2018

High Speed, Micron Precision Scanning Technology for 3D Printing Applications

Nicholas Emord
University of North Florida, n00723236@ospreys.unf.edu

High Speed, Micron Precision Scanning Technology for 3D Printing Applications

by

Nicholas G. Emord

A Thesis submitted to the Department of Electrical Engineering

in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

UNIVERSITY OF NORTH FLORIDA

April 2018

This Thesis titled “High Speed, Micron Precision Scanning Technology for 3D Printing Applications” is approved:

_____ _____

Dr. Juan Aceros
Assistant Professor of Electrical Engineering

_____ _____

Dr. Patrick Kreidl
Assistant Professor of Electrical Engineering

_____ _____

Dr. Peyton Hopson
Adjunct Professor

Accepted for the School of Engineering:

_____ _____

Dr. Osama Jadaan
Director

Accepted for the College of Computing Engineering & Construction:

_____ _____

Dr. Mark Tumeo
Dean

Accepted for the University of North Florida:

_____ _____

Dr. John Kantner
Dean of the Graduate School

DEDICATION

I would like to dedicate my work to my fiancée Kelsey Lorenz.

Without her I would not have grown into the man I am today.

ACKNOWLEDGMENTS

I would like to thank Dr. Juan Aceros, Dr. Patrick Kreidl, Dr. Peyton Hopson, and Dan Preza for their extraordinary support and guidance throughout the thesis process. I would like to thank Kelsey Lorenz for her support and her work editing this thesis.

Regarding the equipment used, I would like to thank everyone at Tetragon Labs for donating one of their harmonic gears. I would like to thank Dr. Nick Hudyma for allowing me to borrow his KEYENCE LJ-V7080 sensor and controller. This project would not have been possible without these specific equipment donations. I would additionally like to thank everyone at the Johnson & Johnson 3D Printing Center of Excellence for all their help and support. Without their contributions I would not have had the opportunity to explore such an interesting engineering field.

Regarding work that was contributed by others, I would like to thank Grzegorz Machura from Lower Silesian Voivodeship, Poland. He has been an invaluable resource regarding all things computer science related and supplied the C# program implementing the sorting algorithm for this project. Finally, I would like to thank Nicholas Salor for preforming the CAD work for the Wuest type gear and the Coupon used in the project.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Modern 3D printing technology is becoming a more viable option for use in industrial manufacturing. As the speed and precision of rapid prototyping technology improves, so too must the 3D scanning and verification technology. Current 3D scanning technology (such as CT Scanners) produce the resolution needed for micron precision inspection. However, the method lacks in speed. Some scans can be multiple gigabytes in size taking several minutes to acquire and process. Especially in high volume manufacturing of 3D printed parts, such delays prohibit the widespread adaptation of 3D scanning technology for quality control. The limiting factors of current technology boil down to computational and processing power along with available sensor resolution and operational frequency. Realizing a 3D scanning system that produces micron precision results within a single minute promises to revolutionize the quality control industry.

The specific 3D scanning method considered in this thesis utilizes a line profile triangulation sensor with high operational frequency, and a high-precision mechanical actuation apparatus for controlling the scan. By syncing the operational frequency of the sensor to the actuation velocity of the apparatus, a 3D point cloud is rapidly acquired. Processing of the data is then performed using MATLAB on contemporary computing hardware, which includes proper point cloud formatting and implementation of the Iterative Closest Point (ICP) algorithm for point cloud stitching. Theoretical and physical experiments are performed to demonstrate the validity of the method. The prototyped system is shown to produce multiple loosely-registered micron precision point clouds of a 3D printed object that are then stitched together to form a full point cloud representative of the original part. This prototype produces micron precision results in approximately 130 seconds, but the experiments illuminate upon the additional investments by which this time could be further reduced to approach the revolutionizing one-minute milestone.

INTRODUCTION

Robust part inspection is the keystone to quality control systems. Traditionally these systems relied solely on the trained eye of a human and simple hand tools to identify and reject defective parts. As technology progressed so too did our ability to quickly detect and reject smaller and smaller defects with the invention of the coordinate measurement machine in the 1950's (COORD3 Metrology, 2014). In our modern digital era, inspection and 3D scanning is dominated by non-contact photon-based sensors and defect detection algorithms that function autonomously. It is here where the cutting edge of 3D part inspection is undergoing innovation, and it is here where this thesis is focused.

As an example, modern 3D scanning technologies, such as CT scanners, produce the resolution needed for micron precision inspection of manufactured parts. However, these technologies are expensive and difficult to streamline in a manufacturing environment. Furthermore, they have a speed limitation, as scanned files can be multiple gigabytes in size and take several minutes to process, making them impractical for rapid quality control of manufactured parts. Hence, there is a real need for a device capable of fast scanning speeds and micron size spatial resolutions.

Quantitatively, the aim of this thesis is to scan a $10cm^2$ surface of a 3D printed object down to a resolution of 5µm and in less than 60 seconds. These benchmark numbers where selected by our industrial partners and represent currently an unreachable combination of parameters by commercially available technologies. However, accomplishing these requirements could revolutionize the quality control industry for high volume manufacturing of 3D printed parts. Thus, the intention of this thesis is to survey the cutting edge of 3D scanning technology and to implement a selected technology and method in a bench top system.

Additionally, this thesis will investigate the processing aspects of 3D scanning relating to iterative closest point (ICP) manipulation to further improve the efficiency of implementing said algorithms into a system. Unlike the research preformed at the University of Alicante (Spain), where researchers have experimented with Chebyshev and Manhattan distance metrics versus the classic

Euclidian metrics (Mora, 2016) to improve the overall efficiency of the (ICP) algorithm, this thesis relates best to the research performed by the Institute of Intelligent Systems for Automation (Italy), where the goal is not to modify the mathematics of ICP, but rather a more efficient implementation method for the ICP algorithm. This includes, for example, a better initial transformation matrix, and/or a "Deletion Mask" referred to as "Crop Stitching". (Marani, 2016)

Thesis Contributions

This thesis investigated a specific set of techniques used as the foundation for the development of a high speed, micron precision 3D scanning system. This thesis complements the body of work for applications of triangulation 3D scanning

(i) by providing a method for precision timing of a triangulation sensor with a mechanical actuation method to achieve micron precision point cloud data acquisition;

and is directly based off the Iterate Closest Point work by (Chen, 1992) and (Besl, 1992) with similar implementation concepts introduced by (Marani, 2016) with the

(ii) development of a computationally efficient array-based point cloud data sorting algorithm,

(iii) and the development of techniques that improve the efficiency of Iterative Closest Point (ICP) algorithms for use in stitching full point cloud models of an object together, such as "Inlier Stitching", "Crop Stitching", and the "Hybrid Stitching" methods.

Thesis Organization

The following thesis presents the development of a system aimed at reaching the aforementioned parameters. The system is based on a triangulation sensor, high precision mechanical actuation, and point cloud image processing. The thesis is organized as follows: "Background" provides an overview of the sensor technology, data mapping, and data manipulation technologies relevant to 3D scanning. "Analysis and Requirements" describes the proposed scanning method, along with the specific requirements for the sensor, the mechanical apparatus, and the overall timing of the system. The "Design" section discusses the conceptual and proof-of-concept designs for the mechanical systems, electrical systems, and algorithms. The "Testing and Validation" section

discusses and mathematically expresses the validity behind mechanical and electronic decisions along a series of functionality tests on the overall 3D scanning system. And finally, the "Conclusion" and "Future Work" sections are presented.

BACKGROUND

This section introduces various technologies typically employed for the inspection of three dimensional objects, including methods to digitally represent such 3D objects. Also provided are brief explanations of the mathematics, techniques, and theory used to manipulate theses digital representations.

Contact Method

Coordinate-Measuring Machine (CMM)

A coordinate measuring machine (CMM) is a device that measures discrete points of geometry on the surface of physical objects. A typical CMM uses the three-dimensional Cartesian coordinate system to navigate its probe along the surface of an object. When the systems probe encounters the surface of an object, the machine samples the current positions of its three-coordinate axis to determine the point-of-contact. This is repeated to generate a 3D point cloud of the object being measured. CMM technology has not changed significantly in the past few decades, although the actuation accuracy continues to improve. One of the best CMM's currently available in the market is Mitutoyo's LEGEX series, as seen in Figure 1, with a resolution of 0.01µm and an accuracy of .35µm. (Mitutoyo, 2018) While highly reliable and accurate, it is also extremely slow.

Figure 1- Mitutoyo LEGEX 774 CMM

Non-contact Methods

Compared to contact methods, non-contact methods are much faster. However, they suffer from a separate set of issues, most of which are induced by optical disturbances. These issues are mainly attributed to the measurement of transparent and reflective surfaces, along with surfaces that may emit low frequency IR.

Time of Flight Sensors

Time of flight sensors use a range finding technique that measures the time that a photon is in flight. This means that the sensor will emit photons at a specified instance, the photons will then travel to object and reflect off its surface. The resulting time the photon takes to get back to the sensing unit is then used to calculate the distance to the measured object by using the known speed of light. Figure 2 below presents this method, where pulses are generated from the source.

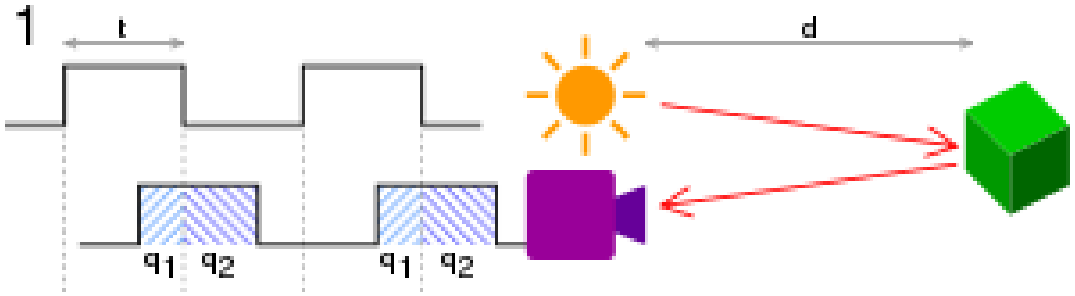


Figure 2 - Time of flight concept

This method is expressed in Equation 1, where (c) is the speed of light, and (t) is the time interval of a received photon pulse emission. The photon sensor registers (q1) and (q2) as both time measurements of receiving exposure.

$$d = \frac{ct}{2} * \frac{q_2}{q_1 + q_2} \qquad eq.1$$

This technique is a derivative of Light Imaging, Detection & Ranging (LIDAR) technology originally developed in the early 1960's for large scale applications. Time of flight sensors can generate point cloud data of 3D geometry with an array detector however, they typically lack in accuracy. This can be attributed to the high speed and the extremely precise high frequency clock

speeds needed to measure photon flight time over small distances. Currently the best point cloud generation resolution for this technology ranges from 250-50µm. (Artec3D, 2018)

Triangulation Sensors

Unlike time of flight sensors, triangulations sensors are only capable of measuring range along the emitted laser. The laser is a 1D line (not point) laser which is cast along the surface of the object and reflects to a sensor. The resulting reflection is representative of a 2D slice of the object's surface that corresponds to depth. By stitching these 2D segments together, a 3D representation of the object is produced. This means the triangulation sensors must be precisely actuated to map the high-resolution 3D geometry of an object.

There are three main components in a triangulation sensor. These include a light source, a pixel array detector, and electronics to control the sensor and calculations. The sensor functions by emitting laser light from a source which is reflected from an object back to a CCD (Charge Coupled Device) or PSD (Position Sensitive Device and/or Position Sensitive Detector) sensor as seen in Figure 3. Depending on the distance of the object (DZ), the reflected light will return to the sensor along a distance of (Dz). This measurement (Dz) is then used in the triangulation calculation.

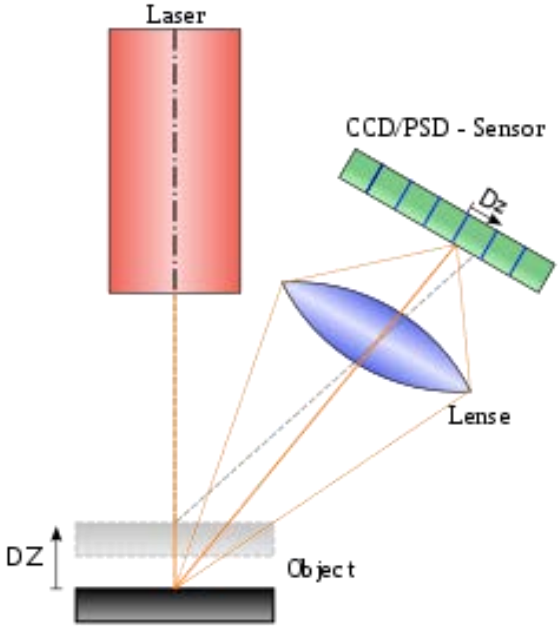


Figure 3 - Fundamental configuration for triangulation sensors

The principal of triangulation is accomplished via basic trigonometry as seen in the Figure 4. The key to determining distance (B) is done by measuring the position of the reflected laser light on the sensor along line (C). The position of the returning light correlates to angle (β) and (γ) though a linear calibration step done by the sensor's manufacturer. Once the angles of the triangles are known, the calculation of distance (B) is then accomplished via the known distance (C) with the Law of Sines as seen in Equation 2.

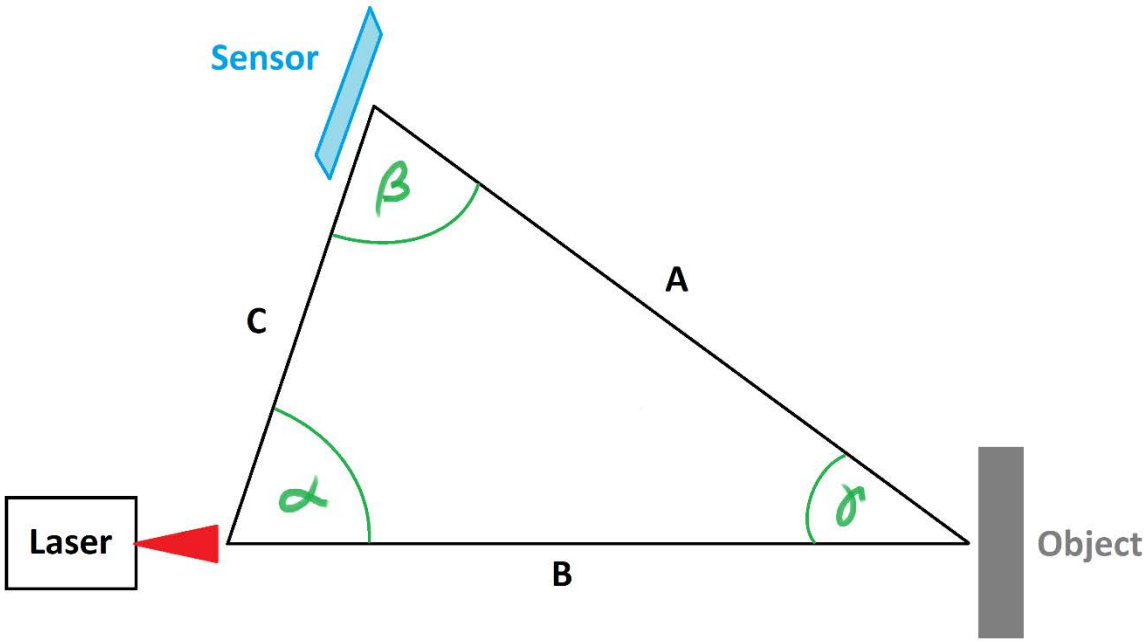


Figure 4 -Triangulation Principal

The distance from the emission point of the laser beam to the object is denoted by (B). The angle at which the laser light reflects off the object and back to the CCD/PSD Sensor is (γ). The angle located by the Sensor is denoted by (β). The known distance between the laser and the detector is denoted as (C).

$$\frac{\sin C}{\gamma} = \frac{\sin B}{\beta} \qquad eq.2$$

While the additional step needed to scan 3D geometry may be cumbersome, the accuracy of current triangulation sensors is the best out of the non-contact methods; a high end sensor's accuracy ranges from 10-50µm in the x-y axis and 0.1-2.0µm in the z-axis. (Keyence, 2018)

Structured Light

Structured light 3D scanners use the same triangulation principle as triangulation sensors. However, scanners using structured light can measure surface topography on a surface simultaneously. This is done by projecting multiple structured light patterns onto an object and then measuring the reflected pattern at a known offset perspective. The structured light pattern reflected to the offset perspective will warp depending on the surface topography of the object. Distance is then triangulated using the same calculations described in the Triangulation Sensors section, Figure 4. The main difference to traditional triangulation sensors is the casting of multiple lines of structured light, thus allowing the sensor to capture 3D geometry simultaneously. The structured light pattern is facilitated by a Digital Light Processing (DLP) projector positioned in-between a stereoscopic pair of cameras. This concept and configuration is best detailed in Figure 5 below. (David Fofi, 2004)

