the system.

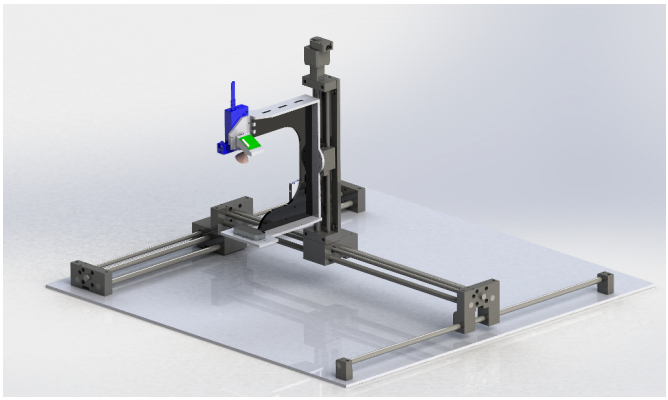


Figure 7.2: Additive Manufacturing Gantry Setup

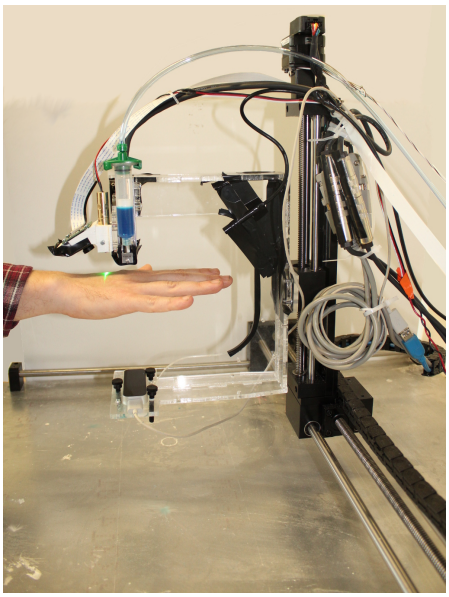


Figure 7.3: Additive Manufacturing Deposition Onto a Moving Hand

The material deposition system is an integral component of this design. While most 3D printers use extrusion, which deposits in a continuous bead, such an approach is not suitable for moving anatomy since the bead would need to be severed instantly if the anatomy moved

out of position. For this approach we instead utilized an active deposition method based on a micro-jetting system capable of depositing micro-beads of solution. Specifically we utilized the PICO Pμlse jetting system (Nordson EFD, Westlake, OH). This system allows the deposition of viscous materials at a rate of $1kHz$ and orifice sizes of $50-600\mu m$. This selected range of orifice sizes roughly permits deposition of materials with viscosities between $200-500$ centipoises, though much larger ranges are available. The pulse timings used were $0.50ms$ pulse length and $8.00ms$ cycle length. With this approach we can employ a temporal-based manufacturing method wherein material is not deposited until the anatomy is positioned correctly beneath the extruder. Once material has been deposited, the flow is paused until correct positioning is achieved again.

The sensing modality consisted of two sub-components. For XY tracking we utilized an off-the-shelf hand tracking system (Leap Motion, San Francisco, CA). This system provides a 120 Hz framerate with a typical positioning accuracy near $1mm$ [148]. Unfortunately, the depth information from the Leap Motion is derived from stereo vision and is less accurate. The depth information is critically important for the deposition of subsequent layers to succeed and to minimize deposition errors from increased droplet travel distance. For accurate depth we designed a custom depth sensing system using a projected line green laser ($532nm$, $1mW$) and monocular camera (Figure 7.4).

The camera used was a hardware Raspberry Pi (Raspberry Pi Foundation, Cambridge, UK) camera that provides low-latency capture at $640x480$ resolution at $90Hz$. The laser produces a single horizontal line of green light in the camera frame. A custom support structure was designed to orient the optical axis of the camera at a 30 angle relative to the axis of the laser projection. This setup allows the laser line to be seen at a range of $10-60mm$. The green laser line appears in the image space as a vertical line. The location of the line G_j $(0-640)$ is found in each row j of the image by finding the brightest pixel. Using the average G_j value of the line in the image, we can compute the distance in mm to the hand using a third-order polynomial, which accounts for radial distortion in the image. This analysis is performed on the Raspberry Pi using OpenCV running at approximately $2-4ms$ per frame.

Communication between sensors, actuators, and controllers is a key component of this system. Each sensor is responsible for communicating specific data to and from the central Teensy microcontroller. The microcontroller is in turn responsible for communicating commands to the various actuators.

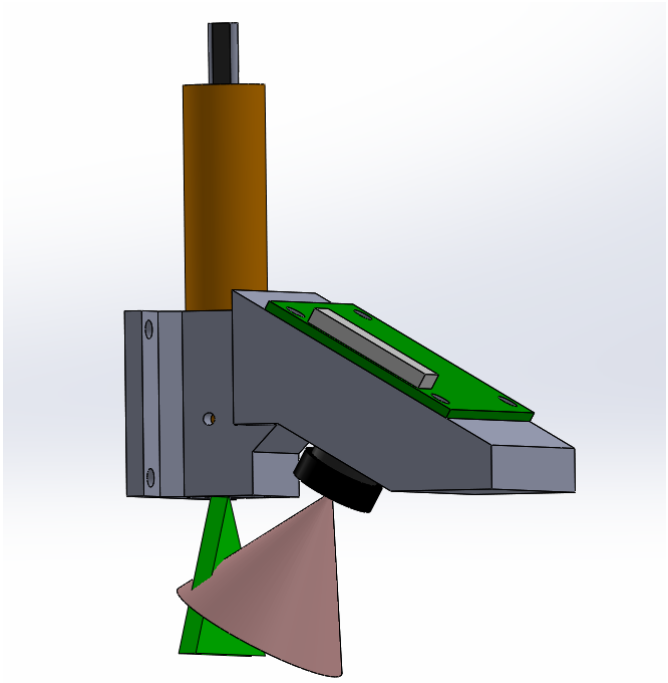


Figure 7.4: Custom Laser Depth Sensor; Green Triangle Laser Field of Projection; Pink Cone Field of View of Camera

7.1.5 Materials

Material selection was a critical design decision. While heated plastic filament hardens once deposited, viscous fluid biomaterials do not typically solidify quickly after deposition and therefore cannot maintain shape. Therefore a material solution was required that could be deposited through the jetting system (given the range of viscosities) and that could be cross-linked either automatically or via an added curing agent. Given these requirements, a sodium alginate solution ($(C_6H_8O_6)_n$) was chosen as the deposition material, with additives of detergent (sodium alkyl sulfates) as a surfactant to allow the solution to keep shape on the workspace prior to cross-linking, and colored food dye for the purpose of visualization and evaluation. Calcium chloride solution ($CaCl_2 \cdot 2H_2O$) was used as the cross-linking agent providing the calcium ions. As is further explained in [149], the free aqueous Ca^+ ions disperse into alginate and displace sodium to promptly cross-link it into a solid hydrogel.

The sodium alginate solution is produced by blending powdered sodium alginate with deionized water and then allowing the mixture to undergo degasification and set. The ratio of powder to water dictates the viscosity of the resulting solution. For the desired viscosity level of the

sodium alginate, we used a ratio of $1g$ powder per $100ml$ water with $2ml$ of aqueous sodium alkyl sulfates and $0.1ml$ of blue food dye #1. For the calcium chloride spray, a ratio of $75g$ per $100ml$ water was used. After the deposition of a given layer of sodium alginate, the aqueous calcium chloride solution was manually applied to the entire build surface with an air brush to cross-link each layer.

The viscosity range of the jetting system and the viscosity of the sodium alginate solution serves well as a surrogate for the eventual bioprinting of cells or other non-alginate bio-inks commonly used in extrusion as in [137, 138].

7.1.6 Software

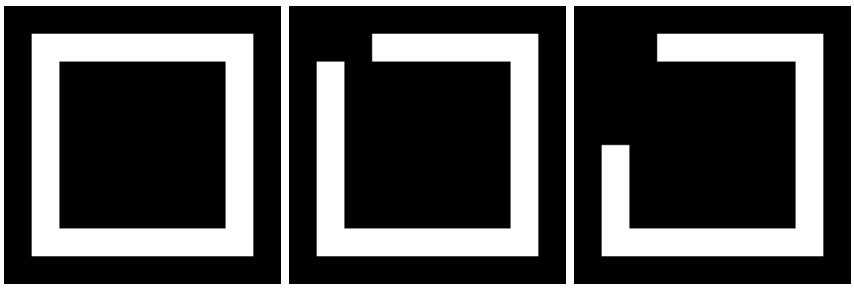
For the tracking, control and logic of the additive manufacturing process, we utilized a custom software stack implemented on a desktop PC running Ubuntu and the Robot Operating System (ROS) [62]. The complete stack consisted of 5 primary components as outlined in Algorithm 7.5, where V_{motor} is a stepper velocity command and C_{jet} is a logic signal to the jetting system.

Algorithm 1: Deposition Control Algorithm

MovingDeposition $(I_{template})$
  inputs: A template of x, y locations $I_{template}$
  foreach *coordinate* $S_i \in I_{template}$ **do**
    while *true* **do**
      $P_t(x, y) \leftarrow currentLeap(x, y)$;
      $P_t(z) \leftarrow currentLaserDepth(z)$;
      $D_t \leftarrow S_i$;
      $E_t \leftarrow P_t - D_t$;
      if $|E_t| < threshold$ **then**
        $V_{motor} \leftarrow (P_t - D_t)K_p$;
        $C_{jet} \leftarrow 0$;
      else
        $V_{motor} \leftarrow V_{min}$;
        $C_{jet} \leftarrow 1$;
        $break$;
      end
    end
  end
  $V_{motor} \leftarrow (0, 0, 0)$;
  $C_{jet} \leftarrow 0$;

Figure 7.5: Deposition Control Algorithm

The first step in this method is to continuously track the position of the anatomy ($P_t = [x, y, z]$) relative to the extrusion head. For this position x and y are sensed via the Leap motion, and z is sensed with the laser distance sensor.



(a) Step 1 (b) Step 2 (c) Step 3

Figure 7.6: Deposition Remaining for a Single Frame

Given the known instantaneous position of the anatomy, we then query a predefined template for the state of that location ($S_i = [x, y, z]$). This template is stored as a series of 2-dimensional binary image matrices with x, y determined by an individual frame and z determined by the frame index. If the corresponding pixel requires material, a low latency command is sent to the jetting system and the corresponding pixel is marked as complete (Figure 7.6). This process is continued until all pixels for the current frame have been completed, and then the next frame is queried.

Determining the optimal trajectory to take for deposition of a single layer requires comparing the end effector's current velocity with the vector to all remaining deposition pixels in the current layer. The end effector's velocity in z is controlled by the laser height sensor through a proportional controller. In x and y the velocity vector is taken as the difference $\Delta P = P_t(x, y) - P_{t-1}(x, y)$ while the vector to remaining pixel i is taken as $V_i = D_i(x, y) - P_t(x, y)$. The difference in bearing between these two headings is computed in Equation 7.1, where α is confined to the interval $[0, 2]$.

$$\alpha = 1 - \Delta P \bullet V_i \tag{7.1}$$

We compute the scaled distance to all remaining deposition pixels as in Equation 7.2, where N_{px} is the width of the binary template matrix so that β is also confined to the interval $[0, 2]$.

$$\beta = 2|V_i|/N_{px} \tag{7.2}$$

All potential targets are evaluated and the target with the lowest cost γ as defined in Equation 7.3 is chosen as D_t.

$$\gamma = \alpha + \beta \tag{7.3}$$

Given the instantaneous desired deposition location (D_t) and the instantaneous anatomy location (P_t) we can compute the end-effector error $E = P_t - D_t$. From this error we compute the required end-effector velocity $V = E * K_p$ where K_p is a gain parameter.

7.1.7 Experimental Design

To evaluate the accuracy and reliability of the proposed system for 3D printing a hydrogel directly onto moving human anatomy, we performed four experiments of increasing complexity.

The first experiment was designed to examine the baseline accuracy of the gantry system and hydrogel material for 2D deposition in a known pattern on a stationary surface. For this experiment a block 'M' pattern (Table 7.1) was deposited on a stationary build plate in a $100x100mm$ area. The deposition pattern was followed in an open loop fashion without tracking of the target substrate position, only encoder feedback of the stepper joints. To provide sufficient material for a complete layer, the pattern was printed three times, spraying the aqueous calcium chloride solution to cross-link between each run.