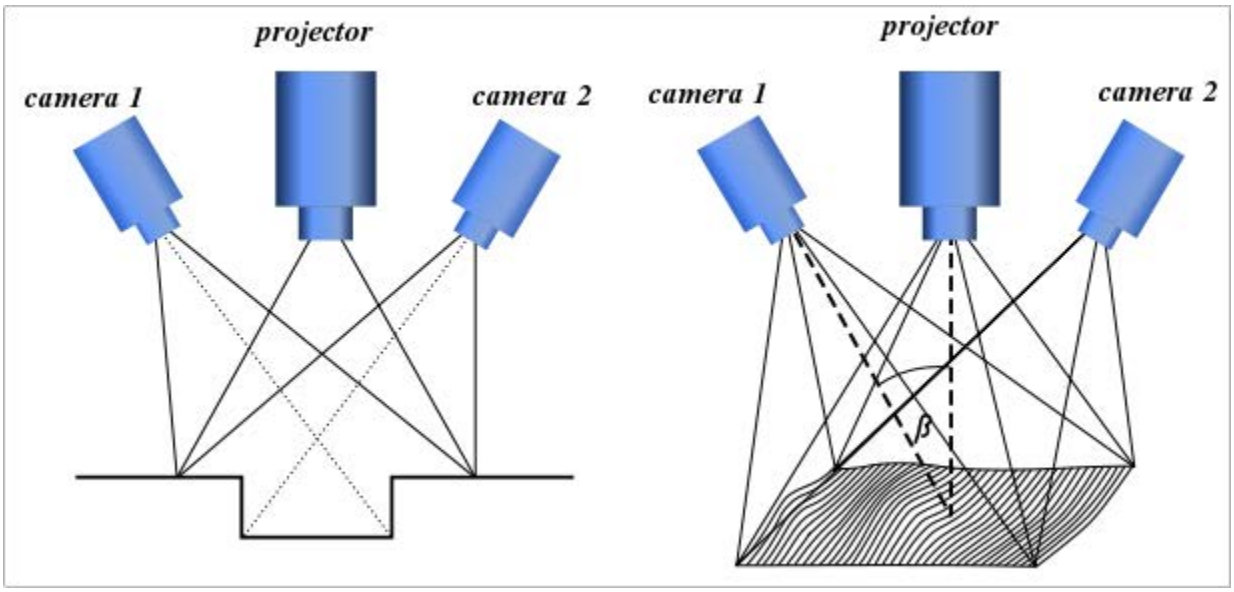


Figure 5 - Structured light sensor configuration

While structured light scanners are exceptionally fast at scanning single surfaces, their accuracy capabilities vary depending upon the field of view and magnification. The KEYENCE VR-3200 in Figure 6 is currently one of the highest end structured light 3D scanners for microscopic applications. While in wide-field mode, it has an accuracy of ±5µm under ideal conditions and is capable of scanning objects with a change in height of 10mm with an area of 6mm x 4.5mm. When

in high magnification mode, the accuracy increases to ±2µm. However, the measurement range shrinks to a height of 1mm with an area of 1.9mm x 1.4mm. (Keyence, 2018)

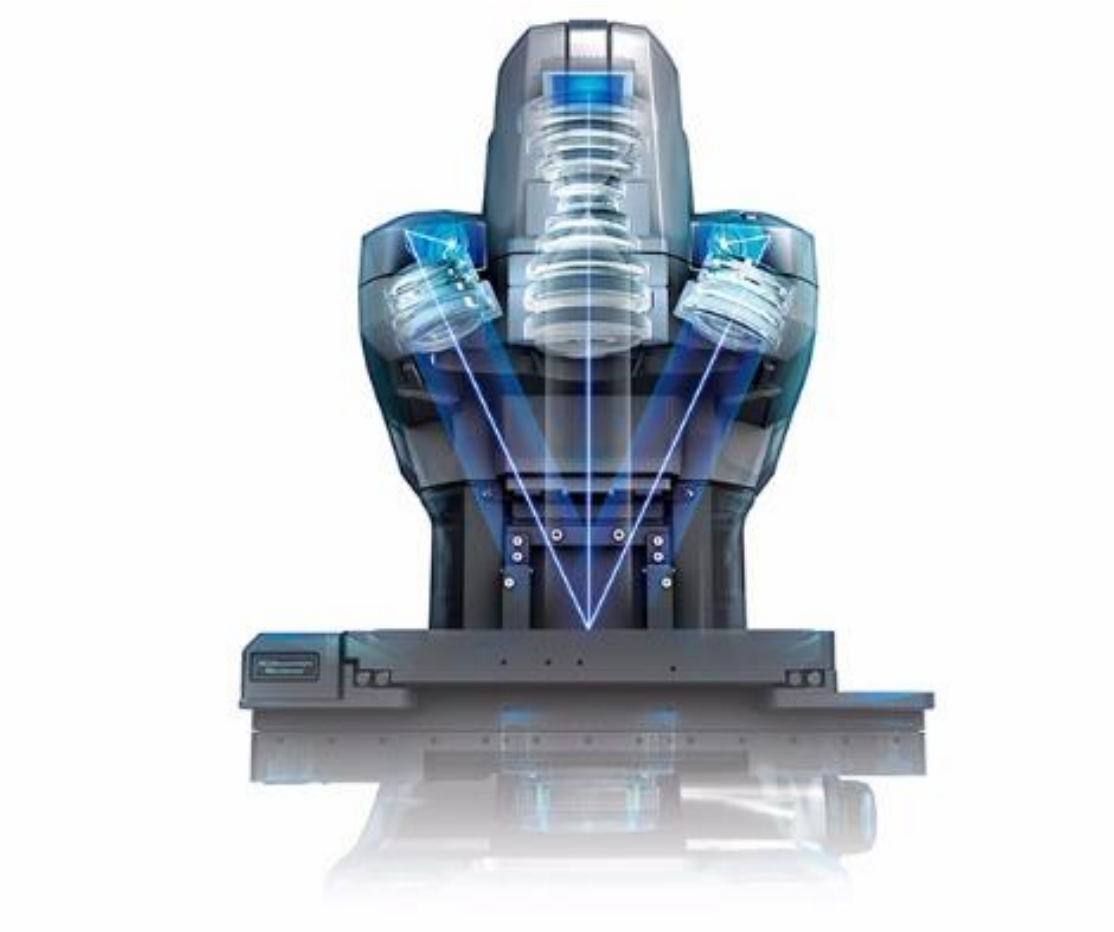


Figure 6 - Keyence VR-3200 Structured Light 3D Scanner

Computed Tomography (CT) Scanners

Computed Tomography (CT) Scanners allow scientists, doctors and engineers to view the internal geometry and densities of objects. Tomography is the process of taking cross-sectional 2D images of a 3D object. The working principle behind a CT scanner is made possible by modern x-ray technology. The basic configuration of a standard CT scanner is seen in Figure 7.

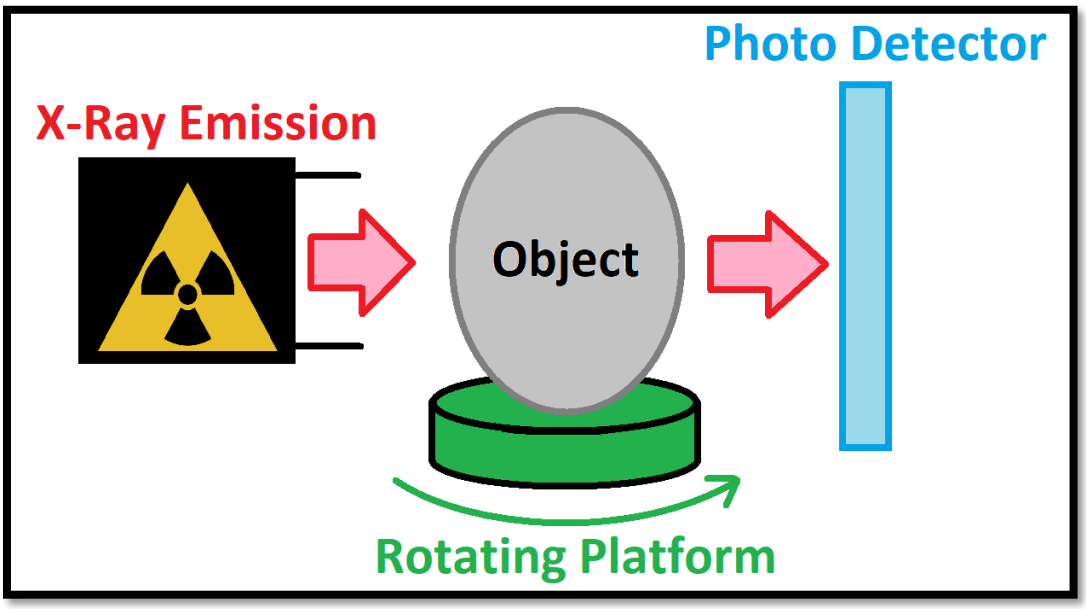


Figure 7 - CT Scanner working principle

On one end of the CT scanner there is an x-ray emission source accompanied on the opposite end by an x-ray photo detector. The photo detector is simply a CMOS (Complementary Metal-Oxide-Semiconductor) or CCD (Charged-Coupled Device) array tuned to detect the x-ray band of the EMF spectrum. (B.S. Verma, 2008) When an object is placed in-between the emitter and detector of the CT scanner, the object's atoms physically deflect or absorb a portion of the emitted photons. This particle interaction changes the incident irradiation intensity on the photo detector's receiving elements. The resulting image captured will correspond to the objects physical geometry along with the objects corresponding incremental density. Once an image is taken, the object is then actuated by a rotating platform, so it can then be imaged again at a different perspective. These steps are repeated until a desired number of images are taken. (Mayo Clinic, 2015) Once all the images have been captured, software then begins the arduous and computationally intensive task

of stitching the images together to form a 3D point cloud of the object. It is interesting to note that the generated point cloud non-binary, meaning that voxels position and intensity correspond to positional geometry and positional density. An example of one of these image segments is shown in Figure 8.

Figure 8 - Cross-sectional view of human skull via CT Scanner. (Ciscle)

Current industrial CT scanners vary based on the intended application. However, the industrial CT scanners relating to this thesis are designed specifically for high resolution inspection of moderate sized parts. For example, the DeskTom CT Scanner seen in Figure 9, is a typical compact CT Scanner with Ultra 3D Accuracy and Resolution. The maximum resolution for these machines is 3µm with an accuracy ranging ±5µm. The main drawback to this technology is the large amount of time needed to complete and process a scan. This can range from five minutes to thirty minutes, depending on the object being inspected. (Laser Design, Inc., 2018)

Figure 9 - DeskTom CT Scanner

Point Cloud Data

Once data points are obtained from a sensor, the next step is to process the data into a form that is useful: the point cloud. A point cloud is a three-dimensional array of points that denote the location of an objects surface through the use of voxels. Voxels are the three-dimensional version of pixels. It is important to note that for the purposes of strictly registering geometry through a point cloud, voxels function as a two-bit system. This means that if a surface is detected at a specific point in 3D space, a voxel is then set to a high value of 1. If surface geometry is not present, then the corresponding voxels will be set to a low value of 0. When viewing a point cloud in a rendered image, the low voxels will then appear as empty space and the high voxels will represent a cloud of points. Figure 10 below shows a point cloud of three human molars. (Afanche Technologies, Inc., 2018)

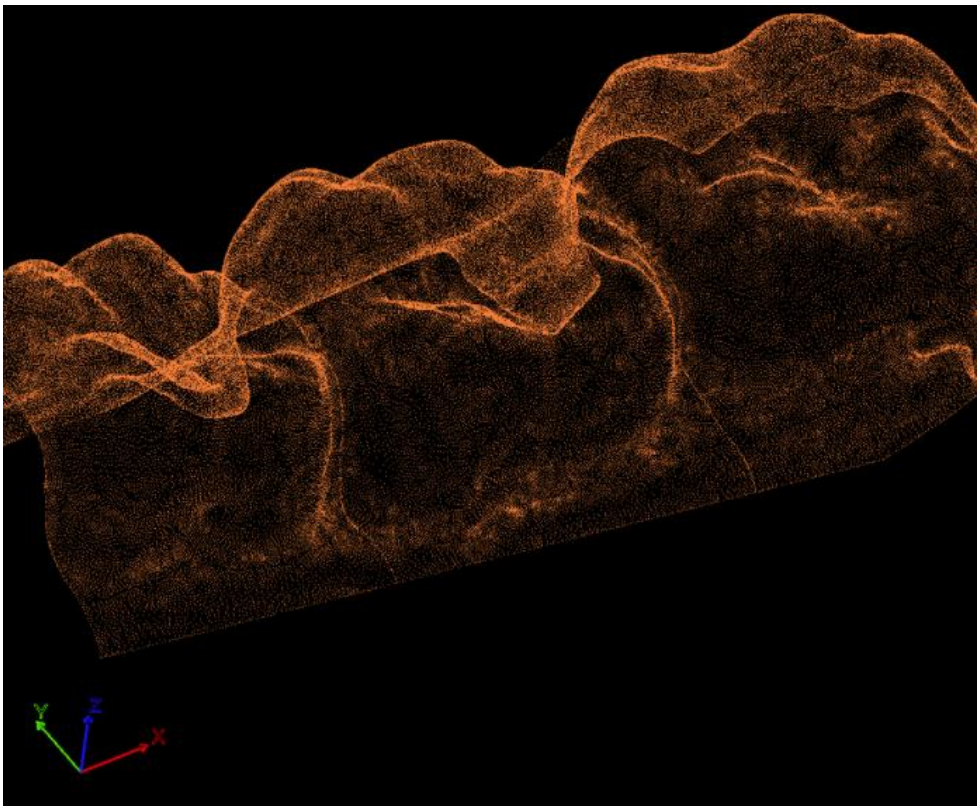


Figure 10 - Point cloud rendering of human molars. (Afanche Technologies, Inc., 2018)

Generation and Manipulation of Point Cloud Files

There is currently no industry standard file type for point clouds, though there are several file types that are used to represent point clouds digitally. However, “The simplest file format supported to store point clouds is a plain text file containing comma separated values or (.csv). Data points are structured in a table format and occupies cells separated into rows and columns. Columns are delimited by commas, tabs, or semicolons while columns are separated via a character return. 2D and 3D point features are supported as well as a limited number of point descriptors. It is common and best practice that the first line of the CSV file be a header identifying each data column. Features are identified by the following names: "x", "y", and "z" if the point cloud is in 3D. Descriptors are divided into columns, and these must be appropriately identified. If the first line does not include a header, the user will be asked to identify which columns correspond to the "x", "y", and "z" feature fields. Any additional content including descriptors is ignored.” (libpointmatcher, 2018)

The main issue with CSV files generated by the Triangulation Sensor being used in the Design portion of this thesis is their output format. The KEYENCE LJ-V Controller outputs a CSV file in the same format that a 2D image would be expressed. That is, corresponding rows and columns designate a pixel location and the corresponding number within the cell is the height of the object in the z-axis. In this format, it is quite simple to generate an 8-bit image by relating the height to pixel intensity. However, to freely convert CSV files with other point cloud files it must be in the specific format as previously explained by Libpointmatcher. The need for the proper CSV format will lead to the development of a formatting algorithm detailed in the Design section of this thesis.

While there are other file formats that represent point clouds such as Visualization Toolkit (.vtk), Polygon File Format (.ply), and Point Cloud Library File Format (.pcd). For the purposes of this thesis, the main area of focus will be on the (.csv) file as described above.

Iterative Closest Point (ICP) Point Cloud Stitching

Iterative Closest Point (ICP) is an algorithm used to close the differences in orientation between two-point clouds of data. This means that if an object is scanned at two different reference positions, the resulting point clouds A and B can be stitched together to form a single point cloud of data represented by matrix M. Mathematically speaking, this is accomplished by solving the orthogonal Procrustes problem as seen in Equation 2.

$$R = argmin_{\Omega}\|\Omega A - B\|_F \quad Subject\ to \quad \Omega^T\Omega = I \qquad eq.2$$

The orthogonal Procrustes problem is a matrix approximation problem. The goal of this problem is to find orthogonal matrix R, which best minimizes the differences between two matrices A and B. The solution to this problem was originally accomplished by Peter Schönemann in his 1964 thesis, "A generalized solution of the orthogonal Procrustes problem". He explains, "The least-squares problem of transforming a given matrix A into a given matrix B by an orthogonal transformation matrix T so that the sums of squares of the residual matrix E = AT – B is a minimum will be called an "Orthogonal Procrustes problem".

Mathematically this problem can be stated as follows.

$$AT = B + E, \qquad eq.3$$

$$TT' = T'T = I, \qquad eq.4$$

$$tr(E'E) = min, \qquad eq.5$$

Where the matrices A and E are both $n \times m$ and over the reals, but otherwise unrestricted, and both are assumed to be "known." In practical work, A will usually be an observed matrix of an earlier study." (Schönemann, 1966)

Equation 3 is referred to as the model for the orthogonal Procrustes problem. Equation 4 is the side condition such that I is the identity matrix of the orthogonal transformation matrix T. Equation 5 states the criterion for successive iteration and is generally expressed as g_1 in Equation 6.

$$g_1 = tr(E'E) = tr(T'A'AT - 2T'A'B + B'B) \quad eq.6$$

The side condition in Equation 4 is generally expressed as g_2 in Equation 7 where L is a matrix of (unknown) Lagrange multipliers.

$$g_2 = tr[L(T'T - I)] \quad eq.7$$

Equation 6 and Equation 7 are then combined to form function g in Equation 8.

$$g = g_1 + g_2 \quad eq.8$$

Partial differentiation of g with respect to (the elements of) T leads to the matrix of partial derivatives" (Schönemann, 1966) as seen in Equation 9.

$$\frac{\partial g}{\partial T} = (A'A + A'A)T - 2A'B + T(L + L') \quad eq.9$$

Equation 9 must be set to zero to meet the extremum case of g_1 as described by minimization in Equation 5. In Schönemann solution, it can be generally understood that the Procrustes problem is equivalent to finding the nearest orthogonal matrix R in Equation 2, to a given matrix M. Where M is equal to matrix B times the transpose of matrix A as seen in Equation 10.

$$M = BA^T \quad eq.10$$

To find the orthogonal matrix R that described the relation of A and B in M, singular value decomposition (SVD) is used in Equation 11 to write Equation 12.

$$M = U\Sigma V^T \quad eq.11$$

$$R = UV^T \quad eq.12$$

The proof for this operation appeared in 1998 from Zhengyou Zhang in "A Flexible New Technique for Camera Calibration" (Zhang, 1998)

While ICP utilizes the fundamental framework set by Schönemann solution to the Procrustes problem, the algorithm is distinct to other methods. This is because ICP's working principal is based on utilizing the correspondence of points within matrixes A and B as a variable to be estimated rather than an input to the Procrustes problem. While there are many variations of the ICP algorithm, the fundamental steps for ICP are as follows:

1. For each individual point within the point cloud A, match the closest point within point cloud B.
2. Estimate a transformation matrix of rotation and translation utilizing the root mean squared minimization technique to best align each individual point to its match from step 1.
3. Apply the transformation matrix obtained in step 2 to point cloud A.
4. Re-associate the new individual point within A to B and repeat cycle.