The second experiment was designed to assess the accuracy of the deposition system in 2D for moving anatomy. For this experiment a hand was placed below the deposition jet and allowed to move freely. Again the block 'M' pattern (Table 7.1) was deposited on the hand while the hand was continuously tracked via the sensing system. Again the pattern was printed three times, spraying the aqueous calcium chloride solution to cross-link between each run.

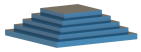
The third experiment was designed to assess the accuracy of this system with regards to multiple layers. For this experiment, multiple layers of a pyramid template were deposited on a stationary build plate. The pyramid was made up of 5 stacked squares, each smaller than the prior step, creating the pyramid template in Table 7.1. Again the pattern was printed three times per layer, spraying the aqueous calcium chloride solution to cross-link between each run.

The fourth experiment was designed to the assess the accuracy of this system with regards to a multi-layer model on a moving hand. For this experiment an unconstrained hand was again placed below the deposition jet. This setup utilized the pyramid model (Table 7.1) from Experiment 3. Each layer was deposited sequentially onto the moving hand and the pattern was printed three times per layer, spraying the aqueous calcium chloride solution to cross-link between each run.

7.1.8 Experimental Evaluation

The deposition pattern was scanned in 2D for Experiments 1 and 2 with a color flatbed scanner at $600dpi$ so that the $100x100mm$ deposition area became an image with dimensions $2362x2362px$. In 3D for Experiments 3 and 4 the deposition pattern was scanned with a 3D scanning system

Table 7.1: Experiment Summary

Experiment	Substrate	Target Object Geometry	Total Layers
1	Stationary		3
2	Unconstrained		3
3	Stationary		15
4	Unconstrained		15

(Artec Spider, Artec 3D, Luxemburg) with a resolution of $G_s = 0.04mm$. The scanned point cloud of the surface was converted to voxels at $600dpi$ to match Experiments 1 and 2. The scans were then programmatically registered with the template via a rigid transformation to minimize error.

Error was calculated at each pixel or voxel in the template as the deviation in value between the template image and the corresponding pixel in the scanned image. True Positive (TP) locations were those where deposition occurred in the desired location (correct deposit). False Negative (FN) locations were those where deposition was desired but did not occur (missing deposit). False Positive (FP) locations were those where deposition occurred but was not desired (incorrect deposit). The TP and FN values were used to compute both the True Positive Rate (TPR) and False Negative Rate (FNR) (Equations 7.4 and 7.5 respectively) where the total number of desired pixels or voxels (N_t) was used as the total Condition Positive. The FP value was used to compute the False Discovery Rate (FDR) (Equation 7.6), where the number of deposited pixels or voxels (N_s) was used as the total Test Outcome Positive. The average error between the outer surface of the template and the surface of the scan was also used as a performance metric to give an outer-shell accuracy in mm.

$$TPR = \frac{\sum(I_{template} \cap I_{scan})}{\sum(I_{template})} \tag{7.4}$$

$$FNR = \frac{\sum(I_{template} \cap \neg I_{scan})}{\sum(I_{template})} \tag{7.5}$$

$$FDR = \frac{\sum(\neg I_{template} \cap I_{scan})}{\sum(I_{scan})} \tag{7.6}$$

Registration between the 3D scan of the resultant deposition and the 3D template was achieved via an Iterative Closest Point (ICP) correspondence. Error was calculated as the deviation between each point in the 3D scan and the corresponding point in the 3D template (Equation 7.7). The TP, FN, and FP measures were computed in 3D between the scanned structure and the pyramid model scaled to the height of the scan. Given a discrete x, y location in the 3D scan, TP locations were those where the z height was the same as the model, FN locations were those where too little material was deposited, and FP locations were those where too much material was deposited. These values were then used to compute the TPR, FNR, and FDR rates (similar to Equations 7.4-7.6). For each x, y scan location, the z height of the scan was compared with the corresponding z height in the model. The height difference was multiplied by G_s which represents the voxel size dictated by the 3D scan resolution. In this case $G_s = 0.04mm$. This provides a volumetric representation of the difference between desired geometry and scan. The mean layer height for the complete structure was calculated as the total height divided by the number of layers.

$$E = \sum_{x=1}^{m}\sum_{y=1}^{n}(Z_{template}(x,y) - Z_{scan}(x,y))G_s^2 \tag{7.7}$$

7.1.9 Results

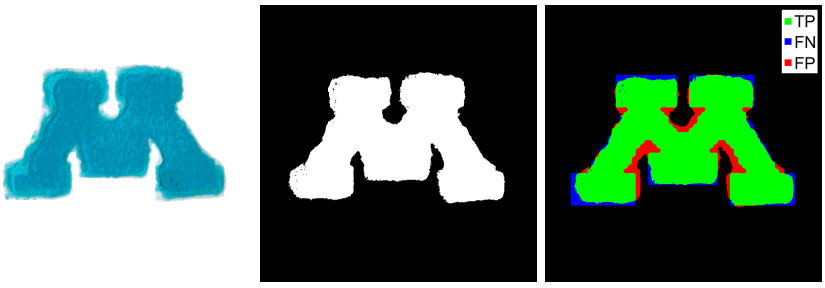
7.1.10 Experiment 1: 2D Stationary Deposition

The baseline printing accuracy for the proposed gantry and deposition system was assessed by depositing the template from Table 7.1 onto a stationary build plate (Figure 7.7). The resultant deposition, threshold image, and comparison is given in Figure 7.8. The accuracy and other metrics are given in Table 7.2.

7.1.11 Experiment 2: 2D Unconstrained

The second experiment was designed to assess the accuracy with which the proposed system could deposit the template image on an unconstrained hand. The hand moved at an average velocity of $5mm/s$ up to a maximum of $25mm/s$. We again utilized the template image from Table 7.1. The resultant deposition, threshold image, and comparison is given in Figure 7.9. The accuracy and other metrics are given in Table 7.2.

Figure 7.7: 2D Deposition Resultant



(a) Scan (b) Threshold (c) Comparison

Figure 7.8: Exp. 1, 2D Stationary Deposition Comparison

7.1.12 Experiment 3: 3D Stationary

The third experiment was designed to assess the accuracy of the proposed system while depositing a multiple-layer model on a stationary build plate utilizing the pyramid template from Table 7.1. The resultant scan of the deposition is given in Figure 7.10a. An error map indicating the regions where incorrect deposition occurred is given in Figure 7.10b. The average deposition layer height for this template was $0.79mm$. The accuracy and other metrics are given in Table 7.2.

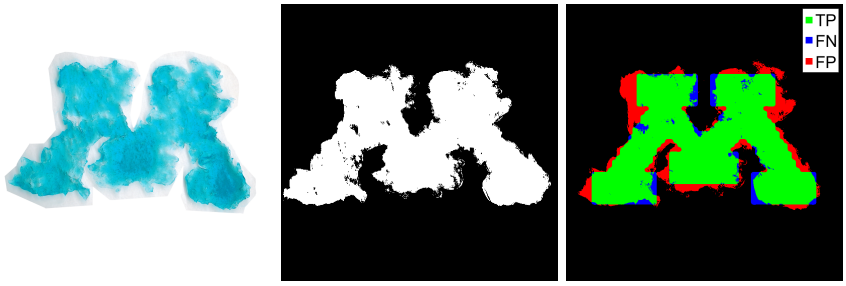
7.1.13 Experiment 4: 3D Unconstrained

The fourth experiment, 3D Dynamic, was designed to assess the accuracy with which the proposed system could deposit a multi-layer template onto an unconstrained hand moving similarly to Experiment 1. We utilized the sequence of template images for the pyramid model (Table 7.1). The 3D scan and comparison is given in Figure 7.11. The accuracy and other metrics are given in Table 7.2.

Table 7.2: Experimental Results

Experiment	TPR (%)	FNR (%)	FDR (%)	Mean Error (mm)	Time (min)
1: 2D Stationary	94.9	5.1	10.8	0.50	11.0
2: 2D Unconstrained	92.6	7.3	17.7	0.73	16.1
3: 3D Stationary	91.5	8.5	38.2	0.95	37.3
4: 3D Unconstrained	87.8	12.3	36.1	1.60	40.9

* TPR: True Positive Rate (correct deposit), FNR: False Negative Rate (missing deposit), FDR: False Discovery Rate (incorrect deposit)



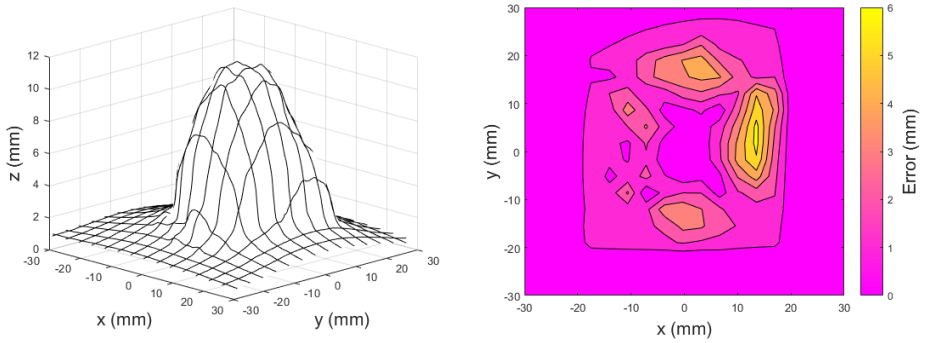
(a) Scan (b) Threshold (c) Comparison

Figure 7.9: Exp. 2, 2D Unconstrained Comparison

7.1.14 Discussion

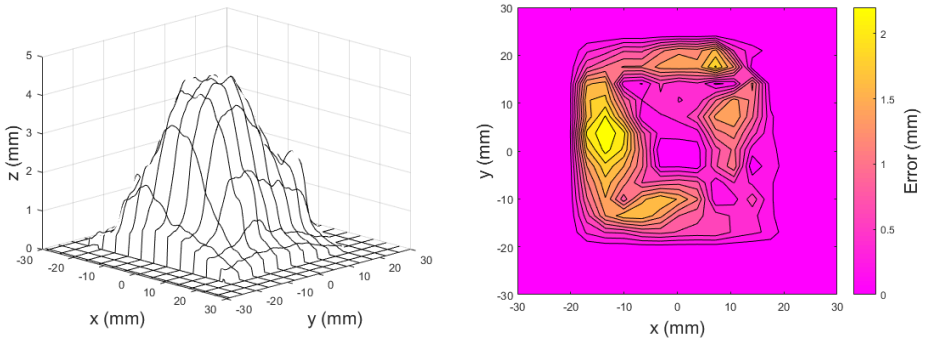
We presented a system for successfully bioprinting desired alginate hydrogel geometries onto unfixtured, moving substrates. To our knowledge this is the first successful attempt at 3D printing onto moving human anatomy. Our system uses precise piezo-electric jetting deposition of viscous hydrogels that allows decoupling of depostion control timing from motion planning and robotic actuation. For a baseline control case, the system achieved an average error of 0.50 mm with 94.9% overall deposition accuracy in a 2D analysis of a planar stationary task (Experiment 1). When the same geometry was printed on an unconstrained hand (Experiment 2), average positional error increased by 0.33 mm (46%) and overall accuracy dropped only slightly by 2.5%. This is a favorable result particularly in light of the unconstrained, natural motion of the hand. Print times were comparable to typical 3D printing speeds for these volumes and did not increase dramatically with the introduction of motion.

The change from 2D to 3D stationary cases (adding 15 layers in Experiment 3 compared with



(a) 3D Scan (b) Error Map

Figure 7.10: Exp. 3, 3D Stationary Comparison



(a) 3D Scan (b) Error Map

Figure 7.11: Exp. 4, 3D Unconstrained Comparison

just 3 layers in Experiment 1) saw significantly larger increases in error compared to changing from 2D stationary (Experiment 1) to 2D dynamic cases (Experiment 2). This underscores some of the accuracy limitations of the selected material deposition and cross-linking processes, independent of substrate motion. Notably, the deposited alginate solution remains fluid and has ample time to flow away before calcium chloride solution is applied and Ca^{+} ions diffuse through the fluid to displace sodium and cross-link the gel into a solid. This is observable in the bunching "roll off" of material indicated by lighter regions in Figures 7.10b and 7.11b and visible as non-flat sides in Figures 7.7 and 7.1.