(Pomerleau, 2015)

ANALYSIS AND REQUIREMENTS

Sensor

As stated previously, the overall goal of this project is to develop a 3D scanning platform that can scan up to 100cm² of an object's surface area down to an accuracy of at least 5µm. The entire process of scanning and processing of data must be done in under a minute. These constraints demand that the sensor must be a non-contact method sensor due to the required speed and accuracy. Of the three non-contact methods, all have comparable accuracies given enough time for exposure. However, the triangulation sensor produced by KEYENCE can acquire data much faster than other current technologies. This is mainly due to its high operational frequency of 1000Hz and wide beam width.

Mechanical Apparatus

Being that the KEYENCE LJ-V sensor must be actuated, a mechanical apparatus must be developed to meet the measurement accuracy and time requirements previously stated. Since the accuracy of measurements in the x and z-axis are inherent to the sensor, the mechanical apparatus must retain and control 5µm of accuracy in the y-axis while in motion. The relating coordinate system can be seen in Figure 11 .

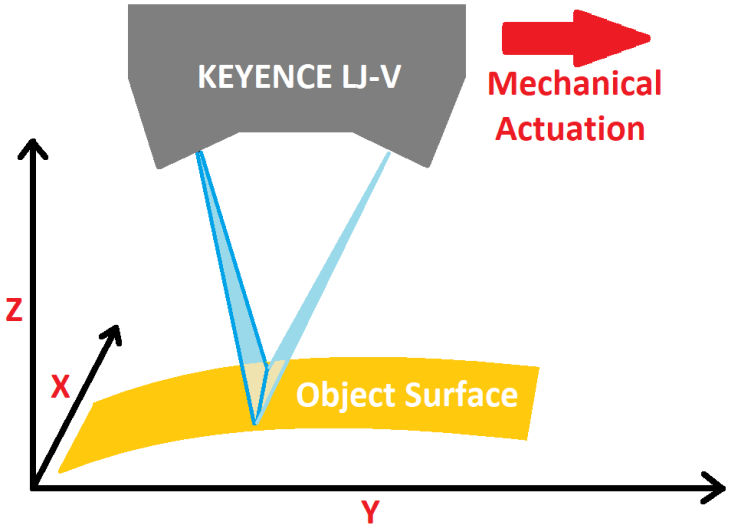


Figure 11 - Detail of coronate system for apparatus

Digital Image Processing

In the interest of speed, multiple triangulation sensors are implemented simultaneously to rapidly collect data of the object. In order to get to the final result of reporting on the quality of the object, accurate digital reconstruction of the scanned object is key. To do this, the data must be formatted and stitched together with an accuracy of ±5µm. The first step after the data points are collected is to then sort the data points into the proper format. This process of formatting the data points must not compromise the accuracy of the raw data. Once each of the scans is formatted into the proper point cloud format, the process of stitching must be executed with extreme precision. Any error in the stitching of point clouds will carry over to the final measurement of the point cloud. This final measurement will take the resulting point cloud generated by the 3D scanner and the 3D printed objects base file geometry and compute the deviation between the two. The threshold deviation must be greater than 5µm as this is the prescribed limitation of the 3D scanner.

Timing

The device must complete a scan, transfer the raw data, and then process these data points into a usable point cloud in less than 60 seconds. The process of acquiring data must be accomplished quickly to allow as much time as possible for transfer and processing considering some files may approach gigabytes in size.

DESIGN

Based off all the available non-contact methods of scanning available, the triangulation sensor technology was used due to its comparable accuracy and repeatability to other technologies. As stated previously in the literature review section, one of the major drawbacks to this sensing technology is its inability to scan in two dimensions simultaneously. This means that the sensor must be actuated across an objects surface to generate a three-dimensional scan. However, given the extra complexity needed to generate a three-dimensional mapping of an object, the overall speed of data acquisition is superior to other sensing technology. This is due to the high operational frequencies of some triangulation sensors such as the KEYENCE LJ-V series.

To fully exploit this high operational frequency characteristic of these sensors, a highly precise mechanical apparatus must be designed and manufactured for the purposes of actuating the sensor. Once an apparatus is designed and built, the sensor can be then programmed to take samples of known target geometry. The resulting scans will then be used to further calibrate the scanning apparatus. Once the overall accuracy of the apparatus meets the prescribed requirements, digital image processing steps are then performed to analyze and stitch the collected data together.

LJ-V7080 Sensor

For the purposes of prototyping the 3D scanner, the LJ-V7080 3D laser profile scanner was used instead of the LJ-V7020 due availability and cost. As previously stated, these units and accompanying electronics cost around $30,000.

The LJ-V7080 has a larger x-axis accuracy of 50µm compared to the LJ-7020's x-axis accuracy of 5µm. The z-axis repeatability of the LJ-V7080 is close to the LJ-V7020 with only a difference of 0.1µm. To account for the fact that the sensor being used is recording a lower resolution image, the final calculation for processing time will be adjusted to reflect an accurate scanning time for the use of the full resolution sensor.

The LJ-V sensor, seen in Figure 12, is a high-end triangulation sensor. The sensor utilizes a blue frequency light with a wavelength of approximately 380-450nm. Its default operational frequency is 1000Hz, however, KEYANCE provide overclocking tools and warrants supported operational speeds up to 4000Hz Simply put, the LJ-V is a line scan camera that takes physical topography.

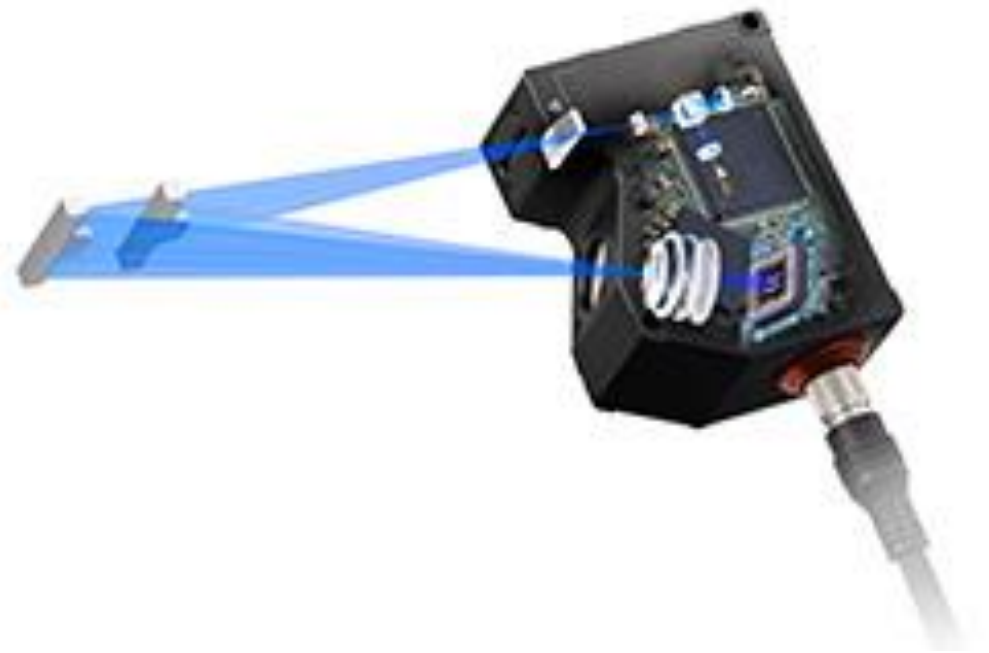


Figure 12 - KEYANCE LJ-V Laser Profile Scanner

Mechanical Apparatus

The mechanical apparatus is crucial for providing a solid foundation for the sensors to execute a highly accurate scan. The mechanical apparatus will house three KEYENCE LJ-V sensors in a delta configuration as seen in Figure 13. The sensors will each be mounted to a sensor actuation tower, which is an assembly of mechanical components consisting of the following: a main riser, a linear rail and carriage, idler pulleys, a belt, pinion, and motor.

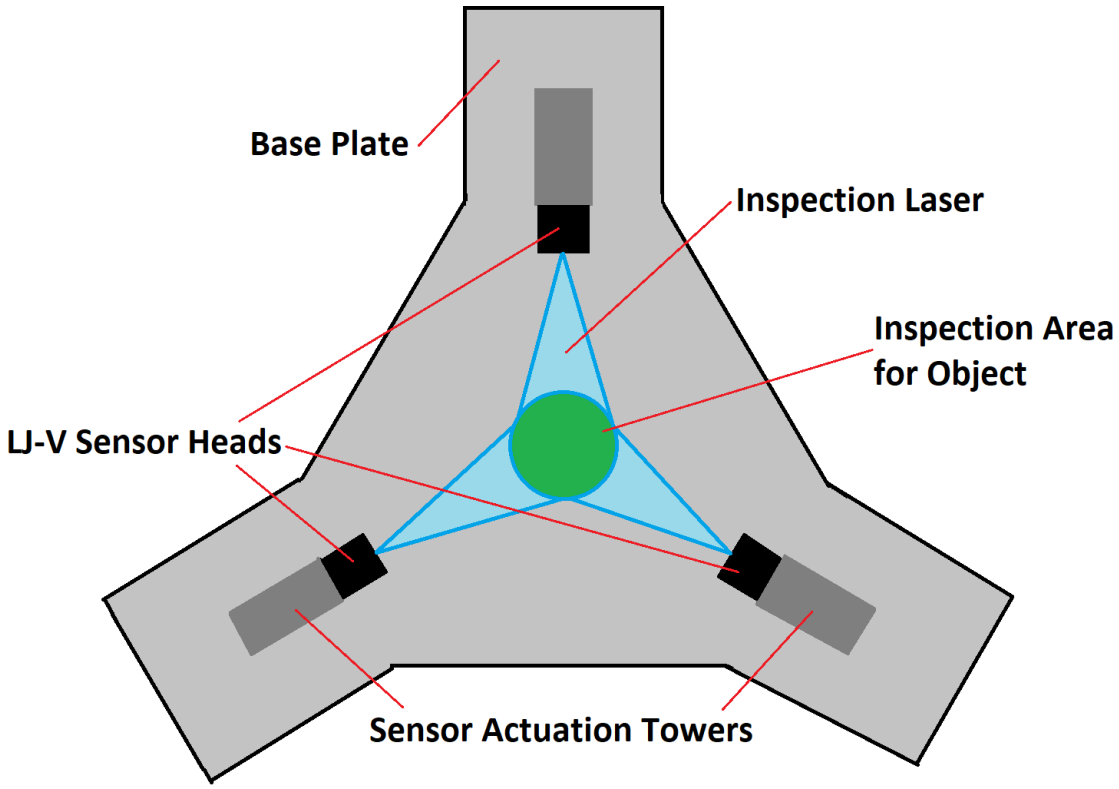


Figure 13 - Top down view of delta configuration apparatus

Since the attempted accuracy is at the micron level, any sort of external vibration needs to be isolated before it can propagate its way to the sensor. Therefore, the mechanical apparatus will be supported by vibration dampening legs to absorb any internal or external disturbances.

Complete Apparatus Assembly

Since only one KEYENCE LJ-V sensor is available for use in development of the 3D scanner, only one of the sensor actuation towers will be developed and manufactured. As previously stated, the sensor actuation tower is an assembly of stationary and moving components. The overall assembly can be seen in Figure 14. Within this main assembly there are four sub-assemblies. These sub-assemblies include: The Sensor Carriage Assembly, The Top Idler Pulley Assembly, The Riser Assembly, The Bottom Idler Pulley Assembly, The Base Assembly, and The Mechatronics Assembly.

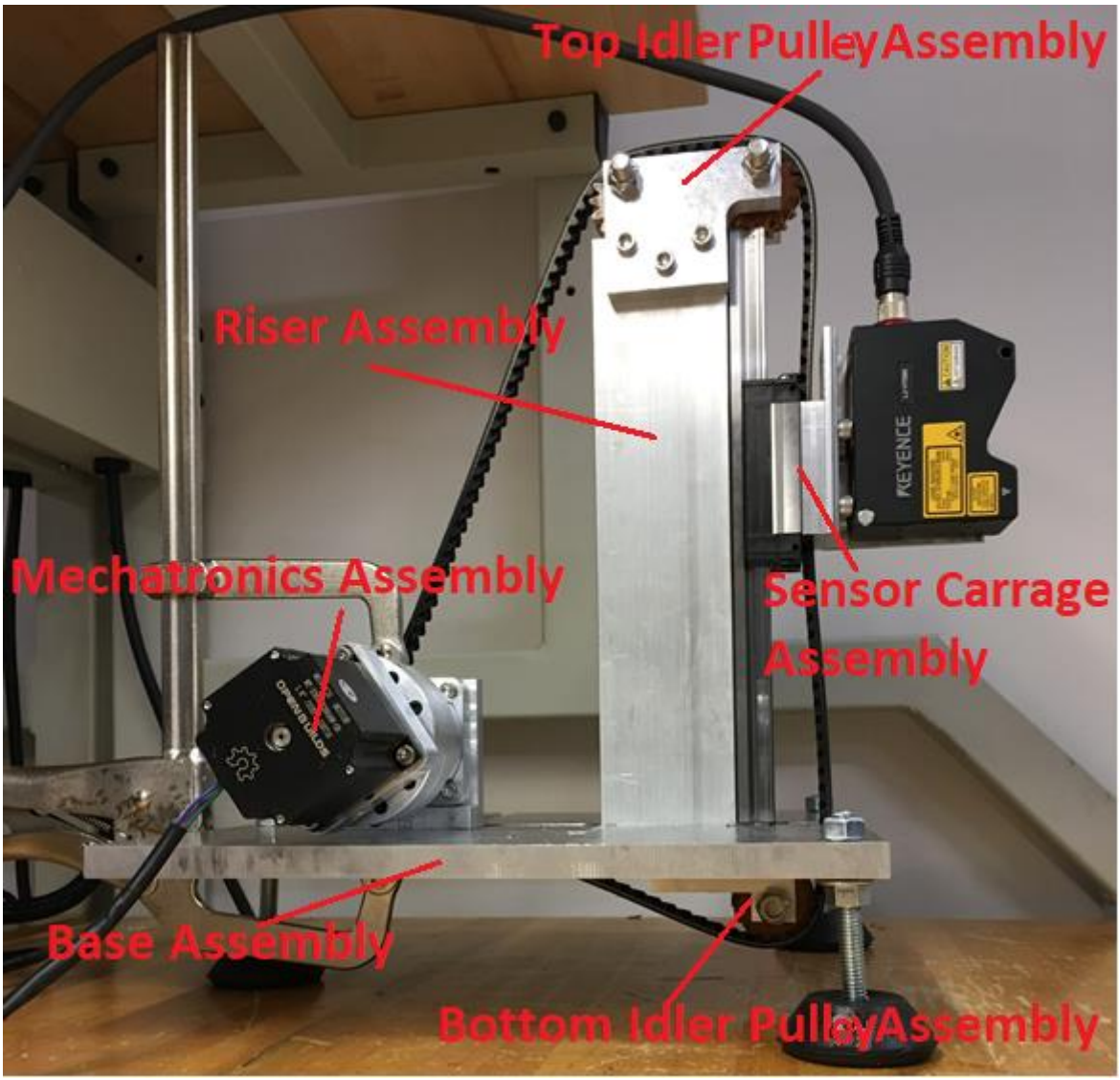


Figure 14 - Complete 3D scanning apparatus assembly

Sensor Carriage Assembly

The first sub-assembly is the Sensor Carriage Assembly seen in Figure 15. This sub-assembly is designed to accomplish three tasks. The first is to attach the sensor to the assembly. The second is to rigidly attach the sensor carriage assembly to the main drive belt, and third is to interface with the vertical linear rail.

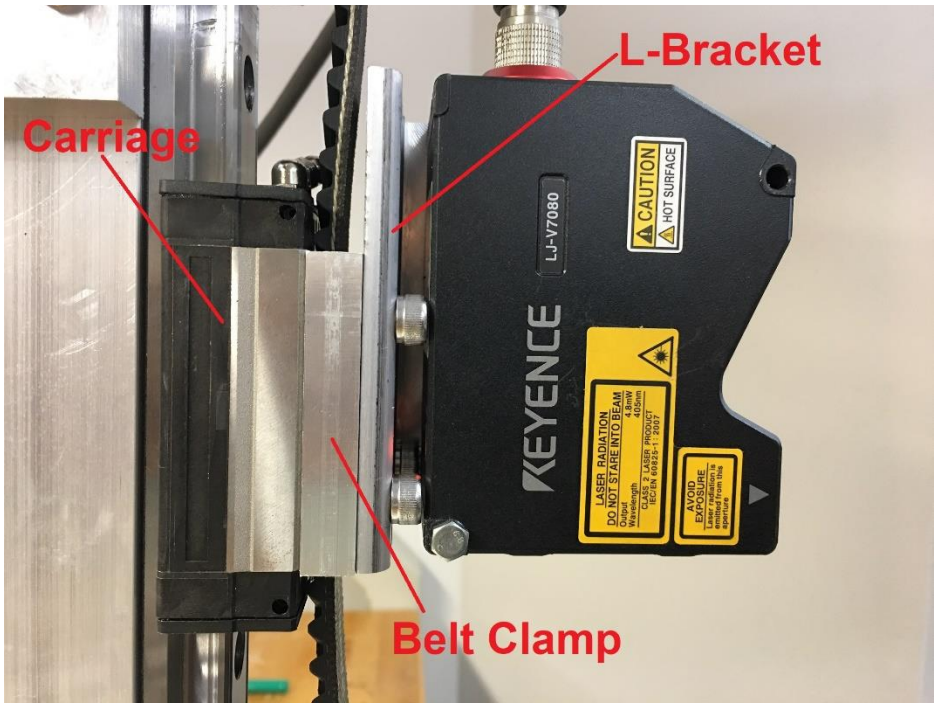


Figure 15 - Sensor Carriage Assembly attached to linear rail

To accomplish these three tasks, three components were designed. The first, seen in Figure 15, is the L-Bracket. This component is responsible for providing a mounting point for the sensor head. The second component in the assembly is the rail carriage. The rail carriage is a standard off-the-shelf part designed to accurately attach to the linear rail. The third is the Belt Clamp seen in Figure 15. This component is a rectangular plate with a near press fit groove through the underside of the part. The depth of this groove is 0.5mm less than the thickness of the main drive belt, so that when assembled the drive belt is rigidly fixed to the sensor head assembly, L-Bracket and the Carriage sandwich the Belt Clamp. This results in the Belt Clamp and the Rail Carriage securing the main drive belt in place.

The main drive belt is a high-strength, ultra-quiet timing belt used for a variety of industrial applications. It is important that the belt is rigid and non-elastic to retain high accuracy motion. The tooth design of the belt is known as Wuest type herringbone teeth. This means that “the teeth on opposite sides of the center line are staggered by an amount equal to one half the circular pitch”. (Daniels, 1921) This is best shown in Figure 16 where the typical herringbone gear is on the left and the Wuest herring bone gear on the right. The main benefit of the Wuest herringbone tooth is its quieter operational noise compared to typical sprocket gears or conventional herringbone gears.

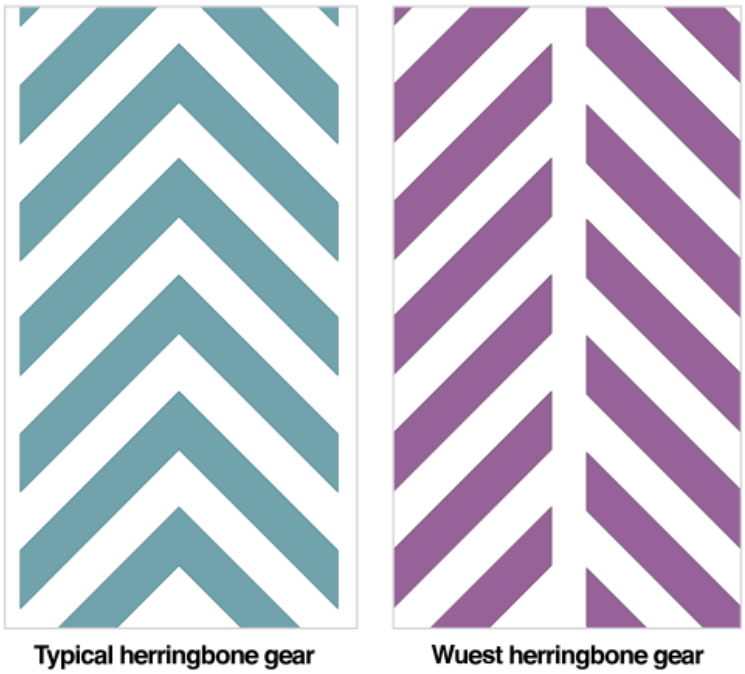


Figure 16 - Wuest herringbone vs. Typical herringbone

Top Idler Pulley Assembly

The second sub-assembly is the Top Idler Pulley Assembly. This configuration of components serves two purposes; the first is to guide the drive belt over and around the main riser. Its second purpose is to precisely align the belt parallel to the linear rail. This is extremely important because any misalignment in the belt with the Sensor Carriage Assembly and the linear rail will cause any belt displacement and carriage displacement to relate nonlinearly. This is mathematically explored in the Validation section of this thesis. The physical components of the Top Idler Pulley Assembly consist of two mounting plates, two idler pulleys, four high precision ball bearings, four retaining rings, and two shafts seen in Figure 17.

Figure 17 - Top Idler Pulley Assembly

The two idler pulleys needed to be custom designed since the Wuest type herringbone teeth were used in the main drive belt. Designing the custom idler pulleys was accomplished by modifying the outputs of Autodesk Inventor's herringbone gear package. The resulting design can be seen in Figure 18. The custom idler pulleys were then 3D printed out of Cyanate Ester via a Carbon M1 3D printer seen in Figure 19.

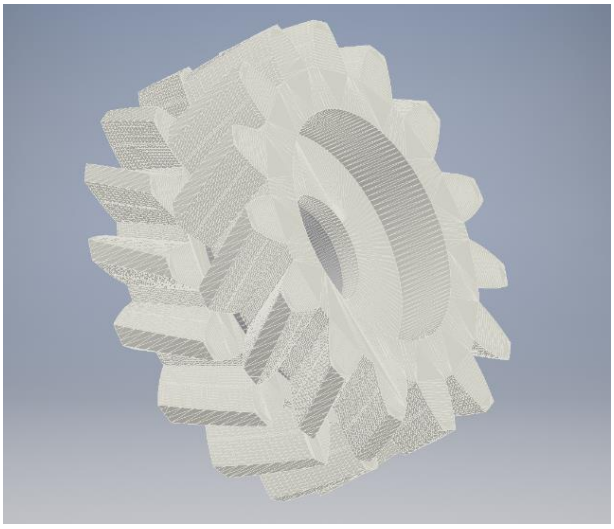


Figure 18 - Wuest type herring bone idler pulley designed in CAD

Figure 19 - 3D Printed Custom Idler Pulley via Carbon M1

Once completed, two ball bearings were press fit into each of the idler pulleys. Once assembled, the idler pulleys have a 2mm gap on either side due to the width of the main riser. To isolate the idler pulleys from moving side to side, four retaining rings were designed and printed using a standard fused deposition modeling (FDM) printer. The installed rings can be seen in between the Idler pulleys and the mounting plates in the right side of image of Figure 17.

Riser Assembly

The third sub-assembly is the Riser Assembly. The body of the Riser Assembly consists of a riser mast and a linear rail. The riser mast is used as a fixture point for the Top Idler Pulley Assembly, and rigidly bolts to the Base Assembly of the device. The linear rail is mounted to the riser mast and interfaces directly with the Sensor Carriage Assembly seen in Figure 20. Precision manufacturing of this assembly is critical to insure proper alignment of all moving components.

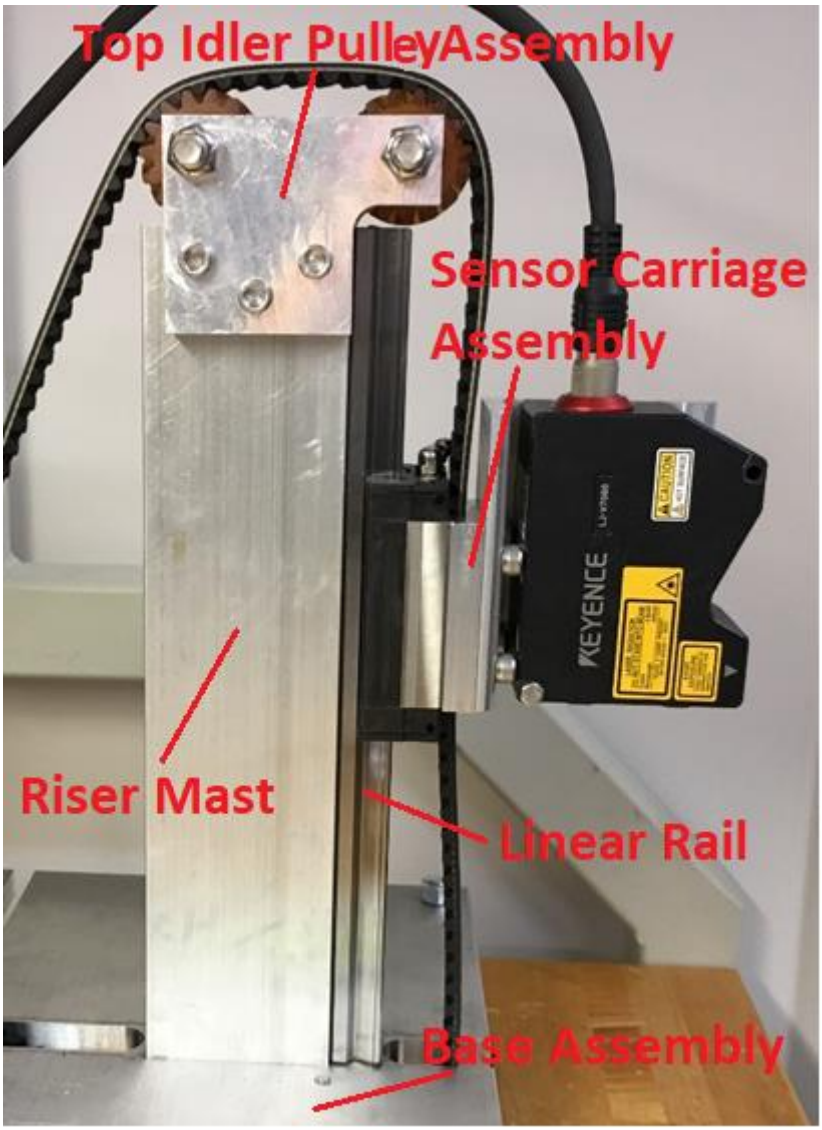


Figure 20 - Full Riser Assembly

Bottom Idler Pulley Assembly

The forth sub-assembly is the Bottom Idler Pulley Assembly. This assembly's operating principal is identical to the Top Idler Pulley Assembly. However, the bottom Idler Pulley Assembly consists of only one custom idler pulley and relies on two mounting plates attached to the underside of the base plate seen in Figure 21.

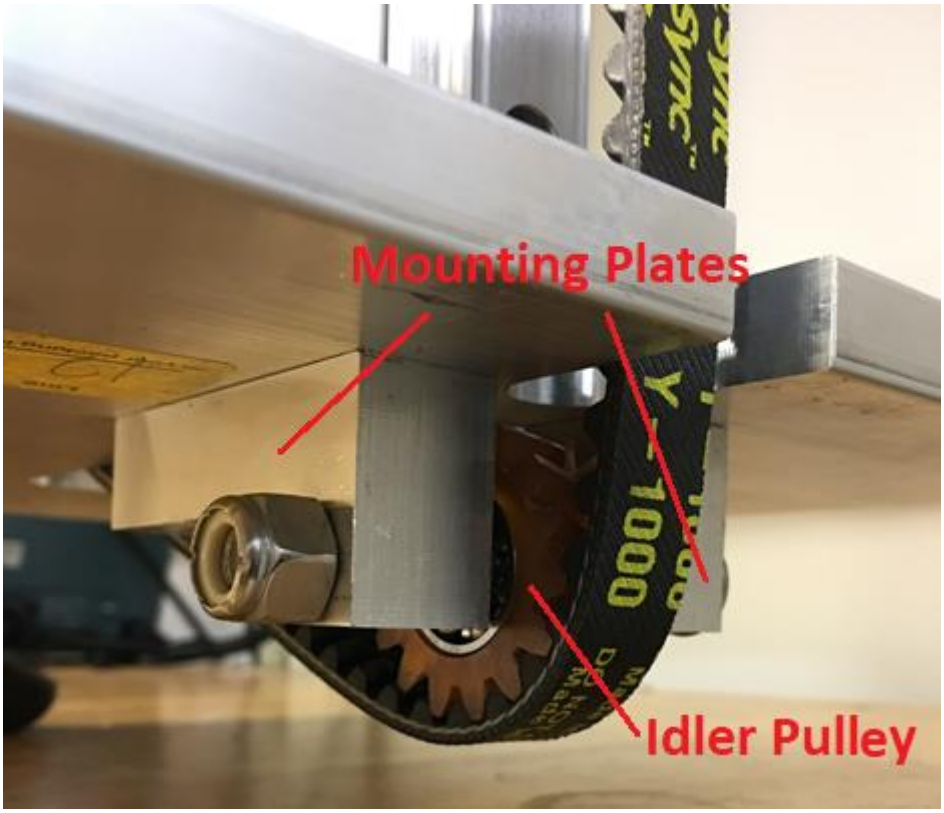


Figure 21 - Bottom Idler Pulley Assembly

Base Assembly

The Base Assembly for the apparatus relies on the base plate for a multitude of tasks. Most importantly, the baseplate contains three access slits for the drive belt seen in Figure 22.

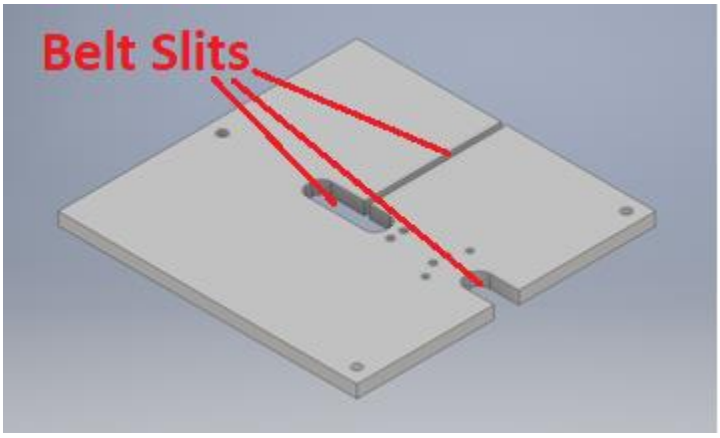


Figure 22 - CAD rendering of Base Plate

The access channel in the front and center of the base plate allow for drive belt clearance while cycling through the apparatus. It is necessary for the drive belt to be diverted underneath the base plate due to the presence of the main riser tower. The third channel (seen on the right side of the base plate in Figure 22) allows for the drive belt to be installed. The implemented design for the belt transition around the main riser can be seen in Figure 23.

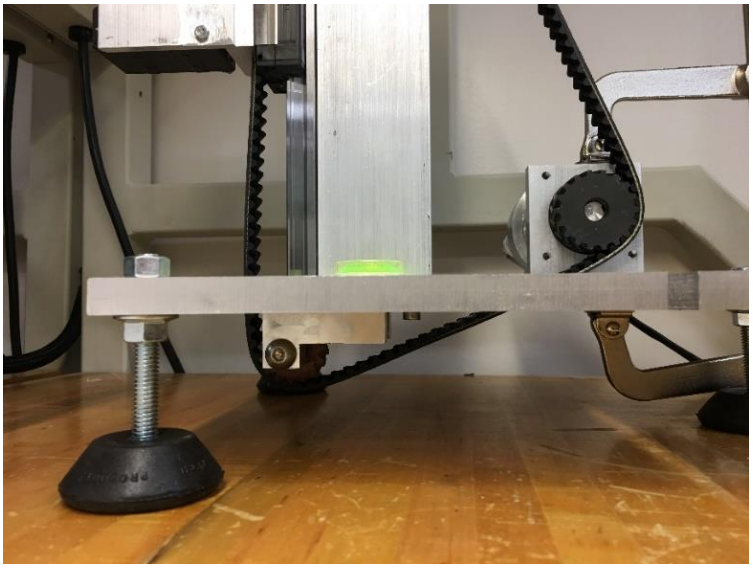


Figure 23 - Side view of belt translon through the base plate

Mechatronics Assembly

The fifth sub-assembly is the Motor Drive Assembly. This assembly contains a NEMA 23 stepper motor, 1:100 ration harmonic gear box, a Wuest type drive gear, and mounting bracket. This assembly can be seen in Figure 24.

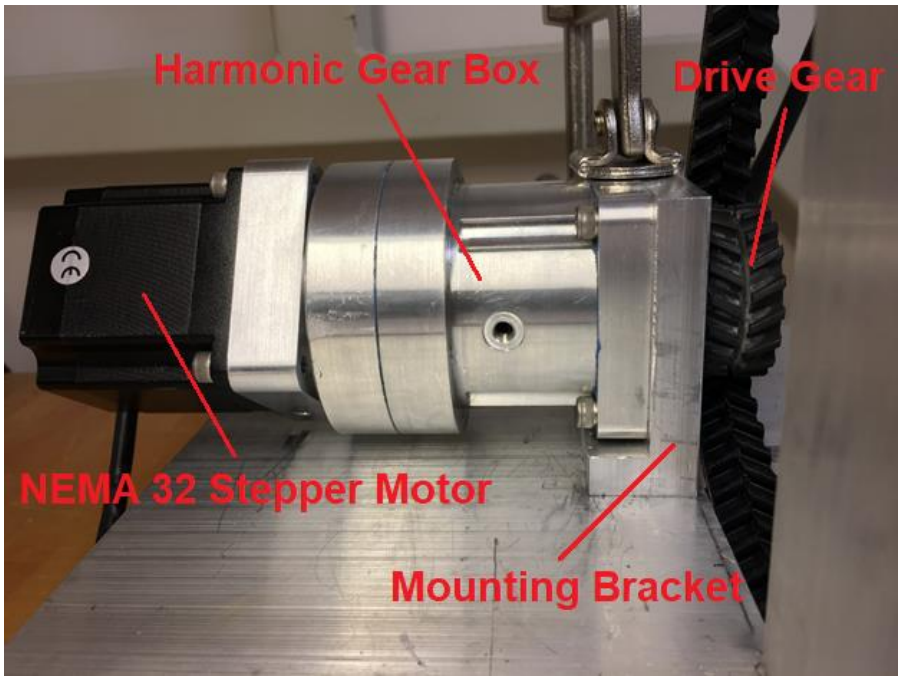


Figure 24 - Motor Drive Assembly held in position via temporary C-Clamp

The key to precise sensor actuation relies on precision motion control. To accomplish this task there are two contenders: the stepper motor and the servo motor. “While servo motors are best suited for high speed, high torque applications that involve dynamic load changes. Stepper control systems are less expensive and are optimal for applications that require low-to-medium acceleration, high holding torque, and the flexibility of open or closed loop operation.” (AMCI, 2018). Additionally, stepper motors do not require feedback to determine shaft position because, stepper motors operate incrementally. Due to the required acquisition velocity of the sensor and high tension of the drive belt, a stepper motor was chosen for its high torque at low-medium speeds and low cost for use in the apparatus. Specifically, a NEMA 23 stepper motor is used due to form factor, and a significantly higher holding torque of 1.25Nm compared to the smaller NEMA 17 holding torque of 0.14Nm.