In this approach we have used a proven bioprinting-compatible alginate hydrogel solution which has viscosities orders of magnitude higher than traditional low viscosity ($\approx$ 1 centipoises) aqueous inkjets. Our jetting hardware can easily scale to a wide viscosity range yet maintain

cell viability as evident in [150]. In principle, this implies that our system is immediately compatible with a wide range of existing bio-inks in widespread use and commonly available to bioprinting research, not just hydrogels (e.g. vendors like biobots). This includes photo-curable or temperature curable inks with simple modifications in our hardware.

There are several limitations in this work. We did not print viable bio-inks and confirm cell survival, though it is well established in the literature with jetting. We do not implement closed-loop sensing and deposition of the material geometry. This is in principle quite feasible with our laser distance sensing and will be explored in future work. However, despite the delays in cross linking, the resulting errors are not substantial. Our deposition of aqueous calcium chloride via manual airbrush is a source of error, delay, and variation between experiments. It would be simple to automate this procedure with an additional jetting head (such as in [146]), but we plan to pursue photocurable materials in the future due to the quicker response time. The desired geometries we printed were unrealistic for medical applications. While they demonstrated programmability of desired geometries, we did not include straight walls or overhanging features—items particularly difficult to 3D print with fluids without, for example, changing orientation relative to gravity. Future work will include adapting our sensing technology to track more varied human anatomy including localized stretch, allowing this system to deposit material onto tissues and organs. We intend to coordinate with researchers from tissue engineering groups to utilize viable bio-inks within our system and explore its accuracy for more complex, realistic geometries.

7.1.15 Conclusion

This work has demonstrated the feasibility of robotically depositing bioprinting compatible materials directly onto unconstrained, moving human anatomy with submillimeter average error rates for 2D surfaces and millimeter-range average error rates for 3D geometries.

7.2 Comparison of Bio-Inks for 3D Bioprinting Directly Onto Moving Human Anatomy

This section is a reproduction of a paper submitted to the 2017 Hamlyn Symposium on Medical Robotics.

7.2.1 Introduction

Advances in bioprinting technology have permitted synthetic tissue, organ, and skin construction using additive manufacturing techniques [137, 139]. The most common bioprinting approach

involves depositing hydrogel solutions embedded with Bio-Inks via pressure driven syringes [142] or via inkjetting [140]. Inkjet approaches are a viable alternative since they do not damage the cell yet permit high-speed control.

Prior art has demonstrated the benefits of bioprinting for tissue engineering [141]. However, prior art has been limited to open loop trajectories on planar, stationary surfaces. This is sufficient for labaoratory settings where culture dishes are utilized or subjects can be sedated and the anatomy fixtured. In some clinical settings it may be neccesary to deposit Bio-Inks onto moving anatomy such as an unfixtured hand of a burn patient that must move during therapy to maintain range of motion for skin grafts. Alternatively, a hand-held precision bioprinting tool (Figure 7.12) may move relative to patient anatomy.

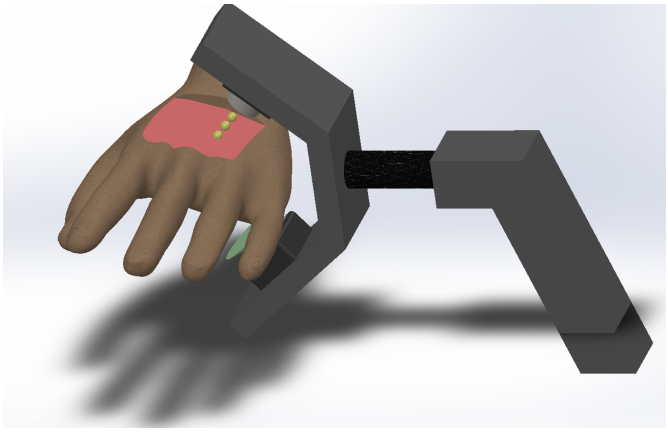


Figure 7.12: Conceptual design of additive manufacturing directly onto moving human anatomy

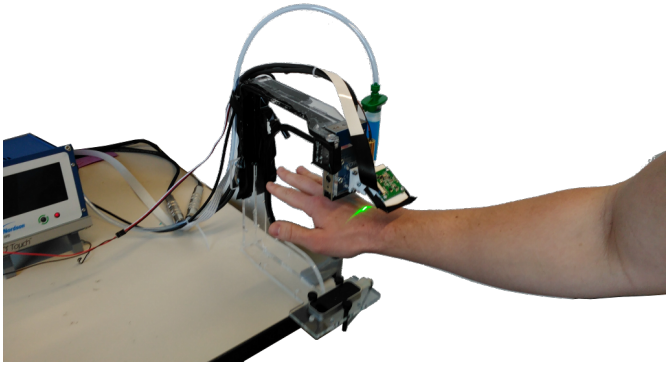
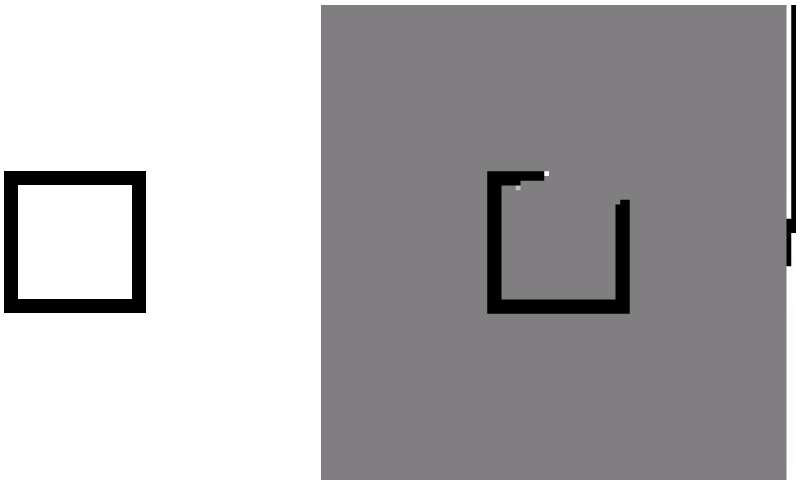


Figure 7.13: Experimental setup, showing users hand below the PICO Plse and above the Leap Motion



(a) 3D Scan (b) Error Map

Figure 7.14: The target template (Left) and an in-progress user view (Right). 1 pixel = 1mm width.

The gap in prior art has been the demonstration of an additive manufacturing technique capable of depositing and adhering viable biomaterials directly onto unconstrained, non-planar, moving anatomy. The objective of this paper is to demonstrate the feasibility of robotically depositing and adhering bioprinting-compatible materials. We evaluate two Bio-Inks for their accuracy and adhesion during bioprinting directly onto moving human anatomy.

7.2.2 Materials and Methods

The Leap Motion was used to track the human hand at 120Hz. A Nordson EFD PICO Plse piezo jetting system was used to propel the fluid onto the hand when it was in the correct position.

The system ran on an Ubuntu PC running ROS, with a graphical display of the current position of the hand relative to the remaining pixels in the pattern with a suggested target highlighted (Figure 7.14), as well as a height bar showing the acceptable distance from the hand to the jet (1cm). The system allowed either the user to move their hand relative to the system, or an operator to move the system relative to the user's hand.

Two hydrogels were used. The first consisted of Sodium Algenate that was deposited onto the hand, and then an aqueous Calcium Chloride solution was airbrushed onto the hand to crosslink into a Calcium Algenate hydrogel. These hydrogels are biocompatible and are used as scaffold for bioprinting [149].

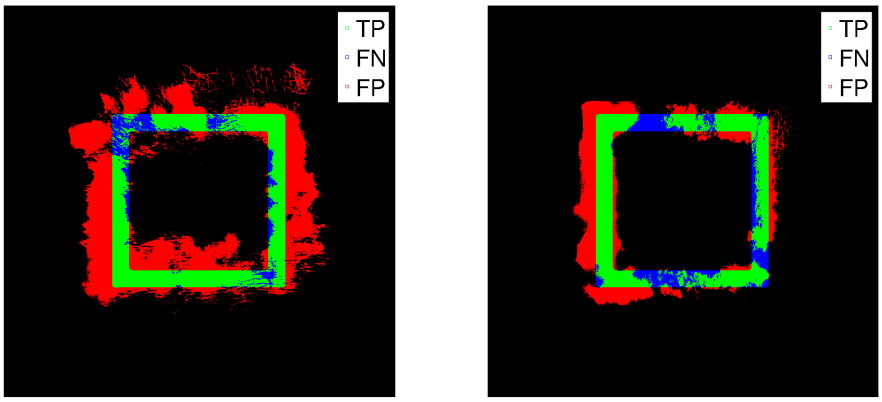
A second hydrogel was created with deionized water containing 10% GelMA (gelatin methacrylate) and 0.5% LAP (lithium phenyl-2,4,6-trimethylbenzoyl-phosphinate) as a photoinitiator to allow use of an ultraviolet flashlight to crosslink between layers. This natural Bio-Ink is also proven to work with a variety of cell types [151].

Blue food dye was used to color the hydrogels to allow for a computer vision based evaluation of the 2D accuracy from scans of the finished gels on a flatbed scanner at 600 DPI (Figure 7.15). The scans were then compared to the target template (Figure 7.14) to determine True Positive (TP), False Negative (FN) and False Positive (FP) areas. These values were then used to determine True Positive Rate (TPR), False Negative Rate (FNR) and False Discovery Rate (FDR) to provide metrics that were independent of the template image.

Figure 7.15: Raw scan of hydrogel on hand.

7.2.3 Results

The robotic system functioned properly, and the user was able to ink at 1.8 pixels per second, leading to an average layer time of 182 seconds.



(a) 3D Scan (b) Error Map

Figure 7.16: True Positive (TP), False Negative (FN) and False Positive (FP) areas for the Calcium Alginate Hydrogel (Left) and the GelMA Hydrogel (Right)

	Bio Compatibility	CrossLink	Adhesion	TPR	FDR
Alginate	Good	Slow	Poor	75%	46%
GelMA	Good	Fast	Good	82%	65%

Table 7.3: Comparison of Calcium Alginate to GelMA Hydrogel.

The Calcium Algenate provided moderate layer to layer adhesion and poor adhesion to the skin of the hand, being easily peeled away by even stretching the underlying skin. The GelMA hydrogel provided good layer to layer adhesion, as well as good adhesion to the underlying skin. Table 7.3 shows True Positive Rate (TPR) and False Discovery Rate (FDR) for both gels.

7.2.4 Discussion

Both hydrogels tested show promise as bio-inks for additive manufacturing on moving anatomy, however both show the need for further development. The Calcium Algenates requirement of the aqueous Calcium Chloride caused the gel to run, and the crosslinking was incomplete leading to poor layer adhesion. The GelMA hydrogel became too warm in the PICO Plse thus becoming too liquid, causing the gel to run and not hold shape, leading to a large False Positive area. If this were adequately addressed (eg by controlling viscosity via temperature or an additive like glycerol) GelMA would prover superior to Algenate for this application.

Chapter 8

Conclusion

8.1 Limitations and Future Work

The experiments presented here were heavily dependent upon the geometry studied, as well as the quality of the scanner used. Therefore, the results should be interpreted only for the category of object studied, those similar to the hand in size and shape, such as similarly sized human organs with skeletal or other internal structure. However, more feature rich objects are beyond the scope of this work, and these results should not be applied to them because they are uncommon in surgery. Rigid or mostly rigid objects are also beyond the scope of this project, and rigid registration research should apply. Alternative scanning modalities are also beyond the scope of this work, as different scanning modalities could provide additional features, and the noise in other scanning modalities may not be as Gaussian as time of flight was shown to be.

Future work related to this thesis should involve finding a more accurate sensor to determine if the newer and better time of flight scanners could provide sufficient quality to find more features. The chip in the real-time scanner used in this study is already five years old as of the writing of this thesis, and considerably more accurate scanners should be commercially available in the future, at which point reevaluating the algorithms could yield new and useful results.