Harmonic Gearing

As previously stated, a 1:100 harmonic gear box is utilized to increase the overall resolution of the NEMA 23 stepper motor. Harmonic gears are known for very high gearing ratios in a small package due to inherent working principal behind the gears. Harmonic gears consist of three main components: a wave generator, a flexible cup spline, and a circular spline. These components can be seen in Figure 25.

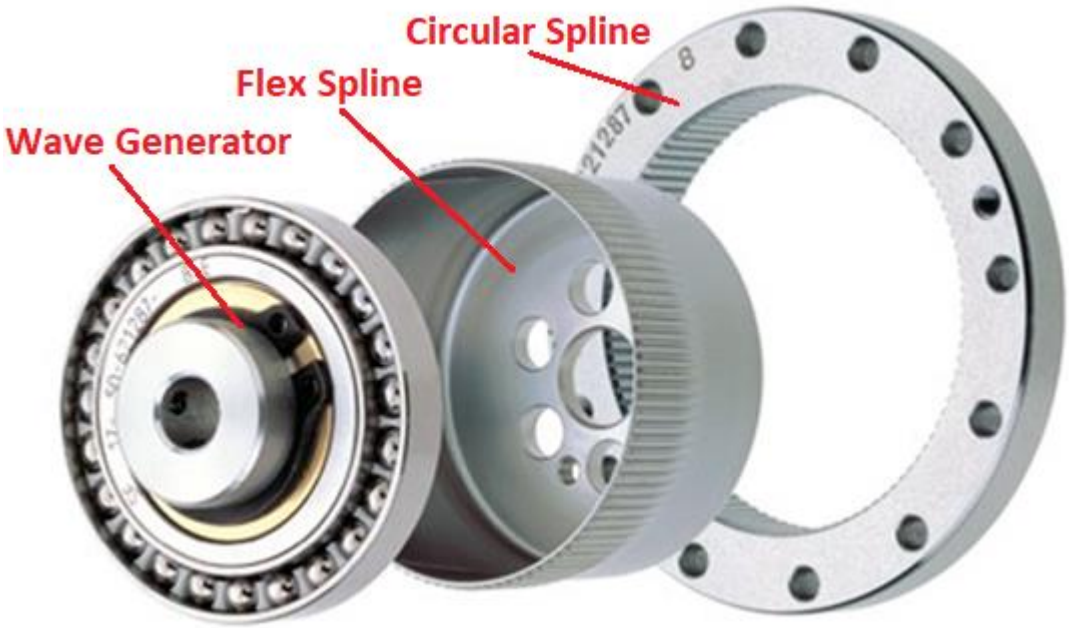


Figure 25 - Harmonic Gear core components, wave generator, flex spline, circular spline

The wave generator is a rigid ellipse with ball bearings on its exterior surface held in place by a flexible retaining ring. This retaining ring interfaces with the flexible cup spline while the major axis of the ellipse deforms the flexible cup spline so that the exterior teeth mesh with the interior teeth of the circular spline. The minor axis of the wave generator allows the flexible cup spline clearance so that it does not interface with the circular spline teeth. This can be seen in Figure 26, where the wave generator is green, the flexible cup spline is red, and the circular spline is blue.

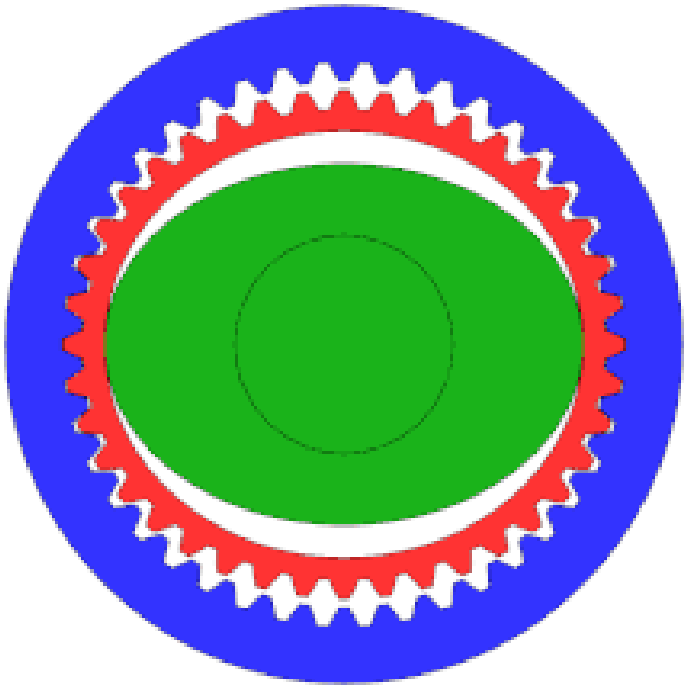


Figure 26 - Cross-sectional interface view of core harmonic drive components

Harmonic gears are used in various applications however, due to the associated cost they are primarily used for cases involving high precision motion as the drives have zero-backlash. This is due to roughly 10-40 teeth being in contact at once depending on the radio of the drive. For the purposes of 3D scanning, this means that additional functionality can be implemented, such as re-scanning specific high noise areas. This retention of high positional accuracy is only possible with the harmonic drive paired with a high precision motor such as a stepper.

Electronic Control Systems and Data Allocation

The electronics for the 3D scanner can be divided in to two sub systems. These sub systems include the electronics that control mechanical actuation and the electronics that acquire and record 3D data. The two sub systems are intended to communicate in conjunction with one another. However, for the purposes of this thesis, the operation of each system is independent and requires the user to operate both systems in parallel.

Mechanical Actuation Electronics

The mechanical actuation electronic system consists of a PLC, a stepper driver, two separate power supplies, and the stepper motor previously mentioned. Figure 27 details the basic configuration of the system.

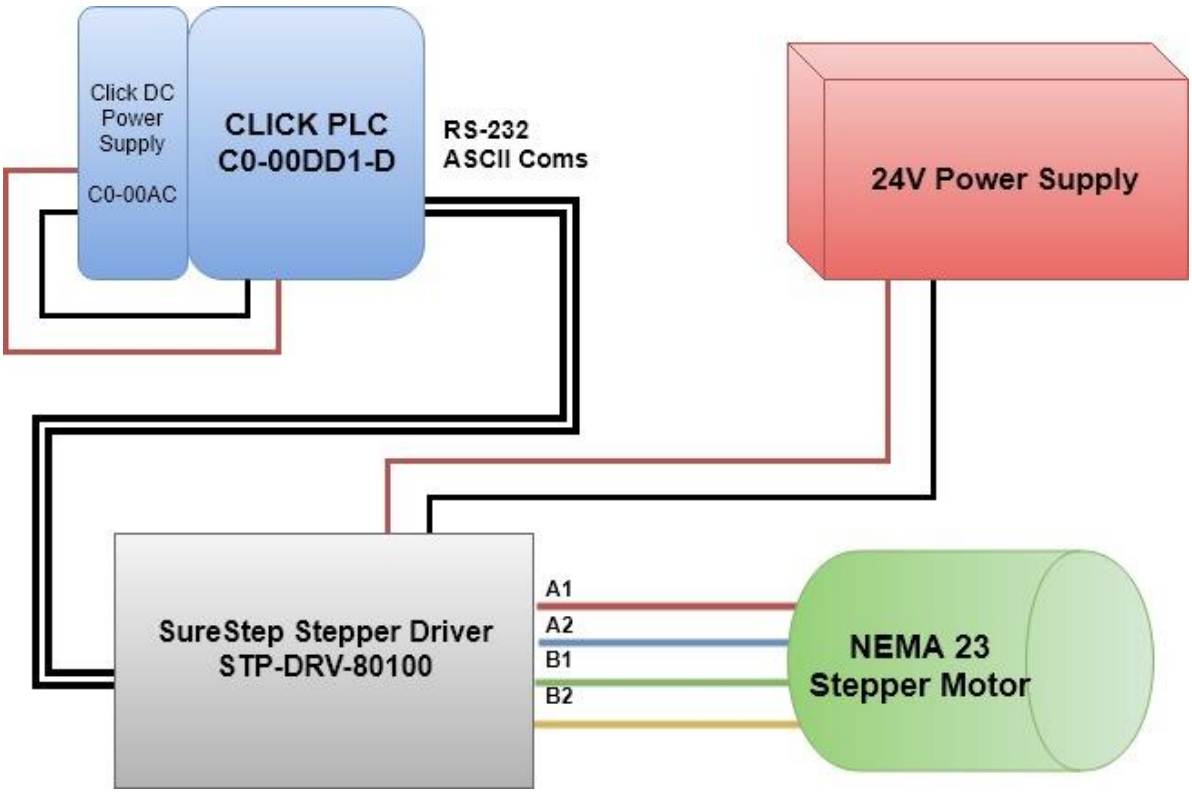


Figure 27 - Basic electronic layout for motion control system

The CLICK PLC & Programming

The CLICK PLC is programed using ladder logic via a PC based programing application called "CLICK Programing Software Ver2.10". The ladder logic program is designed to send four separate ASCII string commands using the Modbus ASCII communication protocol over a RS-232 communication line. The Stepper Driver receivers these four ASCII commands that specify the accretion of the stepper motor ("AC#"), the deceleration of the stepper motor("DC#"), the terminal velocity of the stepper motor("VE#"), and the total number of steps to perform the motion in ("FL#"). The ladder logic for the CLICK PLC can be found in Appendix B.

For the purposes of prototyping, the initiation of mechanical actuation electronics is manually controlled by the user. For example, Up and Down actuation for the sensor head is input to the PLC via a single throw double pole switch as seen in the right side of Figure 28. The resulting actuation of the sensor can either be manually controlled with a stop override button or will automatically terminate after a set distance.

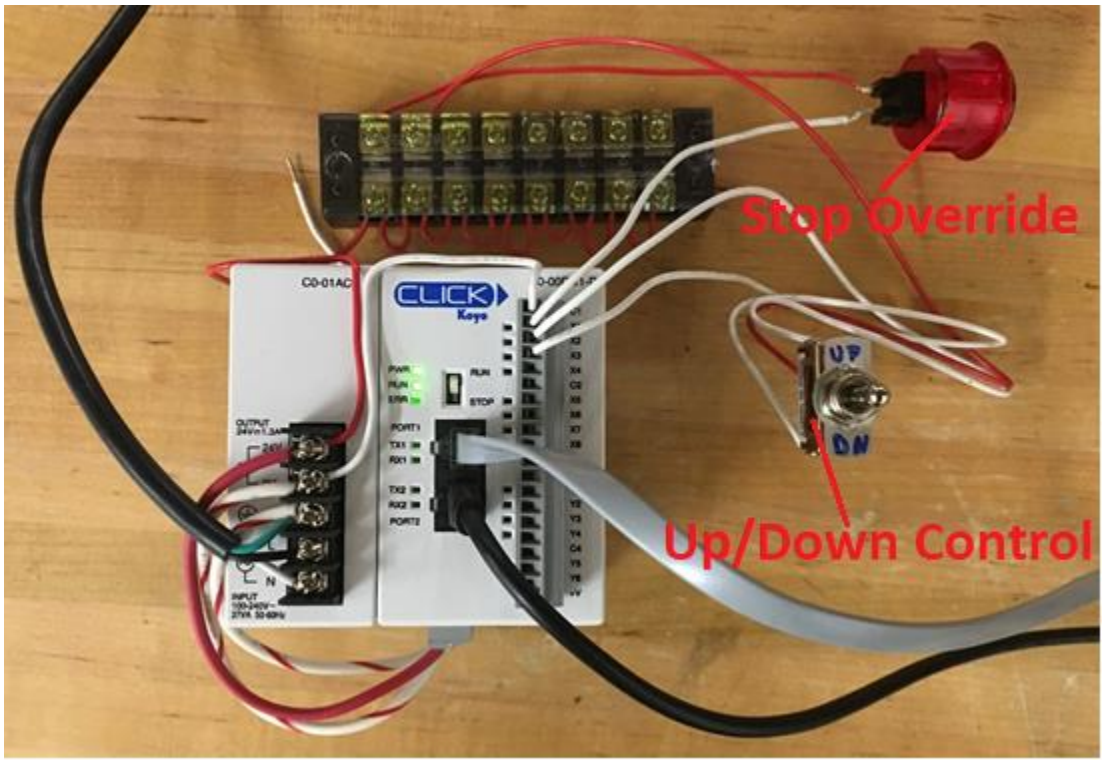


Figure 28 - PLC, PLC power supply, and user control electronics

Stepper Motor Driver & Power Supply

The SureStep stepper driver needs to be configured using an application called “SureStepRRO”. This application is used to configure the specific method of incoming communication from the PLC and how to translate the incoming information into output signals for the stepper motor. The application additionally configures the type of stepper motor being used to set current limits and any microstepping or smoothing parameters.

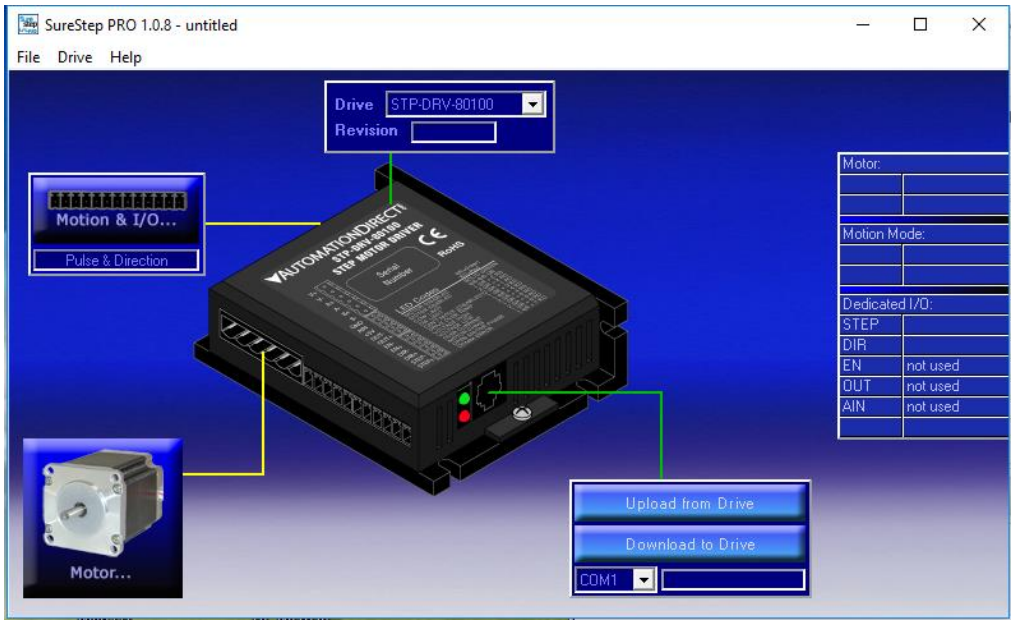


Figure 29 - SureStep PRO programing application

As seen in Figure 27, there are two separate power supplies for the system this is because the current draw for the stepper motor would overwhelm the 24V 1.3A power supply for the PLC. The Click PLC power supply is exclusively for supplying power to the PLC’s processing unit, and to supply power for the sinking /sourcing input channels. The sinking/sourcing input channels are used for the emergency stop button and the up and down toggle switch.

3D Scanning and Processing Electronics

The main 3D scanning processing electronics consists the LJ-V7080 sensor head, the KEYECNE LJ-V7001 sensor controller, the MS2-H50 power supply, and a host computer. Figure 30 shows the fundamental configuration of this system.

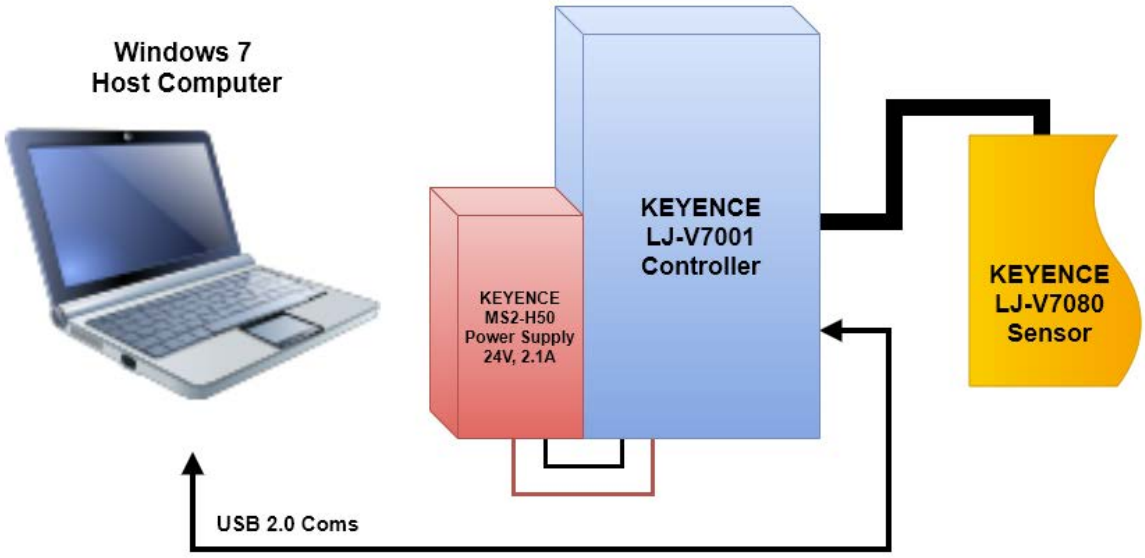


Figure 30 - Basic electronic configuration for scanning and processing electronics

The KEYENCE LJ-V7080 is the triangulation laser profile sensor used for acquiring data directly from an objects surface. Please see the Sensor section in the Design portion of this paper for specifics on the LJ-V7080. The LJ-V7001 control unit is responsible for controlling and driving the electronics within the LJ-V7080 sensor head. The LJ-7001 controls the operational frequency of data acquisition by controlling the laser intensity and the receiving CMOS chip's exposure time. Additionally, the controller stores the acquired data from the sensor head in the controller's system memory because streaming this information directly to the host computer would oversaturate USB 2.0 bandwidth. The host computer is the main terminal for the user to operate the sensor electronics via the "LJ-Navigator 2" software. This program allows the host computer to interface with the LJ-7001 control unit to alter various sensor and system acquisition parameters.

Processing of Data

Generating a Point Cloud

The raw data that the KEYENCE LJ-V7001 generates from the sensory inputs from the sensor head are similar in format to a standard image such that pixel X and Y location correspond to the rows and columns of the image array. The raw output data format can be seen on the left side of Figure 31. While this data format technically describes a point cloud, MATLAB and other point cloud software such as Sequoia will not accept input data in this format. This may be due to the fact that the raw image format will occlude different Z data points who share the same X and Y coordinates. Out of necessity, an algorithm was developed to sort the data into an accepted CSV format seen in the right side of Figure 31. In this sorted point cloud data format each point occupies its own row while the columns of each row describe the points exact X, Y, and Z location.

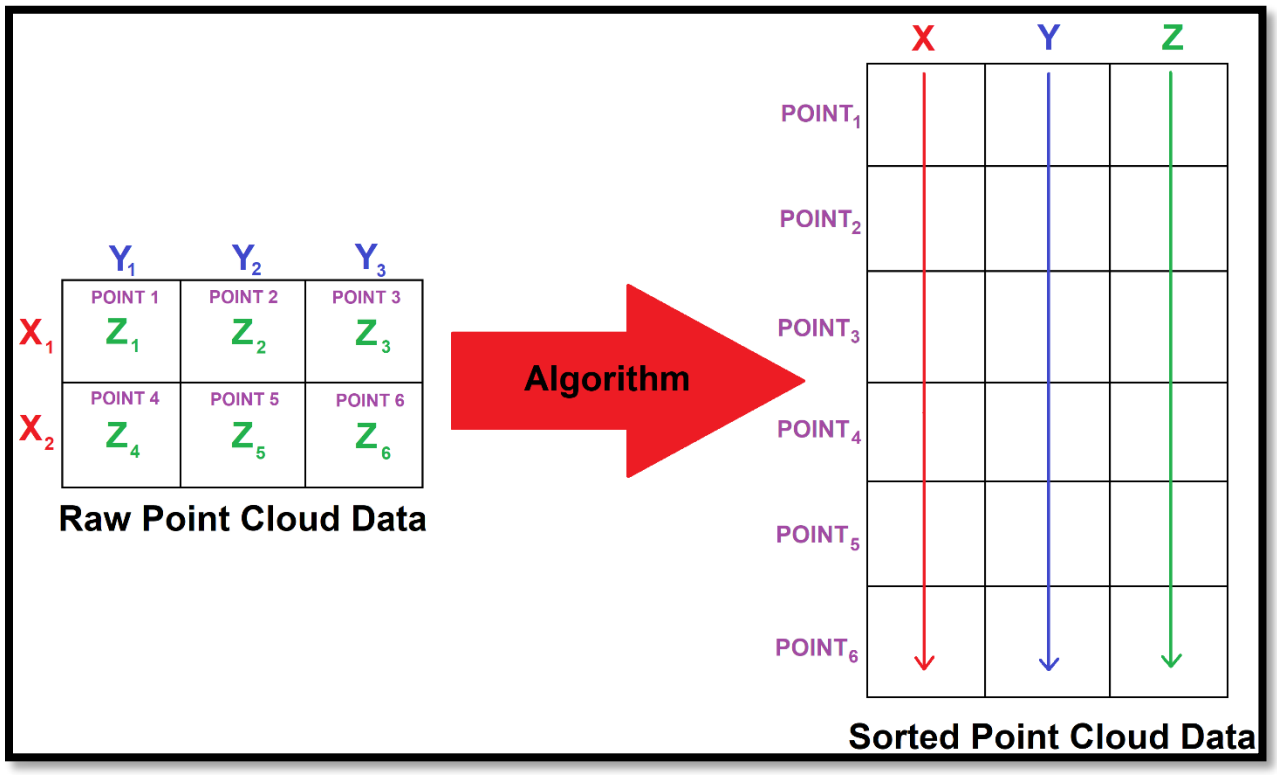


Figure 31 - Visual representation of converting raw data into acceptable point cloud data format

It is important to understand that MATLAB is extremely powerful when it comes to the computations of linear algebra. At its core, MATLAB utilizes Basic Linear Algebra Subprograms (BLAS), which are a set of low-level function/subroutines for preforming common linear algebra operations (MathWorks, 2018). Case in point, the reason MATLAB is so powerful for matrix-based math is due to the highly optimized Fortran compilers that run the BLAS libraries within MATLAB (Borini, 2009).

However, MATLAB lacks in efficiency when it comes to non-linear algebra-based processing. For example, when MATLAB is told to read or write a file, or to open a function, the environment utilizes a frame work based on C++ and Java. These operations will be sluggish because MATLAB uses an interpreter rather than a compiler to generate machine executable code. The significance of this is that compiled code can be optimized and generally runs faster due to the code being compiled in advance rather than on the fly. (Fellicious, 2016) For these reasons a secondary sorting algorithm programed with C# in the .NET framework will be used to help optimize any inefficiencies related to MATLAB. A detailed analysis of these two algorithms can be found in the Testing and Validation section of this paper.

The MATLAB Sorting Algorithm

The logical flowchart for the matrix transformation MATLAB algorithm can be seen in Figure 32 while the physical code can be seen in Figure 33. The algorithm first starts by reading and cropping the raw CSV data collected and exported by the KEYENCE LJ-V7001. The read data points are then assigned to matrix M. The algorithm then replaces all values below a specified threshold value of -90 with not a number (NaN). This is because the sensor reports the Z value of empty space as (-99.9999) in the CSV file. Once this operation is complete MATLAB finds the minimum value in the matrix, produces its absolute value, and adds this value to every number in the matrix. The point of this operation is to zero the Z axis for the entire point cloud. The next step in the algorithm is to set the X and Y cell spacing to a physical distance. The algorithm then generates two vectors whose values correspond to the cells X and Y positions where a number exists. These vectors are then used in the last line of code where the resulting output matrix (N) combines the X and Y

vectors with a corresponding Z vector. The values within this Z vector correspond to the height value of the cells with corresponding X and Y locations as described by Figure 31. All MATLAB code can be found in Appendix E.

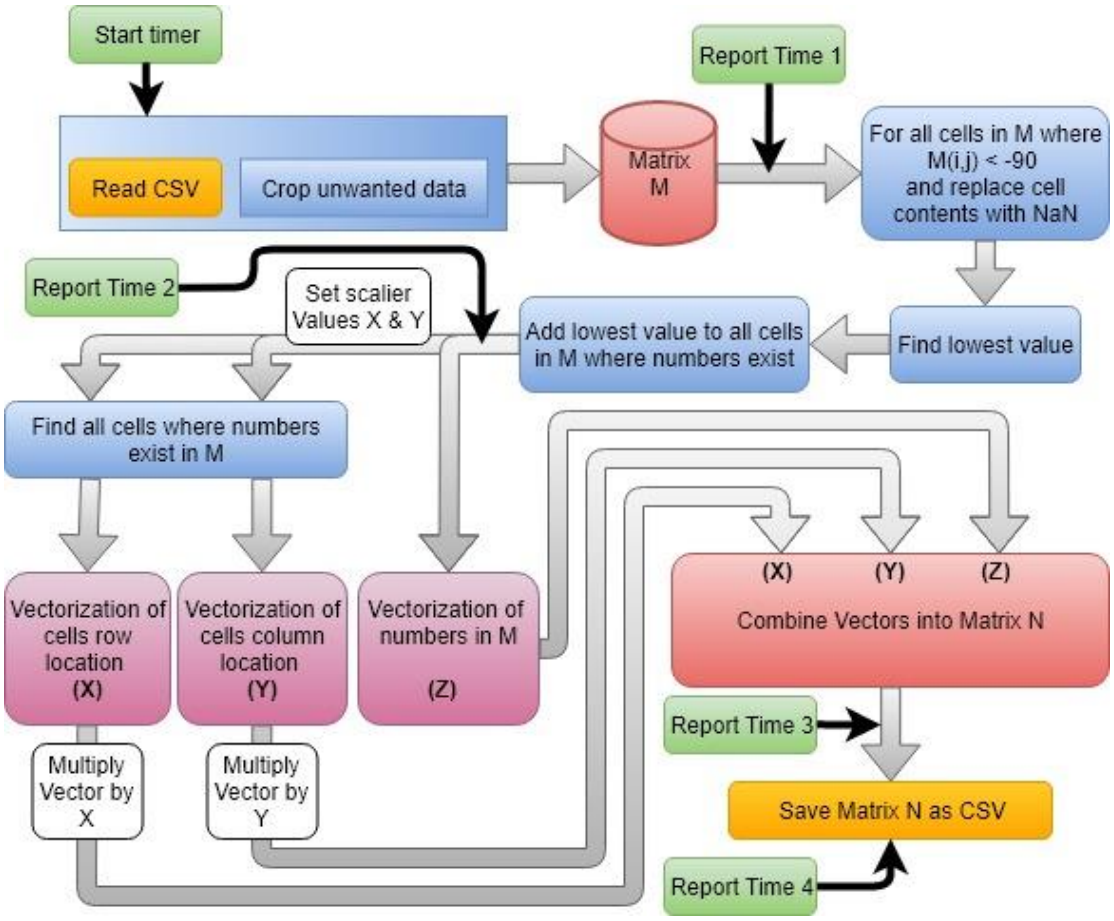


Figure 32 – Logic flow chart for the MATLAB sorting algorithm

```
M = csvread('LBack.csv',17,20); %Load CSV files matrix into memeory
M(M < -90) = NaN;               %Make all values grater than thereshold NaN
mn = min(min(M));               %Find minimum value in matrix
Mn = abs(mn);                   %Make this value positive
M = M + Mn;                     %Add this value to all numbers in matrix
x = 0.005; y = 0.005;           %Scalling values for X, Y, and Z
[rows,cols] = find(~isnan(M));  %Vectorization of Rows and Colums
N = [rows*x, cols*y, M(~isnan(M))];%Combines X, Y, and Z vector to output
csvwrite('LBackOut.csv');       %Saves matrix N to CSV file
```

Figure 33 - MATLAB code for sorting algorithm

The C# .NET Framework Algorithm

Due to the inherent simplicity of the programing language compared to MATLAB's very high-level scripting language, C# allows more programing control over data sets than MATLAB. For example, C# is a Strongly Typed language while MATLAB is Weakly Typed. The significant difference between the C# program and the MATLAB algorithm is the method in which data points are handled. MATLAB uses a buffer approach, meaning all data and variables must be loaded into RAM in order to begin algorithm computations. The C# program uses and unbuffered approach, meaning data points are streamed in from the source file directly without loading the file into RAM. The logical flow of processing each line is best shown in Figure 34.

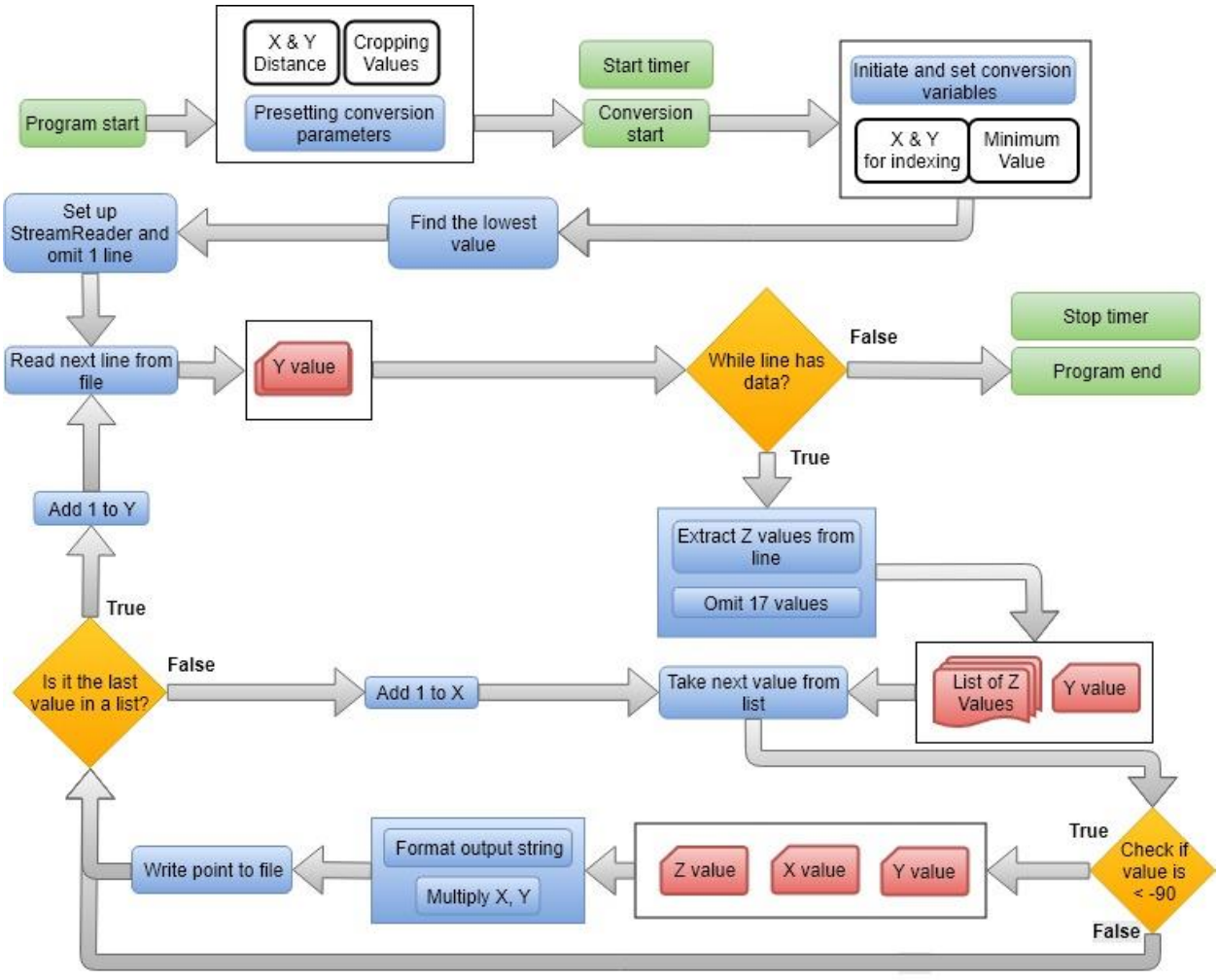


Figure 34 - Logical flow chart of C# sorting algorithm

The C# program provides a handy graphical interface that can be used to preset conversion parameters before conversion. The default method of operating this program includes setting the X & Y distance and cropping variables and specifying the input file and output location. The algorithm begins computation immediately after the user specifies a save location. Once the program beings, a debug timer is initiated, and the program searches for the lowest value in the input file. This value is used to translate all 'Z' values such that the lowest value corresponds to 0. Once the lowest value is determined, the algorithm prepares an input and output stream such that these streams can read and write data from the hard drive. If the first line in the data contains header information from the sensor, the first line is omitted.

After the initial preparations for the program are set, the main loop beings reading the data file line by line. Since the program only reads one line at a time, all values within that line correspond to the same Y value. Therefore, each loop iteration will increment the Y value. The While loop reads the current line as a string of data. If there are data points within the string, the program then converts the string to a list of individual Z values. The order of this list indicates the X position of the Z value and is incremented accordingly. Note the program omits the first 17 values due to unnecessary data relayed by the sensor. From this point a For loop begins by converting each list to double type and checking if the current Z value is below a threshold value of -90. If the value is below the threshold, the Z value is not saved, and the lists X value is incremented by 1. If the Z value is above the threshold the corresponding X, Y, and Z values are formatted to an output string where the X and Y increments are then multiplied by the initial X and Y distance constants of 5µm. The resulting string is then saved to an output file. The For loop breaks when every Z string from the list of strings is exhausted. When this happens, Y is incremented and the While and For loops are fed a new line of data from the input file. This process repeats until the While loop is broken after all input data points are processed. The full code for this program can be found in Appendix C.
(Machura, 2018)

Stitching Point Clouds in MATLAB

Once a proper point cloud is introduced to the MATLAB environment, a multitude of built in MATLAB Point Cloud functions can be applied to the collected data. Of these functions is the built-in point cloud stitching process that utilizes Iterative Closest Point as mentioned in the Literature Review Section of this thesis. In order to stitch two point-clouds together, one point cloud must be designated as the stationary reference frame and the other must be designated as the point cloud to be manipulated. From this point an affine transformation matrix is generated using the MATLAB function "pcregrigid". This function is a cloud point registration algorithm based on the iterative closest point algorithm and requires the user to manipulate the function with Name-Value Pair Arguments to produce the best result. The arguments used are listed below.

1. **(Metric)** This tells the ICP algorithm to either iterate each point on either point cloud independently with the value 'pointToPoint', or to iterate each point of the rotating matrix with surface normal vectors from the stationary point cloud using 'pointToPlane'. The normal vectors are generated automatically using 6 local points.
2. **(Extrapolate)** This argument accepts values of either 'true' or 'false'. When it is true, the function adds an extrapolation step that traces out a path in the registration state space
3. **(InlierRatio)** This argument accepts scalar values to set a percent of inliers to use in sequential iterations. "A pair of matched points is considered an inlier if its Euclidean distance falls within the percentage set of matching distances. By default, all matching pairs are used." (MathWorks, 2018)
4. **(MaxIterations)** This argument accepts a positive integer that represents the number of iterations the ICP algorithm will execute. This argument is additionally useful for stress testing the computational speed of the computer being used.
5. **(InitialTransform)** This argument is used to preemptively rotate the moving point cloud into a coarse alignment with the fixed-point cloud. This is mainly done to prevent an error in the initial alignment of the point cloud so that the ICP algorithm does not waste iterations attempting to correct for an initial false alignment. The input value for this argument must be a three-dimensional affine geometric transformation.