8.2 Conclusion

The experiments laid out above evaluated the algorithms' potential to increase safety of surgery, and show the feasibility of the system in an operating room environment. The experiments also demonstrated and quantified the limits of real time end effector tracking and registration

of soft anatomy. These experiments showed that a nested rigid transform using iterative closest point provided the optimum trade-off of speed and accuracy for both simulated data as well as real-world scans.

From all three of the primary experiments it was clear that the Nested versions of each algorithm provided an improvement in Root Mean Squared Error at the expense of Framerate, for example reducing from $3.6mm$ to $2.0mm$ for Iterative Closest Point in experiment 2. For all three experiments the Fast Point Feature Histogram algorithms vastly underperformed the Iterative Closest Point algorithm in both Framerate and Root Mean Squared Error. This shows that Fast Point Feature Histograms are not suitable for nonrigid tracking of geometric feature poor human tissues.

The secondary experiments validated the theoretical algorithmic complexity and showed that reverse nested matching showed no improvements. The secondary experiments also showed that the Graphics Processing Unit provides enough speed to perform Iterative Closest Point in real-time ($> 10Hz$) and that time of flight depth sensing works through an endoscope.

Given the poor results of the novel algorithms in the primary experiments, additional studies were performed in order to provide alternative contributions to the field of real time robotic tracking for the medical community. The first was the successful invention of a stretchable and flexible Smart Skin for sensing position and force with sub-centimeter accuracy, with possible applications in medical training and prosthetics. Future work in the area of tracking strechable anatomy could immediately benefit from this technology by adding layers of smart skin to synthetic stretchable organs to evaluate the accuracy of tracking algorithms such as those evaluated in this thesis. The second additional study was the development of an Dynamic 3D Bioprinting system for Bioprinting on moving human anatomy, with applications in reconstructive surgery and burn treatment.

References

[1] Linda T Kohn, Janet M Corrigan, Molla S Donaldson, et al. To err is human: building a safer health system, volume 627. National Academies Press, 2000.

[2] Sherry L Murphy, Jiaquan Xu, Kenneth D Kochanek, et al. National vital statistics reports. National vital statistics reports, 60(4):1, 2012.

[3] Chunliu Zhan and Marlene R Miller. Excess length of stay, charges, and mortality attributable to medical injuries during hospitalization. Jama, 290(14):1868–1874, 2003.

[4] Anna Mases, Antonio Montes, Rocio Ramos, Lourdes Trillo, and Margarita M Puig. Injury to the abdominal aorta during laparoscopic surgery: an unusual presentation. Anesthesia & Analgesia, 91(3):561–562, 2000.

[5] Philip A Philips and Joseph F Amaral. Abdominal access complications in laparoscopic surgery. Journal of the American College of Surgeons, 192(4):525–536, 2001.

[6] Atul Gawande. Two hundred years of surgery. New England Journal of Medicine, 366(18):1716–1723, 2012.

[7] James E Allen, Alison F Davis, Wuyang Hu, and Emmanuel Owusu-Amankwah. Residents willingness-to-pay for attributes of rural health care facilities. The Journal of Rural Health, 31(1):7–18, 2015.

[8] Rodney Dockter, Robert Sweet, and Timothy Kowalewski. A fast, low-cost, computer vision approach for tracking surgical tools. In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pages 1984–1989. IEEE, 2014.

[9] Richard M Satava. Surgical robotics: the early chronicles: a personal historical perspective. Surgical Laparoscopy Endoscopy & Percutaneous Techniques, 12(1):6–16, 2002.

[10] Danail Stoyanov, Marco Visentini Scarzanella, Philip Pratt, and Guang-Zhong Yang. Real-time stereo reconstruction in robotically assisted minimally invasive surgery. In International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 275–282. Springer, 2010.

[11] David P Noonan, Peter Mountney, Daniel S Elson, Ara Darzi, and Guang-Zhong Yang. A stereoscopic fibroscope for camera motion and 3d depth recovery during minimally invasive surgery. In Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, pages 4463–4468. IEEE, 2009.

[12] Dean C Barratt, Graeme P Penney, Carolyn SK Chan, Mike Slomczykowski, Timothy J Carter, Philip J Edwards, and David J Hawkes. Self-calibrating 3d-ultrasound-based bone registration for minimally invasive orthopedic surgery. IEEE transactions on medical imaging, 25(3):312–323, 2006.

[13] Peter Mountney, Benny Lo, Surapa Thiemjarus, Danail Stoyanov, and Guang Zhong-Yang. A probabilistic framework for tracking deformable soft tissue in minimally invasive surgery. Medical Image Computing and Computer-Assisted Intervention–MICCAI 2007, pages 34–41, 2007.

[14] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. Computer Vision and Image Understanding, 89(2):114–141, 2003.

[15] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, volume 2, pages 690–696. IEEE, 2000.

[16] Michael J Black and Yaser Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In Computer Vision, 1995. Proceedings., Fifth International Conference on, pages 374–381. IEEE, 1995.

[17] HG Tanner, KJ Kyriakopoulos, and NI Krikelis. Advanced agricultural robots: kinematics and dynamics of multiple mobile manipulators handling non-rigid material. Computers and electronics in agriculture, 31(1):91–105, 2001.

[18] S Davis, JO Gray, and Darwin G Caldwell. An end effector based on the bernoulli principle for handling sliced fruit and vegetables. Robotics and Computer-Integrated Manufacturing, 24(2):249–257, 2008.

[19] WY Lau, CK Leow, and Arthur KC Li. History of endoscopic and laparoscopic surgery. World journal of surgery, 21(4):444–453, 1997.

[20] MJ Mack. Minimally invasive and robotic surgery. JAMA: the journal of the American Medical Association, 285(5):568, 2001.

[21] G.H. Ballantyne. Robotic surgery, telerobotic surgery, telepresence, and telementoring. Surgical Endoscopy, 16(10):1389–1402, 2002.

[22] Gyung Tak Sung and Inderbir S Gill. Robotic laparoscopic surgery: a comparison of the da vinci and zeus systems. Urology, 58(6):893–898, 2001.

[23] B.J. Sandler and S. Horgan. Robotic Surgical Outcomes and Safety, chapter 33, pages 335–346. Springer, 2012.

[24] Francis Michael Longstreth Thompson. Chartered Surveyors: The growth of a profession. Routledge & Kegan Paul Books, 1968.

[25] Merrill I Skolnik. Introduction to radar, chapter 1, page 1990. McGraw-Hill, Incorporated, 1962.

[26] Merril I Skolnik. Fifty years of radar. Proceedings of the IEEE, 73(2):182–197, 1985.

[27] Paul J Besl. Active, optical range imaging sensors. Machine vision and applications, 1(2):127–152, 1988.

[28] Lena Maier-Hein, Peter Mountney, Adrien Bartoli, Haytham Elhawary, D Elson, Anja Groch, Andreas Kolb, Marcos Rodrigues, J Sorger, Stefanie Speidel, et al. Optical techniques for 3d surface reconstruction in computer-assisted laparoscopic surgery. Medical image analysis, 17(8):974–996, 2013.

[29] Oscar G Grasa, Ernesto Bernal, Santiago Casado, Ismael Gil, and JMM Montiel. Visual slam for handheld monocular endoscope. IEEE transactions on medical imaging, 33(1):135–146, 2014.

[30] Robert R Edelman and Steven Warach. Magnetic resonance imaging. New England Journal of Medicine, 328(10):708–716, 1993.

[31] Bradley P Thomas, E Brian Welch, Blake D Niederhauser, William O Whetsell, Adam W Anderson, John C Gore, Malcolm J Avison, and Jeffrey L Creasy. High-resolution 7t mri of the human hippocampus in vivo. Journal of Magnetic Resonance Imaging, 28(5):1266–1272, 2008.

[32] Godfrey N Hounsfield. Computed medical imaging. Journal of computer assisted tomography, 4(5):665–674, 1980.

[33] Willi A Kalender. X-ray computed tomography. Physics in medicine and biology, 51(13):R29, 2006.

[34] Martin J Murphy. Tracking moving organs in real time. Seminars in radiation oncology, 14(1):91–100, 2004.

[35] Hiroki Shirato, Shinichi Shimizu, Tatsuya Kunieda, Kei Kitamura, Marcel van Herk, Kenji Kagei, Takeshi Nishioka, Seiko Hashimoto, Katsuhisa Fujita, Hidefumi Aoyama, et al. Physical aspects of a real-time tumor-tracking system for gated radiotherapy. International Journal of Radiation Oncology* Biology* Physics, 48(4):1187–1195, 2000.

[36] Kazuhiro Katada, Ryoichi Kato, Hirofumi Anno, Yuko Ogura, Sukehiko Koga, Yoshihiro Ida, Motohiko Sato, and Kazuhiko Nonomura. Guidance with real-time ct fluoroscopy: early clinical experience. Radiology, 200(3):851–856, 1996.

[37] Robert A Kane. Intraoperative ultrasonography history, current state of the art, and future directions. Journal of Ultrasound in Medicine, 23(11):1407–1420, 2004.

[38] G Unsgaard, OM Rygh, T Selbekk, TB Müller, F Kolstad, F Lindseth, and TA Nagelhus Hernes. Intra-operative 3d ultrasound in neurosurgery. Acta neurochirurgica, 148(3):235–253, 2006.

[39] Svein Brekke, Charlotte B Ingul, Svein A Aase, and Hans G Torp. Increasing frame rate in ultrasound imaging by temporal morphing using tissue doppler. Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on, 53(5):936–946, 2006.

[40] Jason Geng. Structured-light 3 d surface imaging: a tutorial. Advances in Optics and Photonics, 3(2):128–160, 2011.

[41] J Chow, K Ang, D Lichti, and W Teskey. Performance analysis of a low-cost triangulation-based 3d camera: Microsoft kinect system. In Int. Soc. for Photogrammetry and Remote Sensing Congress (ISPRS), volume 39, page B5, 2012.

[42] Texas Instruments. Introduction to the time-of-flight (tof) system design: User's guide. Technical report, Texas Instruments, 2013.

[43] D Van Nieuwenhove, W Van der Tempel, R Grootjans, and M Kuijk. Time-of-flight optical ranging sensor based on a current assisted photonic demodulator. In Proceedings Symposium IEEE/LEOS Benelux, pages 209–212, 2014.

[44] Jochen Penne, Christian Schaller, Rainer Engelbrecht, Lena Maier-Hein, Bernhard Schmauss, Hans-Peter Meinzer, and Joachim Hornegger. Laparoscopic quantitative 3d

endoscopy for image guided surgery. In Bildverarbeitung für die Medizin, pages 16–20, 2010.

[45] Jochen Penne, Kurt Höller, Michael Stürmer, Thomas Schrauder, Armin Schneider, Rainer Engelbrecht, Hubertus Feußner, Bernhard Schmauss, and Joachim Hornegger. Time-of-flight 3-d endoscopy. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2009, pages 467–474. Springer, 2009.

[46] Thomas Köhler, Sven Haase, Sebastian Bauer, Jakob Wasza, Thomas Kilgus, Lena Maier-Hein, Hubertus Feußner, and Joachim Hornegger. Tof meets rgb: novel multi-sensor super-resolution for hybrid 3-d endoscopy. In International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 139–146. Springer, 2013.

[47] Thomas Köhler, Sven Haase, Sebastian Bauer, Jakob Wasza, Thomas Kilgus, Lena Maier-Hein, Hubertus Feußner, and Joachim Hornegger. Outlier detection for multi-sensor super-resolution in hybrid 3d endoscopy. In Bildverarbeitung für die Medizin 2014, pages 84–89. Springer, 2014.

[48] Sven Haase, Christoph Forman, Thomas Kilgus, Roland Bammer, Lena Maier-Hein, and Joachim Hornegger. Tof/rgb sensor fusion for 3-d endoscopy. Current Medical Imaging Reviews, 9(2):113–119, 2013.

[49] Martin Uecker, Shuo Zhang, Dirk Voit, Alexander Karaus, Klaus-Dietmar Merboldt, and Jens Frahm. Real-time mri at a resolution of 20 ms. NMR in Biomedicine, 23(8):986–994, 2010.

[50] Alexander W Leber, Andreas Knez, Franz von Ziegler, Alexander Becker, Konstantin Nikolaou, Stephan Paul, Bernd Wintersperger, Maximilian Reiser, Christoph R Becker, Gerhard Steinbeck, et al. Quantification of obstructive and nonobstructive coronary lesions by 64-slice computed tomographya comparative study with quantitative coronary angiography and intravascular ultrasound. Journal of the American College of Cardiology, 46(1):147–154, 2005.

[51] Mark S Pearce, Jane A Salotti, Mark P Little, Kieran McHugh, Choonsik Lee, Kwang Pyo Kim, Nicola L Howe, Cecile M Ronckers, Preetha Rajaraman, Alan W Craft, et al. Radiation exposure from ct scans in childhood and subsequent risk of leukaemia and brain tumours: a retrospective cohort study. The Lancet, 380(9840):499–505, 2012.

[52] A Roux, N Bronsard, N Blanchet, and F de Peretti. Can fluoroscopy radiation exposure be measured in minimally invasive trauma surgery? Orthopaedics & Traumatology: Surgery & Research, 97(6):662–667, 2011.

[53] David Geer. Chip makers turn to multicore processors. Computer, 38(5):11–13, 2005.

[54] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. IEEE computational science and engineering, 5(1):46–55, 1998.

[55] Mark D Hill and Michael R Marty. Amdahl's law in the multicore era. Computer, 41(7), 2008.

[56] John D Owens, Mike Houston, David Luebke, Simon Green, John E Stone, and James C Phillips. Gpu computing. Proceedings of the IEEE, 96(5):879–899, 2008.

[57] John Nickolls and William J Dally. The gpu computing era. IEEE micro, 30(2), 2010.

[58] CUDA Nvidia. Compute unified device architecture programming guide. Technical report, NVIDIA Corporation, 2007.

[59] David H Bailey, Eric Barszcz, John T Barton, David S Browning, Robert L Carter, Leonardo Dagum, Rod A Fatoohi, Paul O Frederickson, Thomas A Lasinski, Rob S Schreiber, et al. The nas parallel benchmarks. The International Journal of Supercomputing Applications, 5(3):63–73, 1991.

[60] Chris Gregg and Kim Hazelwood. Where is the data? why you cannot debate cpu vs. gpu performance without the answer. In Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on, pages 134–144. IEEE, 2011.

[61] Jens Wetzl, Oliver Taubmann, Sven Haase, Thomas Köhler, Martin Kraus, and Joachim Hornegger. Gpu-accelerated time-of-flight super-resolution for image-guided surgery. In Bildverarbeitung für die Medizin 2013, pages 21–26. Springer, 2013.

[62] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In ICRA workshop on open source software, volume 3, page 5. Kobe, 2009.

[63] Jonathan Boren and Steve Cousins. Exponential growth of ros [ros topics]. IEEE Robotics & Automation Magazine, 18(1):19–20, 2011.

[64] Oliver Zettinig, Bernhard Fuerst, Risto Kojcev, Marco Esposito, Mehrdad Salehi, Wolfgang Wein, Julia Rackerseder, Edoardo Sinibaldi, Benjamin Frisch, and Nassir Navab. Toward real-time 3d ultrasound registration-based visual servoing for interventional navigation. In Robotics and Automation (ICRA), 2016 IEEE International Conference on, pages 945–950. IEEE, 2016.

[65] Peter Kazanzides, Zihan Chen, Anton Deguet, Gregory S Fischer, Russell H Taylor, and Simon P DiMaio. An open-source research kit for the da vinci® surgical system. In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 6434–6439. IEEE, 2014.

[66] Nikolas Engelhard, Felix Endres, Jürgen Hess, Jürgen Sturm, and Wolfram Burgard. Real-time 3d visual slam with a hand-held rgb-d camera. In Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden, volume 180, pages 1–15, 2011.

[67] Javier Díaz Alonso, Eduardo Ros Vidal, Alexander Rotter, and Martin Muhlenberg. Lane-change decision aid system based on motion-driven vehicle tracking. IEEE Transactions on Vehicular Technology, 57(5):2736–2746, 2008.

[68] HC Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. Nature, 293:133–135, 1981.

[69] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In Computer VisionECCV'94, pages 151–158. Springer, 1994.

[70] JB Maintz and Max A Viergever. A survey of medical image registration. Medical image analysis, 2(1):1–36, 1998.

[71] Gary KL Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C Langbein, Yonghuai Liu, David Marshall, Ralph R Martin, Xian-Fang Sun, and Paul L Rosin. Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. Visualization and Computer Graphics, IEEE Transactions on, 19(7):1199–1217, 2013.

[72] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A review of image denoising algorithms, with a new one. Multiscale Modeling & Simulation, 4(2):490–530, 2005.

[73] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In Robotics-DL tentative, pages 586–606. International Society for Optics and Photonics, 1992.

[74] Michel A Audette, Frank P Ferrie, and Terry M Peters. An algorithmic overview of surface registration techniques for medical imaging. Medical Image Analysis, 4(3):201–217, 2000.

[75] Tommi Tykkala, Cédric Audras, and Andrew I Comport. Direct iterative closest point for real-time visual odometry. In Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pages 2050–2056. IEEE, 2011.

[76] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In Proceedings. Third International Conference on 3-D Digital Imaging and Modeling, pages 145–152. IEEE, 2001.

[77] Dmitry Chetverikov, Dmitry Stepanov, and Pavel Krsek. Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. Image and Vision Computing, 23(3):299–309, 2005.

[78] Charles V Stewart, Chia-Ling Tsai, and Badrinath Roysam. The dual-bootstrap iterative closest point algorithm with application to retinal image registration. Medical Imaging, IEEE Transactions on, 22(11):1379–1394, 2003.

[79] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981.

[80] Andrew W Fitzgibbon. Robust registration of 2d and 3d point sets. Image and Vision Computing, 21(13):1145–1153, 2003.

[81] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. Robotics and Autonomous Systems, 56(11):927–941, 2008.

[82] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pages 1–4. IEEE, 2011.

[83] Victoria J Hodge and Jim Austin. A survey of outlier detection methodologies. Artificial Intelligence Review, 22(2):85–126, 2004.

[84] Peter Lancaster and Kes Salkauskas. Surfaces generated by moving least squares methods. Mathematics of computation, 37(155):141–158, 1981.

[85] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T Silva. Robust moving least-squares fitting with sharp features. ACM Transactions on Graphics (TOG), 24(3):544–552, 2005.

[86] George Vosselman, Ben GH Gorte, George Sithole, and Tahir Rabbani. Recognising structure in laser scanner point clouds. International archives of photogrammetry, remote sensing and spatial information sciences, 46(8):33–38, 2004.

[87] Simone Milani and Giancarlo Calvagno. Joint denoising and interpolation of depth maps for ms kinect sensors. In Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, pages 797–800. IEEE, 2012.

[88] Benjamin Huhle, Timo Schairer, Philipp Jenke, and Wolfgang Straßer. Robust non-local denoising of colored depth data. In Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on, pages 1–7. IEEE, 2008.

[89] Zhou Wang and David Zhang. Progressive switching median filter for the removal of impulse noise from highly corrupted images. Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, 46(1):78–80, 1999.

[90] Michael Elad. On the origin of the bilateral filter and ways to improve it. IEEE Transactions on image processing, 11(10):1141–1151, 2002.

[91] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In Computer Vision, 1998. Sixth International Conference on, pages 839–846. IEEE, 1998.

[92] Li Chen, Hui Lin, and Shutao Li. Depth image enhancement for kinect using region growing and bilateral filter. In Pattern Recognition (ICPR), 2012 21st International Conference on, pages 3070–3073. IEEE, 2012.

[93] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. Design of an image edge detection filter using the sobel operator. IEEE Journal of solid-state circuits, 23(2):358–367, 1988.

[94] WR Crum, T Hartkens, DLG Hill, WR Crum, T Hartkens, and DL Hill. Non-rigid image registration: theory and practice. The British journal of radiology, 77:S140–53, 2003.

[95] Baojun Li, Gary E Christensen, Eric A Hoffman, Geoffrey McLennan, and Joseph M Reinhardt. Establishing a normative atlas of the human lung: intersubject warping and registration of volumetric ct images. Academic Radiology, 10(3):255–265, 2003.

[96] Richard Szeliski and Stéphane Lavallée. Matching 3-d anatomical surfaces with non-rigid deformations using octree-splines. International journal of computer vision, 18(2):171–186, 1996.

[97] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. Image and vision computing, 10(3):145–155, 1992.

[98] Xuming Ge. Non-rigid registration of 3d point clouds under isometric deformation. ISPRS Journal of Photogrammetry and Remote Sensing, 121:192–202, 2016.

[99] Maryam Seif, Huanxiang Lu, Chris Boesch, Mauricio Reyes, and Peter Vermathen. Image registration for triggered and non-triggered dti of the human kidney: Reduced variability of diffusion parameter estimation. Journal of magnetic resonance imaging, 41(5):1228–1235, 2015.

[100] Ralph G Luthardt, Gido Bornemann, Susanne Lemelson, Michael H Walter, and Alfons Hüls. An innovative method for evaluation of the 3-d internal fit of cad/cam crowns fabricated after direct optical versus indirect laser scan digitizing. International Journal of Prosthodontics, 17(6), 2004.

[101] Nick Van Gestel, Steven Cuypers, Philip Bleys, and Jean-Pierre Kruth. A performance evaluation test for laser line scanners on cmms. Optics and Lasers in Engineering, 47(3):336–342, 2009.

[102] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. Geoscientific Model Development, 7(3):1247–1250, 2014.

[103] Fumihiko Ino, Jun Gomita, Yasuhiro Kawasaki, and Kenichi Hagihara. A gpgpu approach for accelerating 2-d/3-d rigid registration of medical images. In International Symposium on Parallel and Distributed Processing and Applications, pages 939–950. Springer, 2006.

[104] Bernardt Duvenhage, JP Delport, and Jason de Villiers. Implementation of the lucas-kanade image registration algorithm on a gpu for 3d computational platform stabilisation. In Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, pages 83–90. ACM, 2010.

[105] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. ACM Transactions on Graphics (TOG), 27(5):126, 2008.

[106] Dominik Neumann, Felix Lugauer, Sebastian Bauer, Jakob Wasza, and Joachim Hornegger. Real-time rgb-d mapping and 3-d modeling on the gpu using the random ball cover data structure. In Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pages 1161–1167. IEEE, 2011.

[107] FC Dunders. On the speed of mental processes. translated by wg koster (1969). Acta psychologica, 30:412–431, 1869.

[108] Larry Li. Time-of-flight camera an introduction. Technical report, Texas Instruments, 2014.

[109] Michael J Cree, Lee V Streeter, Richard M Conroy, and Adrian A Dorrington. Analysis of the softkinetic depthsense for range imaging. In Image Analysis and Recognition, pages 668–675. Springer, 2013.

[110] Zhengyou Zhang. A flexible new technique for camera calibration. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 22(11):1330–1334, 2000.

[111] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. In Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pages 1–8. IEEE, 2007.

[112] Shaoyi Du, Nanning Zheng, Gaofeng Meng, and Zejian Yuan. Affine registration of point sets using icp and ica. Signal Processing Letters, IEEE, 15:689–692, 2008.

[113] Deyuan Qiu, Stefan May, and Andreas Nüchter. Gpu-accelerated nearest neighbor search for 3d registration. In Computer Vision Systems, pages 194–203. Springer, 2009.

[114] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, pages 3212–3217. IEEE, 2009.

[115] George EP Box, Mervin E Muller, et al. A note on the generation of random normal deviates. The annals of mathematical statistics, 29(2):610–611, 1958.

[116] SC Chay, RD Fardo, and M Mazumdar. On using the box-muller transformation with multiplicative congruential pseudo-random number generators. Applied Statistics, pages 132–135, 1975.

[117] Zvi Gutterman, Benny Pinkas, and Tzachy Reinman. Analysis of the linux random number generator. In Security and Privacy, 2006 IEEE Symposium on, pages 15–pp. IEEE, 2006.

[118] Gordon Moore. Cramming more components onto integrated circuits, electronics,(38) 8, 1965.

[119] Robert R Schaller. Moore's law: past, present and future. Spectrum, IEEE, 34(6):52–59, 1997.

[120] Sami Haddadin, Alin Albu-Schäffer, and Gerd Hirzinger. Requirements for safe robots: Measurements, analysis and new insights. The International Journal of Robotics Research, 28(11-12):1507–1527, 2009.