The result of the ICP algorithm is a transformation matrix that relates the stationary point cloud to the moving point cloud. The transformation matrix is an affine transformation. This means that the applied transformation will generalize the properties of Euclidean spaces such that each of the point clouds configuration of discrete points will not be internally manipulated. Simply put, the transformation matrix will be applied to cloud of points without deformation. The affine transformation combines the translation and rotation transformation matrixes along with a scale and shear transformation matrix. The scale and shear transformation matrixes are used due to the non-uniformities associated with applying the ICP algorithm to real world data. Figure 35 shows in detail the specific component transformations within the Affine Transformation used.

Affine Transform	Example	Transformation Matrix	
Translation		$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	t_x specifies the displacement along the x axis t_y specifies the displacement along the y axis.
Scale		$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	s_x specifies the scale factor along the x axis s_y specifies the scale factor along the y axis.
Shear		$\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	sh_x specifies the shear factor along the x axis sh_y specifies the shear factor along the y axis.
Rotation		$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	q specifies the angle of rotation.

Figure 35 - Array components in Affine Transformation

Once an acceptable transformation is produced by the ICP algorithm, the transformation matrix is then applied to the point cloud that was referred to as “moving” with the MATLAB function ‘pctransform’. The last step is simply combine the two point clouds into a single point cloud with the ‘pcmerge’ function. After this step, the two point clouds are officially “Stitched” together and can be saved as either a .CSV or .STL file.

TESTING AND VALIDATION

Actuation Velocity

The general applied working principle for the operation of the sensor can be grasped by understanding the capabilities of the sensor and the accuracy requirements set. Because the operational frequency is in Hz, or samples per second, and the required accuracy is a distance, multiplication of these two variables yields a velocity. Equation 13 thus reflects the velocity at which the sensor must be actuated to obtain sample spacing of 5µm.

$$1000Hz \times 5\mu m = 5000\,\frac{\mu m}{s} \rightarrow 5\,\frac{mm}{s} \qquad eq.13$$

Precision of Mechatronic Actuation

The motion characteristics of the NEMA 23 stepper used, are such that each step translates to a shaft rotation of 1.8º. This means that one complete, rotation will be achieved after 200 steps. Since displacement is input into the system rotationally via a stepper motor and transmitted to the sensor head via a drive belt translationally, it is pertinent to understand the precise relation between input displacement and output displacement. This relation is expressed by the arc length formula of a circle seen in Equation 14.

$$x = arc\ lenght = \left(\frac{\theta}{360}\right)(2\pi r) \qquad eq.14$$

Due to the radius of the sprocket being 44.45mm and the stepper motors stepping angle of 1.8º±5%, each step would result in an unacceptable sensor displacement of 1.39mm per step with an accuracy of ±70µm per step. To remedy this, a (1:100) harmonic gear was used. This reduces the sensor displacement per step to 15µm with an accuracy of ±0.7µm. Even with the gear reduction, the linear motion is still outside of the required 5µm however, micro stepping can solve this. While whole stepping instantaneously changes coil current from high to low, half stepping and micro stepping incrementally varies coil current to achieve smaller step intervals. Figure 36 details the process of micro stepping and the relation between the variation in applied coil current and magnet position.

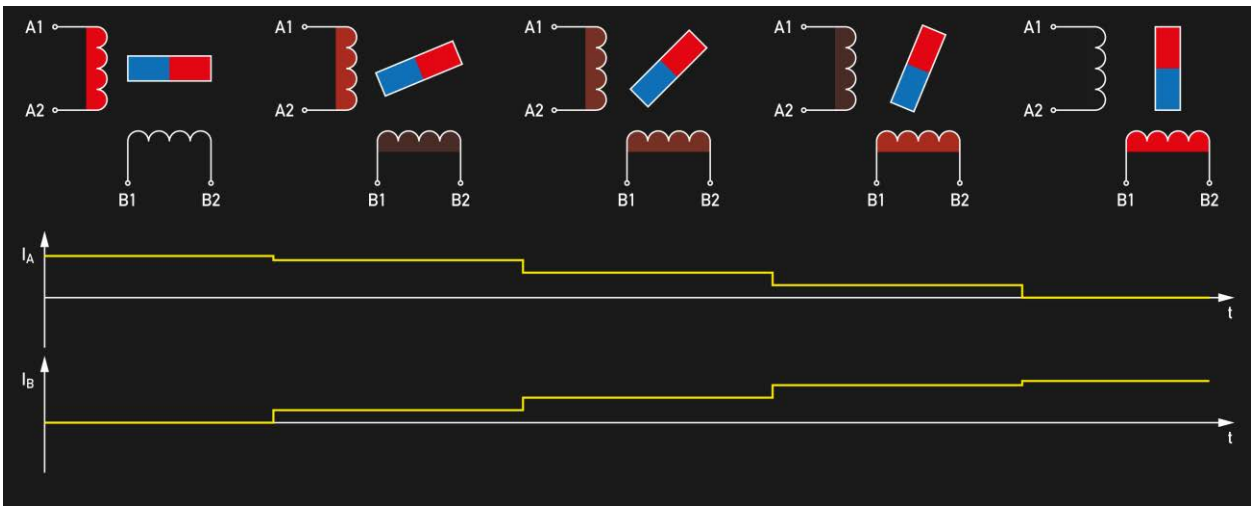


Figure 36 - Visualization of micro stepping

With the implementation of micro stepping at 1/8th intervals, the step angle reduces from 1.8º to 0.225º while the error increases from ±5% to ±20%. If these values are applied back to Equation 14, the resulting sensor displacement becomes 1.59µm and an accuracy of ±0.3µm. Micro stepping is available down to 1/256th intervals. However, the error at these intervals will be greater than 90% depending on load as holding torque degrades significantly with higher rates of micro stepping. (Walter, 2016)

Calibration of The Device

In order to calibrate the actuation of the mechanical apparatus, a calibration process must occur. Specifically, the calibration of relation between the stepper motors angular velocity and the translational velocity of the sensor head. As stated previously, the desired sensor displacement is 5µm. Given that the operational frequency is 1000Hz, the translational velocity needed to ascertain 5µm intervals is 5mm/s. However, actuation is driven rotationally via a 47mm pinion. To relate angular velocity to translational velocity the stoichiometric ratio seen in Equation 15, it relates the circumference of a circle to 1 full rotation and the angle (θ) to the displacement of (x) along the circumference.

$$\frac{C}{x} = \frac{1rev}{\theta} \quad eq.15$$

Equation 15 can then be rearranged to solve for (x). By then taking the derivative of (x) and (θ) with respect to time yields variables (V) and (ω). This result is expressed in Equation 16 which describes what angular velocity is needed give a set pinion radius and a desired translational velocity.

$$\omega = \frac{V}{2\pi r} \quad eq.16$$

Since the pinion is known to have a diameter of 47mm and the desired velocity of the sensor must be 5mm/s, the input angular velocity of the pinion is calculated to be 0.3386275rev/sec. Since there is 1:100 gearing ratio in between the pinion and the stepper motor, the stepper motor's angular velocity must be 3.386275rev/sec. The previous calculation holds theoretically, however in practice the pinon and belt may have geometric discrepancies. Thus, the velocity previously calculated is used as a starting point for calibrating actuation.

The calibration was carried out using a coupon designed using Autodesk Inventor and 3D printed with the material RUP-60 on the CARBON M1 3D printer seen in Figure 37. The coupon is a 20mm x 20mm x 50mm rectangular box with a 10mm x 10mm x 30mm rectangular box removed from one of its sides to later aid in depth calibration.

Figure 37 - Coupon used to calibrate mechanical actuation of the sensor

Before the coupon can be used, it's dimensional accuracy must be quantified. This was accomplished via measuring the coupons geometry with a Mitutoyo micrometer as seen in Figure 38.

Figure 38 - Measurement technique used to verify the calibration coupon

The measurement was repeated 3 times with readings of 50.003µm, 50.003µm and 50.002µm. Based off these measurements, the dimensional height of the coupon will be set to 50,003µm. However, for the purposes of iteration, the height of the coupon will be assumed to be 50,000µm, and an inherent error of ±3µm will be added at the end any calculation to account for this assumption.

Calibrating the mechanical device first starts with ensuring that the baseplate and sensor head are level and plumb. This was accomplished through the use of a Starrett 98 series Engineer's spirit level with a NIST calibrated accuracy of ±0.47mm/m or ±0.0264375º. (Starret, 2018) From this point, the coupon was placed on a calibrated granite table and scanned with the sensor head. The data points collected from the scan are then opened in excel. Finding the top and bottom of the coupon within the data is accomplished by finding the rows of data that transition from empty space to cells that contain valid height data to determine the number of scans taken over the distance of 50,000µm.

By subtracting the top row number from the bottom row, the overall number of measurements along the surface of the coupon can be found. Since the coupon is 50000µm and the desired scan interval is 5µm, there should be 10,000 samples overall. Using this information with the actual number of scans along the surface of the coupon and the current angular velocity of the stepper motor, Equation 17 is then used to calculate the new angular velocity that will be programed to the PLC.

$$New\ \omega = Current\ \omega \times \frac{measured\ \#\ of\ Points}{10{,}000} \quad eq.17$$

This process of scanning, counting the number of scans, and calculating a closer angular velocity is repeated until the result converges. This iterative method of calibration is the same as used to calibration the stepper motor extrusion length of FDM 3D printers. The results from this celebration test can be seen in Table 4 .

Iteration	Stepper Angular Velocity (rev/s)	Number of Punts in Coupon
1	3.120709	11,435
2	3.568531	10,247
3	3.656673	9,896
4	3.618643	9,950
5	3.600549	10,009

Table 4 - Variables and results used for calibration iterations

Linear Rail Alignment

Proper alignment of the linear rail is extremely important. Misalignment with the Sensor Carriage Assembly and the idler pulleys will result in a nonlinear displacement relationship between drive belt and carriage. This relationship is best shown in Figure 39 and expressed by the Pythagorean Theorem seen in Equation 18.

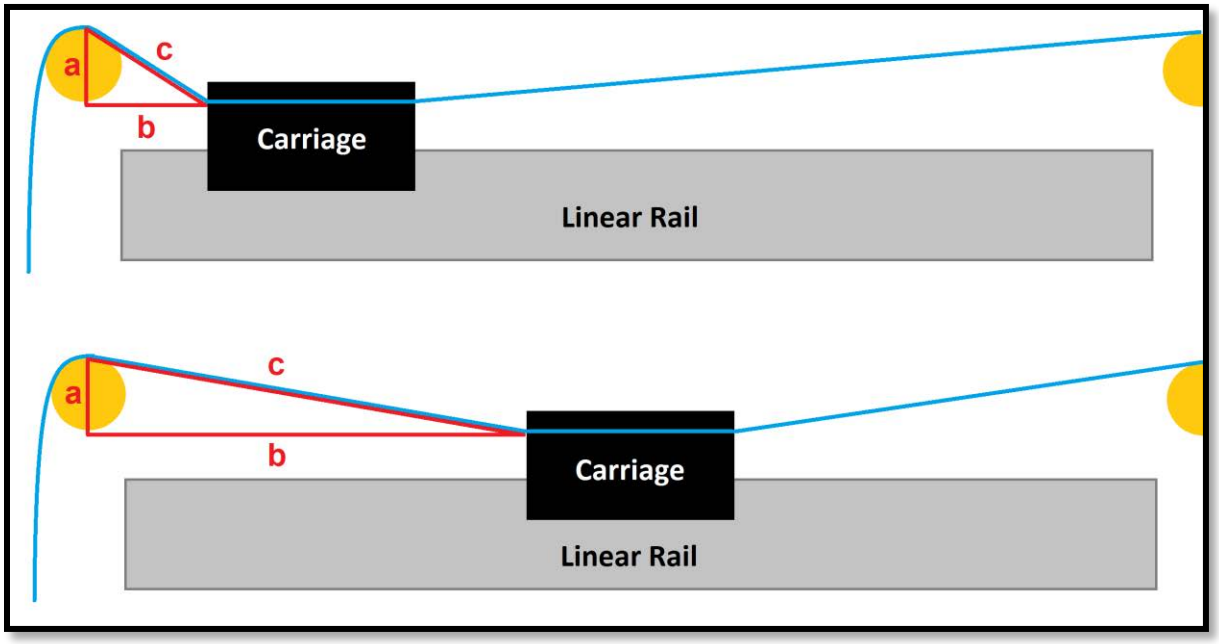


Figure 39 - Geometric relation of belt misalignment

$$c = \sqrt{a^2 + b^2} \qquad eq.18$$

For example, if there is a belt misalignment of 1mm (A = 1mm), and the carriage is 10mm from the pinion (B = 10mm), the difference between linear actuation of the carriage and the belt will be 10µm. At B=100mm the difference will be 5µm, and at B=1000mm the difference will be 0.5µm. Table 2 shows the calculated errors for varying degrees of misalignment (A) with carriage distance (B).

A \ B	10mm	100mm	1000mm
1mm	10µm	5 µm	0.5 µm
3mm	307.8 µm	31.2 µm	3.1 µm
5mm	1180.3 µm	124.9 µm	12.5 µm

Table 5 – Computed mathematical significance of belt misalignment

It is important to note that these values do not represent the measurement error of misalignment. To calculate error per measurement Equation, 19 is used. Equation 19 states that the difference between the belt length (C) and carriage length (B) from an initial position (B_0) to an end position (B_1) can be subtracted from on another to calculate error.

$$error = \left(\sqrt{A^2 + B_0^2} - B_0\right) - \left(\sqrt{A^2 + B_1^2} - B_1\right) \quad eq.19$$

Given that our accuracy must be 5µm and our operational frequency over one second is 1000, the carriage must travel 5mm given 1000 measurements. The actuation error over 1000 measurements can then be described by Equation 20. In this equation there is an assumed misalignment of 1mm with a starting carriage position of 10mm. The results can be seen in Equation 21.

$$\frac{error}{1000} = \left(\sqrt{1^2 + 10^2} - 10\right) - \left(\sqrt{1^2 + 15^2} - 15\right)\Big/_{1000} \quad eq.20$$

$$error/_{1000\ samples} = 16.5\mu m \quad eq.21$$

Being that the individual actuation error per sample is 16.5nm, it would be acceptable to tolerance the idler pulleys to be ±1mm. However, in the interest of producing the best 3D scanning apparatus possible, machining tolerances will be set to the limitation of the milling machine of 100µm.

Tuning the ICP Stitching Algorithm

While the functions that MATLAB provides to implement the ICP algorithm are relatively straight forward, tuning the algorithm to produce the correct result is difficult when attempting to process nonlinear and nontrivial data. The goal for this test is to stitch together two-point cloud surfaces from data obtained from the prototyped 3D scanner. These point clouds represent the back right and back left sides of a nondisclosed 3D printed object seen in Figure 40. For the purposes of explanation within this section the point cloud on the left will be refer to as “Point Cloud A” and the point cloud on the right will be refer to as “Point Cloud B”. Time and RMS data for the best results of each method can be found in Table 3 on page 74.

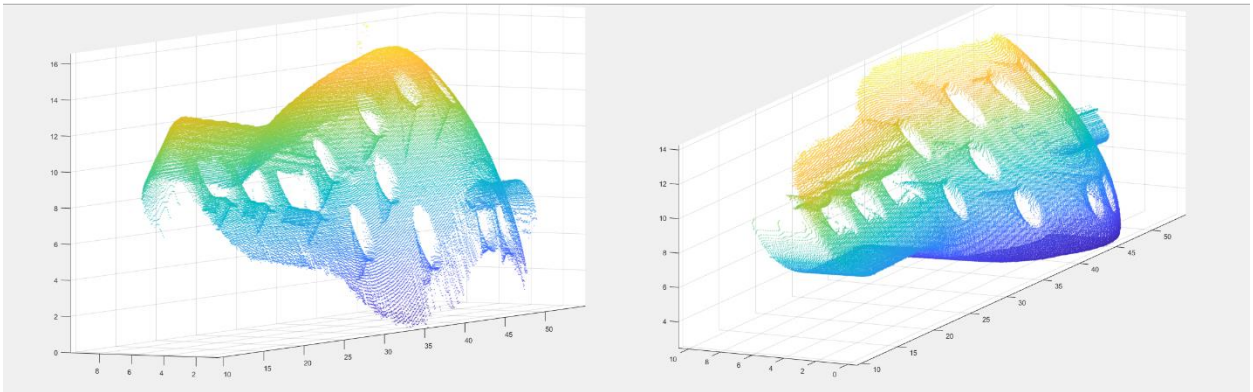


Figure 40 – Rendered point clouds of a 3D printed object scanned at two different perspectives

These test scans were conducted in roughly 120º intervals from one another to roughly represent data obtained from the delta configuration sensor design. For the purposes of iteration Point Cloud A will remain stationary, while the transformation matrixes will be applied to Point Cloud B.

Coarse ICP Tuning

For the purpose of coarsely tuning the algorithm, three arguments were set and held constant for the later fine-tuning arguments. The first of these arguments to be set was the 'Initial Transform' argument. As seen in Figure 41, the initial orientations for the two point clouds are both rotationally and translationally misaligned.