[121] Ravinder S Dahiya, Giorgio Metta, Maurizio Valle, and Giulio Sandini. Tactile sensingfrom humans to humanoids. Robotics, IEEE Transactions on, 26(1):1–20, 2010.

[122] Jong-Ho Kim, Jeong-Il Lee, Hyo-Jik Lee, Yon-Kyu Park, Min-Seok Kim, and Dae-Im Kang. Design of flexible tactile sensor based on three-component force and its fabrication. In IEEE International Conference on Robotics and Automation, 2005.

[123] Jian Hua Shan, Tao Mei, Lei Sun, De Yi Kong, Zheng Yong Zhang, Lin Ni, Max Meng, and Jia Ru Chu. The design and fabrication of a flexible three-dimensional force sensor skin. In IEEE International Conference on Intelligent Robots and Systems, 2005.

[124] John Ulmen and Mark Cutkosky. A robust, low-cost and low-noise artificial skin for human-friendly robots. In IEEE International Conference on Robotics and Automation, 2010.

[125] Y R Wang, J M Zheng, G Y Ren, P H Zhang, and C Xu. A flexible piezoelectric force sensor based on pvdf fabrics. IOP Smart Materials and Structures, 20:1–7, 2011.

[126] Byungjune Choi, Hyouk Ryeol Choi, and Sungchul Kang. Development of tactile sensor for detecting contact force and slip. In IEEE Intelligent Robots and Systems, 2005.

[127] Thomas V. Papakostas, Julian Lima, and Mark Lowe. A large area force sensor for smart skin applications. In Proceedings of IEEE Sensors, 2002.

[128] Jonghwa Park, Youngoh Lee, Jaehyung Hong, Youngsu Lee, Minjeong Ha, Youngdo Jung, Hyuneui Lim, Sung Youb Kim, and Hyunhyub Ko. Tactile-direction-sensitive and stretchable electronic skins based on human-skin-inspired interlocked microstructures. ACS nano, 8(12):12020–12029, 2014.

[129] Rebecca K Kramer, Carmel Majidi, and Robert J Wood. Wearable tactile keypad with stretchable artificial skin. In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pages 1103–1107. IEEE, 2011.

[130] Ganna Pugach, Viacheslav Khomenko, Artem Melnyk, Alexandre Pitti, Patrick Henaff, and Philippe Gaussier. Electronic hardware design of a low cost tactile sensor device for physical human-robot interactions. In IEEE International Scientific Conference Electronics and Nanotechnology, 2013.

[131] Marc-Antoine Lacasse, Vincent Duchaine, and Clement Gosselin. Characterization of the electrical resistance of carbon-black-filled silicone: Application to a flexible and stretchable robot skin. In IEEE International Conference on Robotics and Automation, 2010.

[132] Jason Lu and Timothy M Kowalewski. Flexible, stretchable skin sensors for two-dimensional position tracking in medical simulators. Journal of Medical Devices, 9(2):020927, 2015.

[133] Robin Walz, Zachary Meier, Michael Winek, and Timothy M. Kowalewski. Medical simulators for developing countries via low-cost two-dimensional position tracking. ASME Journal of Medical Devices, 8:1–3, 2014.

[134] John O'Neill, Jason Lu, Rodney Dockter, and Timothy Kowalewski. Practical, stretchable smart skin sensors for contact-aware robots in safe and collaborative interactions. In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pages 624–629. IEEE, 2015.

[135] John J ONeill and Timothy M Kowalewski. Online free anatomy registration via non-contact skeletal tracking for collaborative human/robot interaction in surgical robotics. ASME Journal of Medical Devices, 2014.

[136] Jaemin Kim, Mincheol Lee, Hyung Joon Shim, Roozbeh Ghaffari, Hye Rim Cho, Donghee Son, Yei Hwan Jung, Min Soh, Changsoon Choi, Sungmook Jung, et al. Stretchable silicon nanoribbon electronics for skin prosthesis. Nature communications, 5, 2014.

[137] Sean V Murphy and Anthony Atala. 3D Bioprinting of Tissues and Organs. Nature biotechnology, 32(8):773–785, 2014.

[138] Ibrahim T Ozbolat and Yin Yu. Bioprinting toward organ fabrication: challenges and future trends. IEEE Transactions on Biomedical Engineering, 60(3):691–699, 2013.

[139] Young-Joon Seol, Hyun-Wook Kang, Sang Jin Lee, Anthony Atala, and James J Yoo. Bioprinting technology and its applications. European Journal of Cardio-Thoracic Surgery, page 148, 2014.

[140] Makoto Nakamura, Akiko Kobayashi, Fumio Takagi, Akihiko Watanabe, Yuko Hiruma, Katsuhiro Ohuchi, Yasuhiko Iwasaki, Mikio Horie, Ikuo Morita, and Setsuo Takatani. Biocompatible inkjet printing technique for designed seeding of individual living cells. Tissue engineering, 11(11-12):1658–1666, 2005.

[141] Aleksander Skardal, David Mack, Edi Kapetanovic, Anthony Atala, John D Jackson, James Yoo, and Shay Soker. Bioprinted amniotic fluid-derived stem cells accelerate healing of large skin wounds. Stem cells translational medicine, 1(11):792–802, 2012.

[142] Vivian Lee, Gurtej Singh, John P Trasatti, Chris Bjornsson, Xiawei Xu, Thanh Nga Tran, Seung-Schik Yoo, Guohao Dai, and Pankaj Karande. Design and fabrication of human skin by three-dimensional bioprinting. Tissue Engineering Part C: Methods, 20(6):473–484, 2013.

[143] Xiaofeng Cui, Thomas Boland, Darryl D D'Lima, and Martin K Lotz. Thermal inkjet printing in tissue engineering and regenerative medicine. Recent patents on drug delivery & formulation, 6(2):149–155, 2012.

[144] John J O'Neill and Timothy M Kowalewski. Online free anatomy registration via noncontact skeletal tracking for collaborative human/robot interaction in surgical robotics. Journal of Medical Devices, 8(3):030952, 2014.

[145] Anna French, John O'Neill, and Timothy M Kowalewski. Design of a dynamic additive manufacturing system for use on free-moving human anatomy. Journal of Medical Devices, 10(2):020941, 2016.

[146] Tao Xu, Catalin Baicu, Michael Aho, Michael Zile, and Thomas Boland. Fabrication and characterization of bio-engineered cardiac pseudo tissues. Biofabrication, 1(3):035001, 2009.

[147] Therese Andersen, Pia Auk-Emblem, and Michael Dornish. 3d cell culture in alginate hydrogels. Microarrays, 4(2):133–161, 2015.

[148] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. Sensors, 13:6380–6393, 2013.

[149] Kuen Yong Lee and David J Mooney. Alginate: properties and biomedical applications. Progress in polymer science, 37(1):106–126, 2012.

[150] Sabina Di Risio and Ning Yan. Piezoelectric ink-jet printing of horseradish peroxidase: effect of ink viscosity modifiers on activity. Macromolecular Rapid Communications, 28(18-19):1934–1940, 2007.

[151] Kan Yue, Grissel Trujillo-de Santiago, Mario Moisés Alvarez, Ali Tamayol, Nasim Annabi, and Ali Khademhosseini. Synthesis, properties, and biomedical applications of gelatin methacryloyl (gelma) hydrogels. Biomaterials, 73:254–271, 2015.

Appendix A

Definitions

A.1 Acronyms

Table A.1: Acronyms

Acronym	Meaning
3D	Three Dimensional
ABS	Acrylonitrile Butadiene Styrene
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CORVUS	Complete Operating room Robotics for Virtually Unassisted Surgery
CPU	Central Processing Unit
CT	Computed Topography
CUDA	Compute Unified Device Architecture
DOF	Degrees Of Freedom
FPFH	Fast Point Feature Histograms
gcc	GNU Compiler Collection
GNU	GNU's Not Unix!
GPU	Graphics Processing Unit
ICP	Iterative Closest Point
IR	Infrared
JPEG	Joint Photographic Experts Group

Continued on next page

Table A.1 – continued from previous page

Acronym	Meaning
LIDAR	Light Detection And Ranging
MRI	Magnetic Resonance Imaging
MAE	Mean Absolute Error
MSE	Mean Squared Error
NAN	Not A Number
nvcc	NVIDIA CUDA Compiler
OR	Operating Room
PCL	Point Cloud Library
PNG	Portable Network Graphics
QVGA	Quarter Video Graphics Array
RGB	Red Green Blue
RMSE	Root Mean Squared Error
ROS	Robot Operating System
TOF	Time Of Flight
US	Ultrasonography
XML	Extensible Markup Language

Appendix B

Software Details

B.1 Software Launch Structure

Each experiment is started with a launch file, an XML formatted file that denotes which nodes to start, and what parameters, if any, to set in them. However, in order to easily loop through the different parameters to use the different algorithms, a bash file was created for each experiment to run them in a consistent manner in series. The algorithms were run in series to avoid competition for computational resources. Experiments 1 and 2 were run in the same bash file, as their dependent variables were tested in all permutations for additional analysis.

B.2 Nonrigid Matching

The primary node that ran the algorithm was named `nonrigid_icp` although it did also run the FPFH algorithms, since it was decided that so much code was common between the algorithms. This node subscribed to a `sensor_msgs::PointCloud2` message named `/depth_registered/points`, which could come from any source. The model was loaded on bootup from the files mentioned in the preprocessing section, and the node output metrics in a custom message type which includes RMSE, Rate, and running averages of those. The node also outputs matched point clouds for visualization purposes. The `nonrigid_icp` node contained two classes, `NonRigidMatch` and `PointCloudDB`.

The `PointCloudDB` class contains the structure for saving a nonrigid point cloud model. The primary data structure is a struct called `cloud_db` which is self-referencing to allow any number of children. For the experiments, there were 22 objects created, with 1 at the top level, 7 at the second level and 14 at the third level, which can be seen in Figure B.1. Each `cloud_db` object

Experiment #	Description	ROS Launch File	Bash Script
1	Pixel Size	sk_sim.launch	experiment1-2.sh
2	Added Noise	sk_sim.launch	experiment1-2.sh
3	TOF Sensor	sk_playback.launch	experiment3.sh

Table B.1: Files used to run experiments.

contains the points of the model, as well as the estimated normals. The following list describes all the contents of the object:

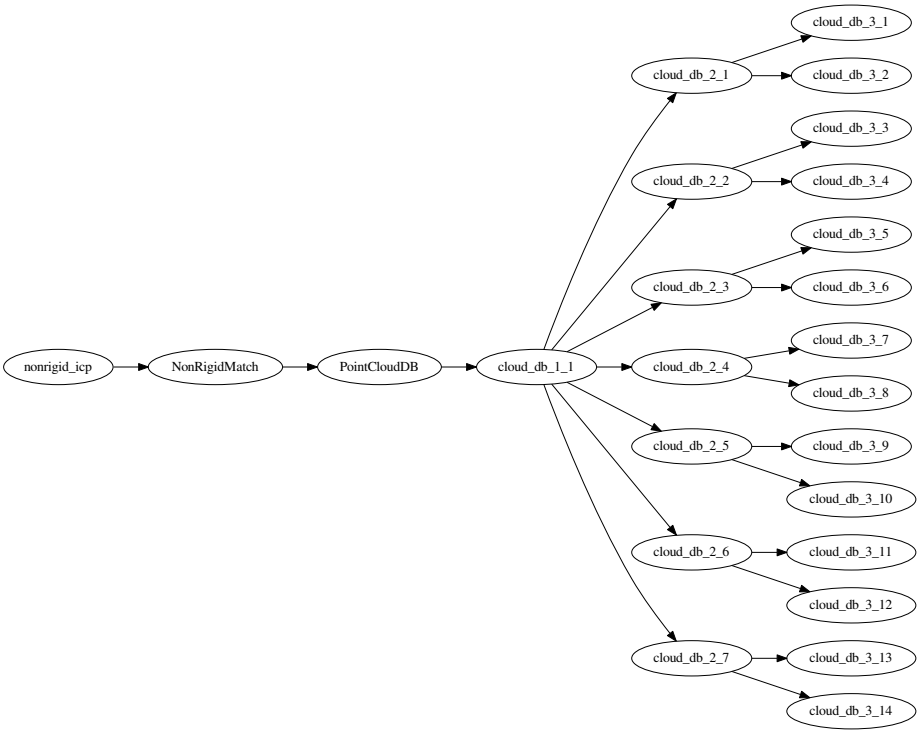


Figure B.1: Nested storage of cloud_db structure within parent classes.

- level: The level of this node, zero indexed
- position: The address of this node, from top to bottom, anything below this node is zero
- name: The overall name of the model. All children should have same name
- filename: The location of the model file
- cloud: The unmodified model, this should be constant
- unfiltered_cloud: The unfiltered cloud, for backup
- scene_normals: The estimated normals for the cloud, for calculating FPFH descriptors
- scene_descriptors: The estimated FPFH descriptors for the cloud
- match_search: The KD-Tree for the cloud, for efficient searching
- fit: The current model fitted to the latest data (Just cloud×trans)
- trans: The transformation matrix of the current fit
- prev_trans: The transformation matrix of the previous fit
- children: Children, which should be limited to only subsets of cloud
- parent: Points to parent object, unless top level
- ancestor: Points to ultimate parent, for easy access

The `cloud_db` is populated upon the loading of a file, which is provided by the `load_model()` function. This looks for the first file specified, then runs recursively to load children until there are no more files to load at that level, or `MAX_LEVELS` is reached.

The `PointCloudDB` class contains a `CloudCat()` function for concatenating the leaves into a colored cloud for visualizing the fit, since the model is transformed to match the scan. The `PointCloudDB` class also contains a `getFitnessScore()` function for evaluating the quality of the fit, which is compared to a ground truth model. This function returns a Mean Squared Error (MSE), but only for nearest neighbors.

The `NonRigidMatch` class contains the actual algorithms for nonrigid registration of point clouds, other than the GPU code which resides in a separate library due to compilation requirements. The class contains a settings struct called `NonRigidMatchSettings` which keeps track of which algorithm is currently being used. The primary function is the recursive function `ICP_tree()`, which calls itself for the nested versions of the algorithms, choosing which algorithm to call based on the settings in `NonRigidMatchSettings`.

The `PCL_ICP()` and `reverse_PCL_ICP()` functions perform the standard iterative closest point algorithm, using the implementation from Point Cloud Library. The main difference between the two is that the reverse changes the target and the source. This should not have a significant effect on the actual fit achieved, but changes the optimization, since it changes which cloud is being matched to which, and therefore which cloud has a KD-Tree made for it. The reverse method has a smaller cloud to build a tree for, but requires the tree be calculated for each leaf, while the forwards method has a larger cloud with a KD-tree, but that tree only need

be calculated once.

The `GPU_ICP()` and `reverse_GPU_ICP()` functions perform the iterative closest point algorithm on the GPU. The GPU code had to be compiled with the Nvidia CUDA compiler, `nvcc`, so the GPU code was made into a class that could be dynamically linked as a library to be used by a ROS node, which uses `gcc` as the compiler. Therefore, all the `GPU_ICP()` and `reverse_GPU_ICP()` functions have to do is copy the data into a format that is able to be copied over to the GPU memory. Also the `IcpParams` have to be set and copied over to the library. The points are manually copied over, to ensure that the memory is organized correctly, and the transformation matrix of the initial guess is manually copied over as well. The data is then transferred to the GPU and the result is returned. The `reverse_GPU_ICP()` function differs in the same way as the `PCL_ICP()` and `reverse_PCL_ICP()` functions, in that the reverse function has to initialize the KD tree for each leaf, not just once for the scan.

The `my_FPFH()` function performs the FPFH matching. There is no reverse version of this function, since the reverse versions of the ICP functions did not show any benefit and the initialization of FPFH is even longer than that of ICP. This function starts by estimating normals for the cloud, if they don't already exist. It was attempted to do this with `IntegralImageNormalEstimation()` for actual scans, since the point cloud corresponds to a depth image, but the `NormalEstimationOMP()` function provided better normals, and is multithreaded with the OpenMP library for speed. The next step of the `my_FPFH()` function is to downsample the cloud if desired, however this was not used since the code was not run in real time, and features were sparse enough already.

The `my_FPFH()` function then calculates the features of the cloud, in order to use those features to match to similar features in the model. This calculation is performed multithreaded using the OpenMP library, although it still takes the majority of the function time. This histogram of features is stored for each point, using the points within a fixed radius around the point to evaluate the features, including slope, concavity, normals, etc.

The `my_FPFH()` function then searches for nearest neighbors in the N-dimensional space of the histograms. If the nearest neighbor is within a threshold, that pairing is added to the correspondence. The N-dimensional search is performed with the use of a KD-Tree to optimize the search.

The final part of the `my_FPFH()` function is to cluster the matching points in order to find a sensible rigid transformation to match the cloud to the model. A 3D Hough grouping is used in order to recognize rotations and translations that will minimize the distance between corresponding points.

The FPFH algorithm was optimized by precalculating all the features of the model upon

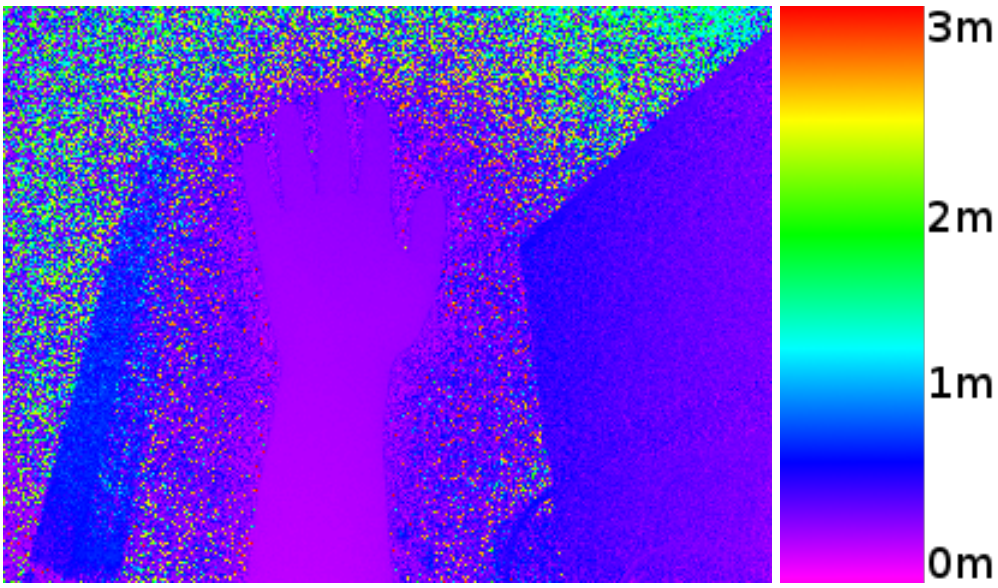


Figure B.2: Unfiltered Depth Image.

loading the model, so as not to have to do it each time. This is performed in the `recalculate_model()` function, which calls itself recursively to cover all layers of the model. The `recalculate_model()` function works the same as the first half of the `my_FPFH()` function, estimating normals if they do not exist, downsampling the cloud if desired, and calculating the features of the cloud. This is as far as the precalculation can go, since the next steps involve matching the model to the cloud.

There is also a `control()` which just applies the prior transformation to the cloud. This is useful in determining if the algorithms are actually providing any real utility, or actually making things worse. Taking the time of this function is not instructive, however, as it is hardly doing anything and will run quickly.

B.3 Depth Image Filter

For experiment 3, a real sensor was used to collect a time of flight image, to be used in matching. The raw image is noisy, as can be seen in Figure B.2, so a number of filters are applied.

A filter that is in the code, but was not used for experiment 3, is a temporal filter. This is an exponential moving average for each pixel, providing an exponential decay for each pixel requiring only one image to be stored in memory between frames. The coefficient of the filter is a parameter that can be varied, however it was not found that the temporal filter improved

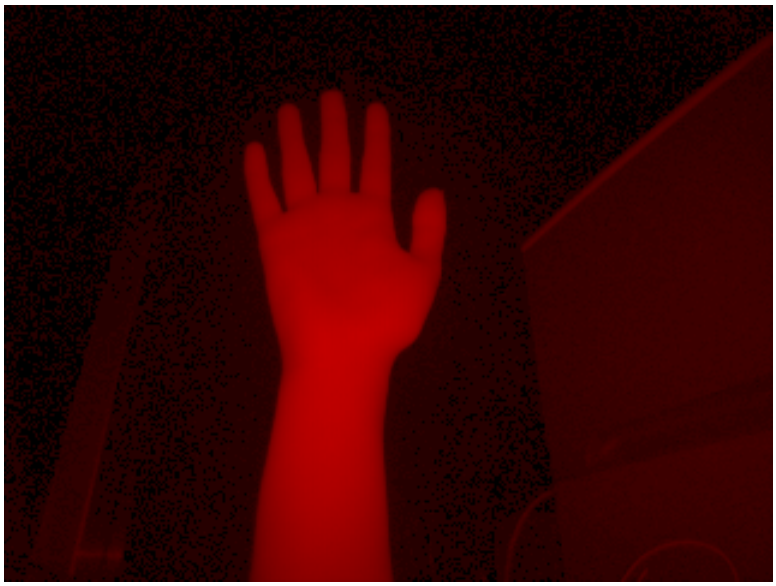


Figure B.3: Raw Confidence Image. Brighter red is more returned IR light.

accuracy of the scan, since it blurred features in addition to providing filtering. Therefore it was decided not to use it for experiment 3. Another filter implemented but not used is the Bilateral filter, which is an edge-preserving non-linear noise-reducing smoothing filter. However, this was slower than the median filter, and did not provide any noticeable benefit, so it was not used in experiment 3.

The median filter was used to use the surrounding pixels to find a more accurate value. The median filter was chosen over the Gaussian blur filter as it preserves sharp edges better, and the scene was expected to have sharp edges, for example the outline of the hand in Figure B.2. The median filter also has be benefit of only pulling values from those actually collected. The only tunable parameter to this filter is the radius used to create the median from, which was set at 5 pixels for experiment 3.

Another filter used was to remove areas with steep gradients, as these areas are either highly noisy, or reflect the actual sharp edge, where the pixels falling off the edge provide unrealistic values. Therefore, a slope was calculated in both X and Y. Any pixel where the sum of the absolute values of the slopes in X and Y was beyond a certain threshold was thrown out. This algorithm had two tunable parameters, the `ksize` for the Sobel gradient operator, which was set to 5 pixels for experiment 3, and the threshold for high slope, which was set to 10 for experiment 3.

The final filter was to remove pixels which had a low confidence. This was possible because

the time of flight sensor, in addition to collecting the average return time at each pixel, also records the approximate number of photons that made it back to that pixel in the chip. This allows a good approximation of confidence, especially for dark objects or objects near or beyond the limit of distance, as they will both register low in confidence. This filter does not help with areas with high specular reflectance. An example of a confidence image is shown in Figure B.3, where far away walls can be seen to be much lower confidence than the hand in the foreground. The confidence threshold is in the scale of 0 to $2^{16} - 1$ for this sensor, but the parameter was scaled to be in the range 0 to 1 for better readability and portability. The value used for experiment 3 was 0.002, which corresponds to 131 in raw units.

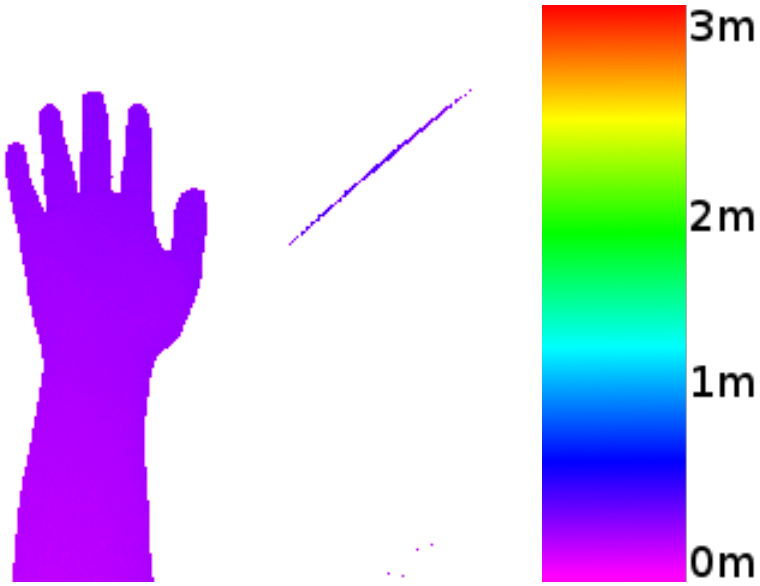


Figure B.4: Filtered Depth Image.

The Depth Image Filter node was set up to receive the unfiltered depth image (example in Figure B.2) and the unfiltered confidence image (example in Figure B.3), while outputting a filtered depth image (example in Figure B.4). This allowed the node to be run either on real-time data or on prerecorded data.

B.4 Point Cloud Simulation

A simulation node was created, which simply loads scan data from a file that was generated by the Artek 3D scanner. Since it was only used in experiments 1 and 2, where the same base scan was used, it is hard-coded to load that file. The code first looks for a binary file, since it will be

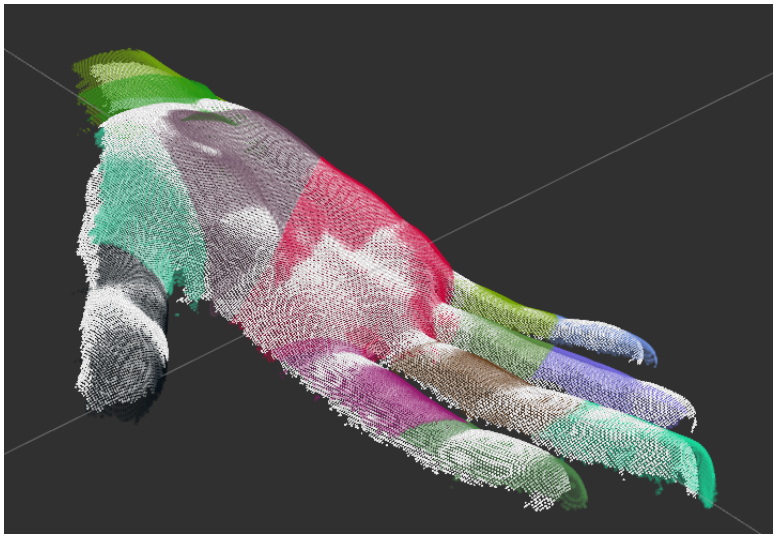


Figure B.5: Example of simulated scan and model. The model is colored, and the simulated scan is white.

faster to load, and will have normals. If that is not available, it reads the ASCII version of the cloud, calculates the normals, then saves that to a binary file. This allows the human readable ASCII file to provide long-term legibility, while the binary file allows more efficient loading for repeated use.

B.5 ROS Binaries

In addition to the custom nodes used, a number of ROS nodes were used off-the-shelf.

Rosbag was used to store the real-time scans for experiment 3 and play them back for analysis by each algorithm, one at a time. Rosbag records any topics timestamped in a binary file.

RViz was used for visualization purposes, especially to manually evaluate fit quality, and to provide sanity checks that the algorithms were performing correctly. RViz was used to visualize the point clouds outputted by the `nonrigid_icp` node, including the colored cloud showing leaf segments. An example of RViz can be seen in Figure B.6.

The ROS Nodelet Manager was used to run the XYZ_Point_Cloud Nodelet, which was used to convert the depth image into a 3D point cloud. The node simply uses the camera calibration matrix to project each pixel out to an X Y Z position. If a pixel was found to be unreliable by the Depth Image Filter node, it was denoted as having the depth NAN which the XYZ_Point_Cloud Nodelet discarded, not including it in the point cloud.

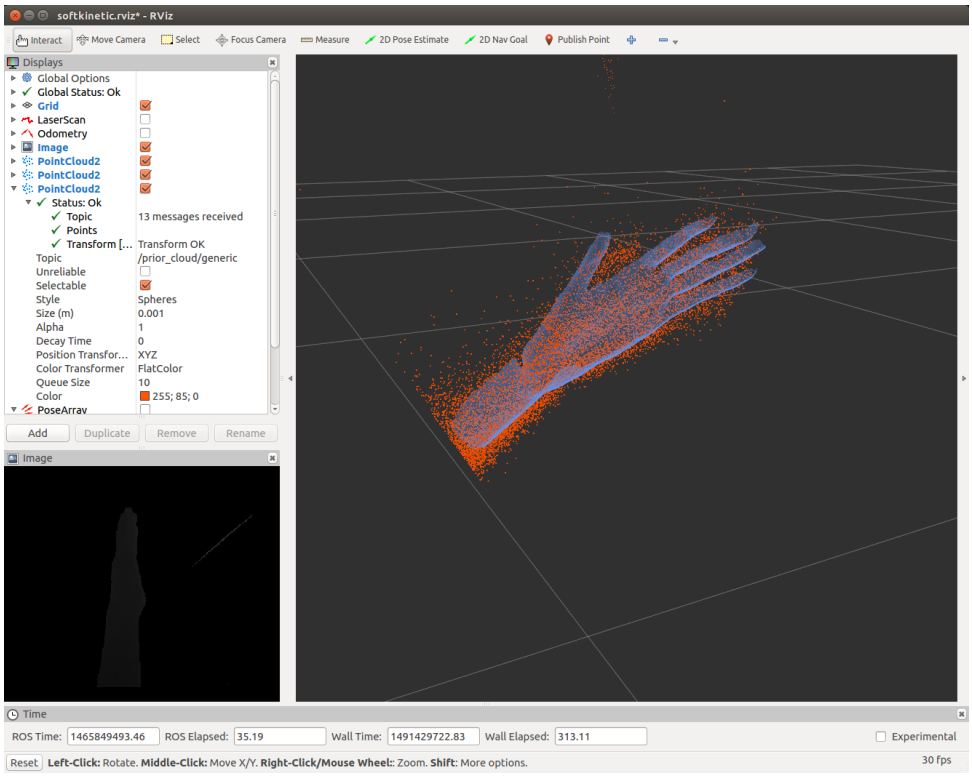


Figure B.6: Example of RViz showing scan and model. The filtered depth image is visible in the lower left.

The ROS Image Transport system was used to be able to store compressed versions of the depth images, confidence images, and RGB images, significantly reducing the bandwidth required and reducing the file size, bringing the size down from over $50MB/sec$ down to $10MB/sec$. The transport system uses JPEG compression for the RGB images, and lossless PNG compression for the depth and confidence images.

A graph showing the connection of nodes during playback for Experiment 3 can be seen in Figure B.7. It should be noted that player is the rosbag playing back the images captured by the time of flight camera, and would be replaced by the DepthSense driver or another depth map source for a real time application.

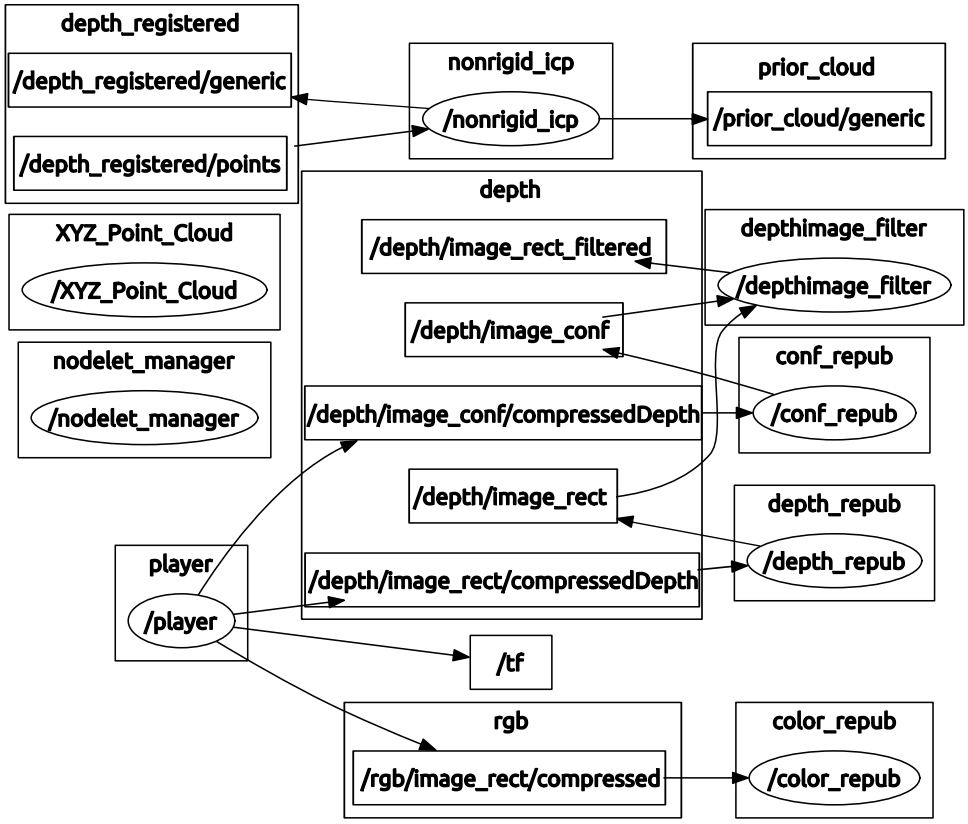


Figure B.7: Node connection graph for Experiment 3.

Appendix C

Biosketch

John received his BSME at the University of Florida in 2009 and worked for two years for GTI Systems Inc. which is a small defense contractor based in Auburndale Florida. While at GTI he worked on ammunition contracts for the Department of Defense. He had the opportunity to work in a wide range of roles, including analyzing quality control data, and collaborating with government officials on failure analyses. He also performed quality control at vendor manufacturing facilities and actively participated in the bidding process for several Department of Defense contracts for 40mm grenades, BDU-48 practice bombs and the M16 upper receiver. He worked with production lines including CNC machining, anodizing and metal finishing, assembly and inspection. He worked on moving a production line to a different facility, which included automated computer vision inspection upgrades and US Government attended prove-outs of all critical defect inspection stations. He experienced all aspects of the defense manufacturing industry in a small company atmosphere before returning to school for his PhD.

John entered the University of Minnesota in 2012. He worked for two semesters on a New Product Design and Business Development project with Medtronic and the Visible Heart Lab creating a an iPad application. The application was created to provide optimal functionality for Medtronic employees, including product pages that integrate various Visible Heart Lab videos, cross-sectional images, and product feature documents as well as a cardiac anatomy section that can display the internal workings of the human heart with Visible Heart inside the heart views. John was the project leader for the second half of the year, and worked on project management, concept design and financial assessments.

John worked on the CORVUS Arm project at the Medical Robotic Devices Lab. The CORVUS Arm (Complete Operating room Robotics for Virtually Unassisted Surgery) is a 6 degree of freedom robotic arm with a PRRRRR setup (a prismatic joint followed by 5 revolute joints in series). The prismatic (linear) joint runs along the side of the operating room (OR)

table from the head to toe. Currently there is one CORVUS Arm on each side of the OR table. This setup allows a reachable workspace that covers the entire patient. The arm could hold any tool at the end, and could also perform tool changes during a surgical operation. John O'Neill worked on it from January 2013 until December 2015, rewriting nearly all the previous code, and contributing nearly 10,000 lines of new code. This code involves the spectrum of low-level hardware programming (CAN-Open, CAN bus, object dictionary) to high level interfaces: e.g. Qt was used to create a UI for debugging and OpenGL was used for visualization. He also upgraded the hardware to run on the ROS framework.

John worked on the Smart Tool project at the Medical Robotic Devices Lab, which was funded by the Army Research Lab. The Smart Tool project aims to create a da Vinci tool driver with torque sensing, with aims to determine tissue properties using only back end sensing. John created electrical and mechanical designs, including custom printed circuit boards. He created a prototype tool used for data collection. John also worked on the Smart Skin project, the 3D Bioprinting project, and Soft Tissue Tracking, which are detailed in this thesis.

John was an invited speaker at the Smart Tissue & Organ Substitutes session at the Design of Medical Devices (DMD) conference in Minneapolis in 2015. John was a presenter at the Get in Touch workshop at the International Conference on Robotics and Automation (ICRA) conference in Seattle in 2015. John was an invited speaker at the Surgical Robots & Computational Surgery session at the Design of Medical Devices (DMD) conference in Minneapolis in 2016. John was awarded the UMII MnDRIVE Graduate Assistantship for January 1, 2016 through August 31, 2017.