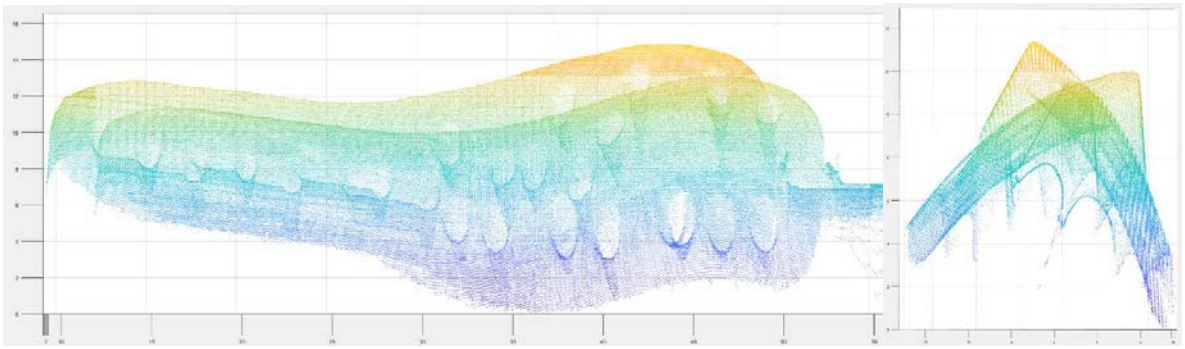


Figure 41 – Initial orientation error, where translation error (Left), and rotation error (Right)

If ICP were to be conducted from this initial orientation the point clouds could stitch together improperly. It then becomes advantageous to set an initial rotation and translation such that the ICP algorithm does not improperly iterate the wrong surfaces together. As seen on the right side of Figure 42 , a rotation of $\frac{4\pi}{3}$ was applied such that the correct surfaces will mesh when the ICP algorithm begins. The left image in Figure 42 shows how the applied translation transformation was applied to correct the misalignment from the left image in Figure 41.

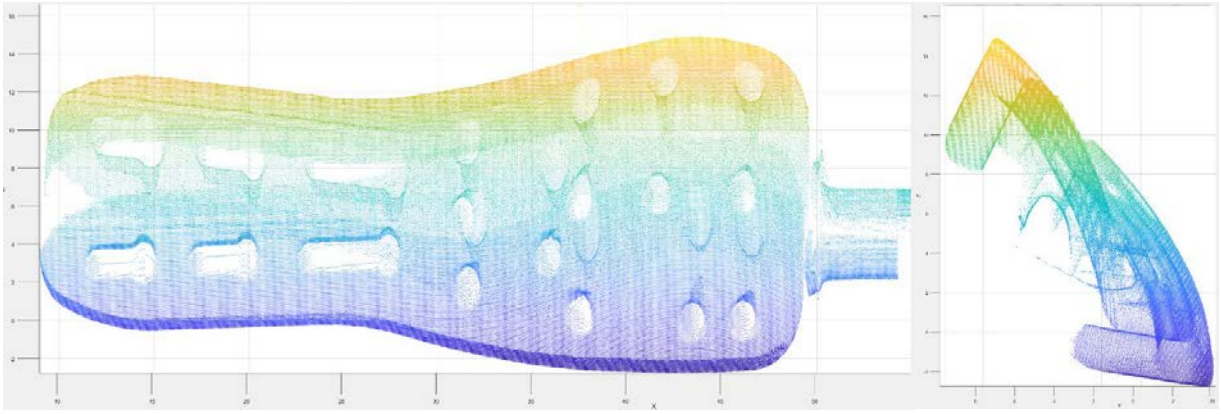


Figure 42 - Point cloud orientation after applying initial transformation

The second argument to be set was the ‘Metric’ argument. This was set to ‘PointToPoint’ rather than ‘PointToPlane’ due to failed results that were traced back to an initial transformation as seen in Figure 43.

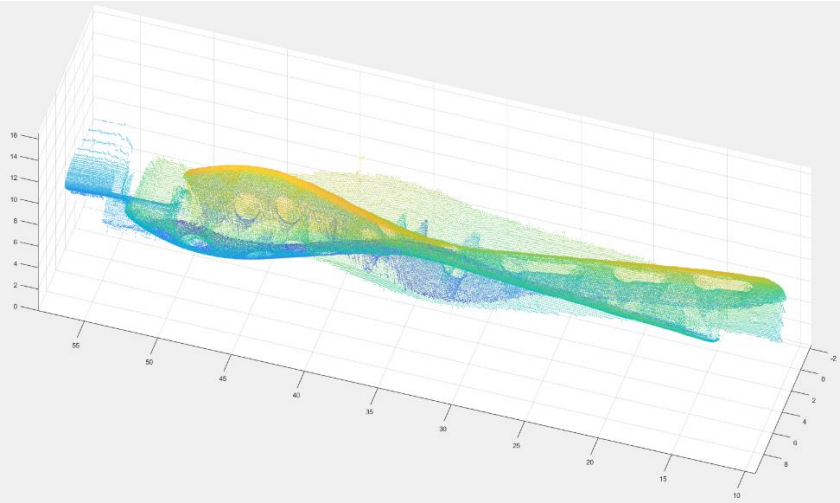


Figure 43 - Failed surface correlated stitching using ‘PointToPlane’ metric

The last “Coarse” ICP tuning argument to be set was the ‘Extrapolate’ argument. As explained previously in the Design section, this parameter is either set to ‘true’ or ‘false’. For the purposes of tuning and obtaining the best stitching result, the input of ‘false’ to improve the accuracy of the stitching operation. The result of running the ICP algorithm with these coarse presets is best shown in Figure 44. From this, it is clear that additional steps are needed for a more accurate stitch.

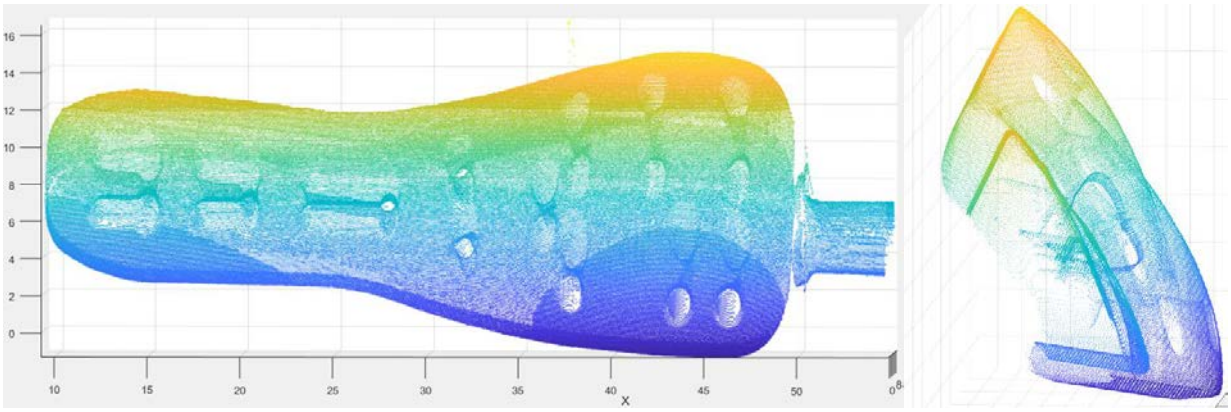


Figure 44 - Resulting point cloud stitch from coarse tuning of ICP algorithm

Fine ICP Tuning

While the "Coarse Tuning" arguments are held constant for the duration of tuning, the proceeding "Fine Tuning" methods vary independently. The following methods for "Fine Tuning" utilized methods that include the physical manipulation of input data to the manipulation of advanced arguments within the ICP algorithm.

Brute Force Method

This method simply applies as many iterations as possible to the ICP algorithm by modifying the 'MaxIterations' metric within the ICP function. The limitation for this method simply becomes the quantity of RAM onboard the computer being used to process the algorithm. During this test the computer fully saturated 32GB RAM with 250,000,000 iterations of the ICP algorithm. The computational intensity of this method is best expressed via RAM consumption seen in Figure 45. The resulting stitch for this method can be seen in Figure 46. RMS results are found in Table 3.

Figure 45 - Task manager view of machines RAM under load of ICP Brute Force Method

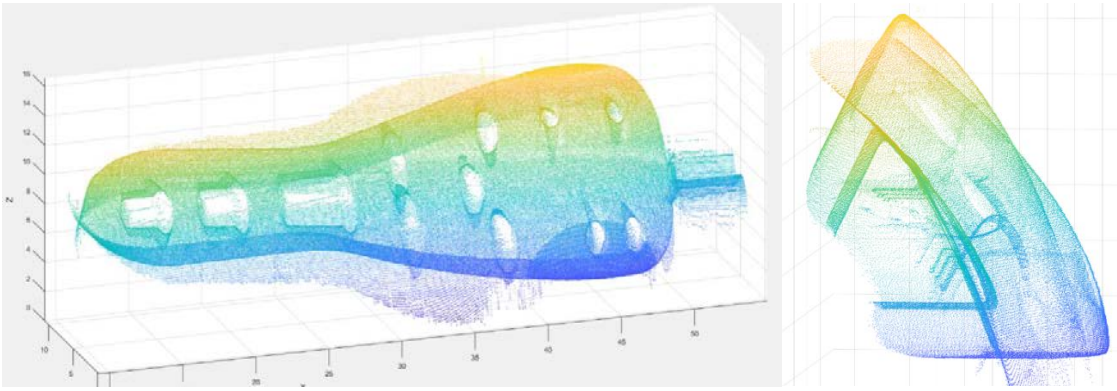


Figure 46 - Resulting point cloud stitch from Brute Force Method

Inlier Stitching Method

The inlier stitching method relies on the 'InlierRatio' argument within the ICP function. This argument sets a percentage of points to use that best match either point cloud. For the purposes of testing inlier ratios of; 100%, 90%, 80%, 70%, 60%, 40%, 30%, 20%, 10%, 5%, and 1% were used to find the best stitch with the lowest RMS value. Figure 47 displays the results for an inlier ratio of 100%, 70% and 40%. Resulting stitches for all tests can be found in Appendix D.

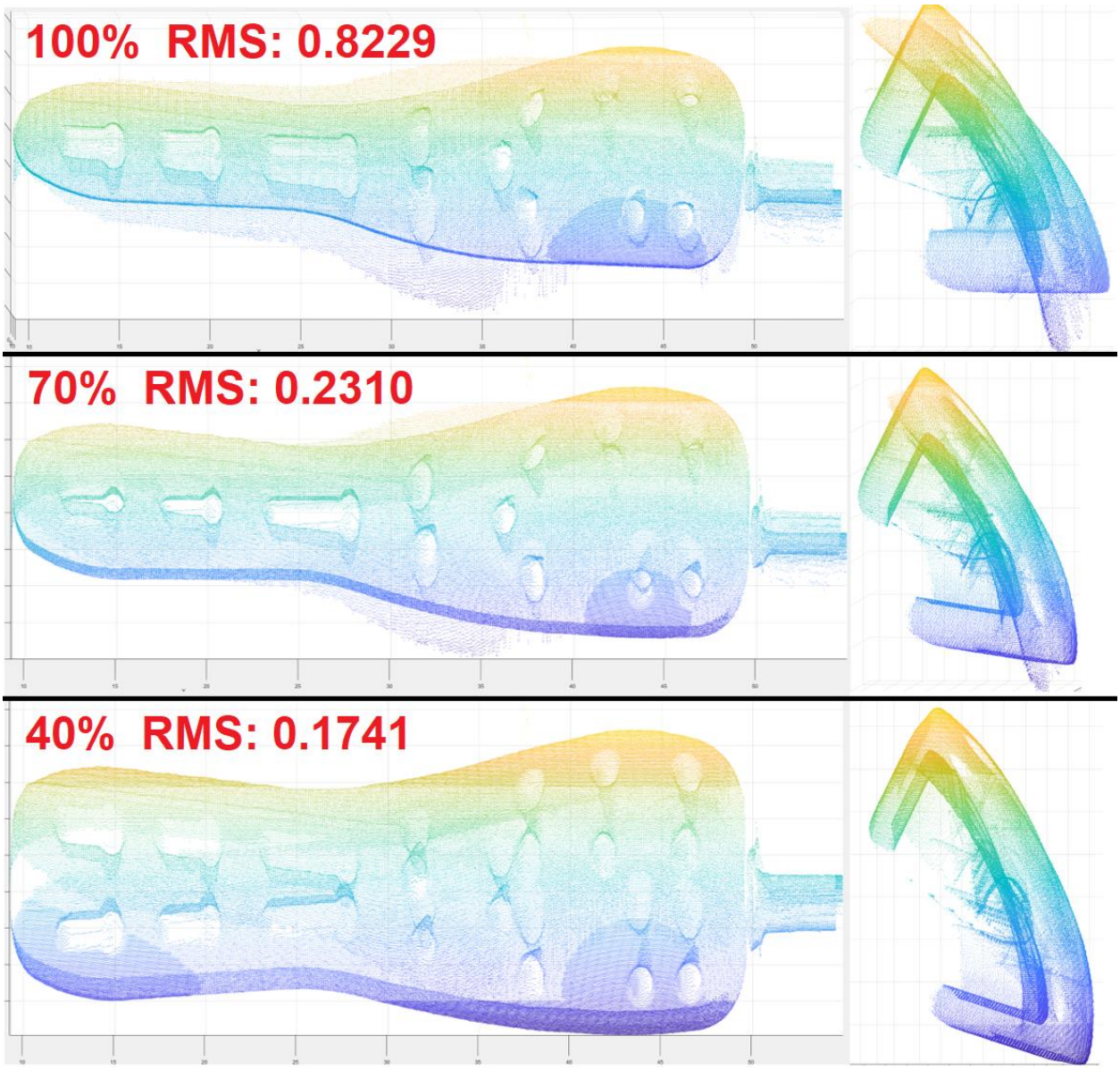


Figure 47 - Result of stitching with Inlier Method for 100%, 70%, and 40%

It is interesting to note as the inlier ratio decreases so does the RMS value. However, hole alignment is best at an inlier ratio of 70%. As the inlier ratio decreases, the two point clouds hole alignment proceeds to deviate while the corresponding side profile improves. This is best seen when transitioning from an inlier ratio of 70% to 40% in Figure 47. This correlation could potentially mean that there is a nonlinearity associated with the sensor when it comes to the perception of depth, and will be discussed further in the Conclusions section.

Crop Stitching Method

Points outside of the overlapping surfaces of either point cloud will not have matching sister points. This will disrupt the ICP algorithms overall accuracy, as these extremity points will carry a statistical and geometric weight throughout the calculation of a transformation matrix. The logic behind this method is to remove all points that do not reflect overlapping surfaces. This is done by loading a spare copy of the raw data in to MATLAB's workspace and cropping out areas of the point cloud that do not overlap one another. The resulting crops from point clouds A and B can be seen in Figure 48.

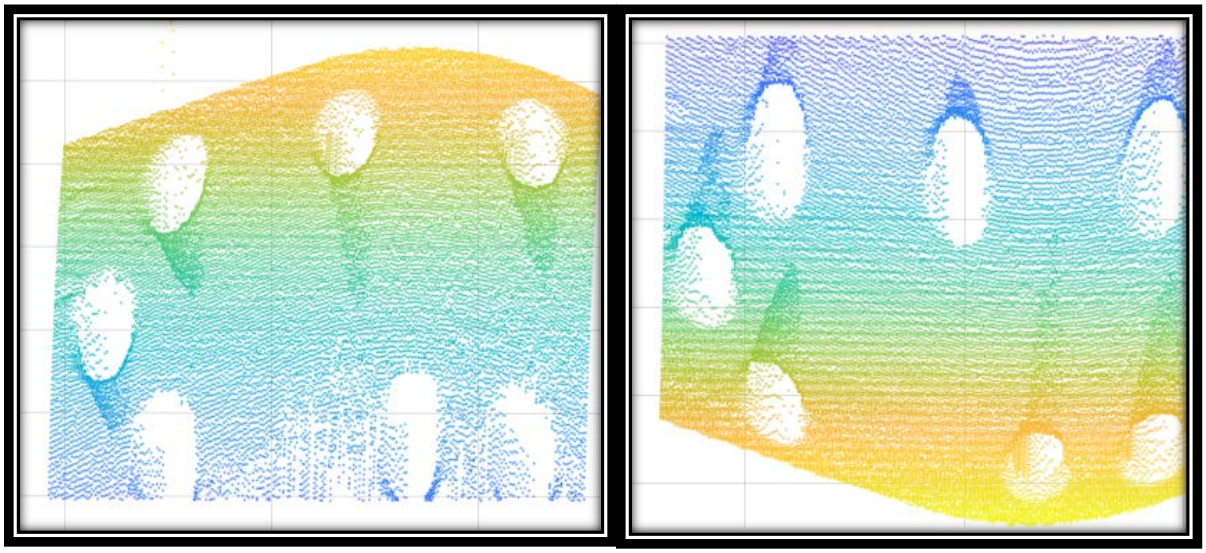


Figure 48 - Result of cropping procedure of point cloud A (left) and B (right)

The process of stitching the two cropped point clouds together are executed under the same guidelines as set in the Brute Force Method. The result of this stich can be seen in Figure 49.

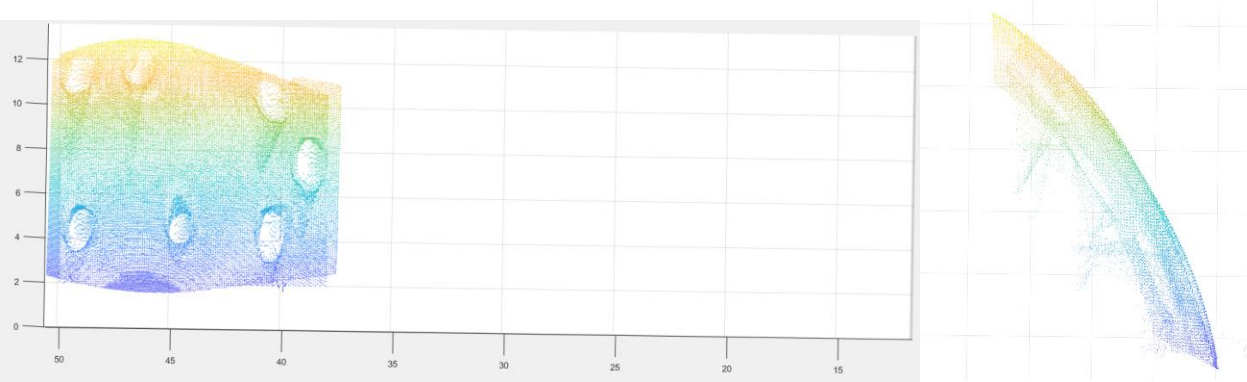


Figure 49 - Result of stitching the two cropped point clouds together via Brute Force

The transformation matrix generated from the previous operation is then applied to the non-cropped data, resulting in the stitch seen in Figure 50. RMS results can be found in Table 3.

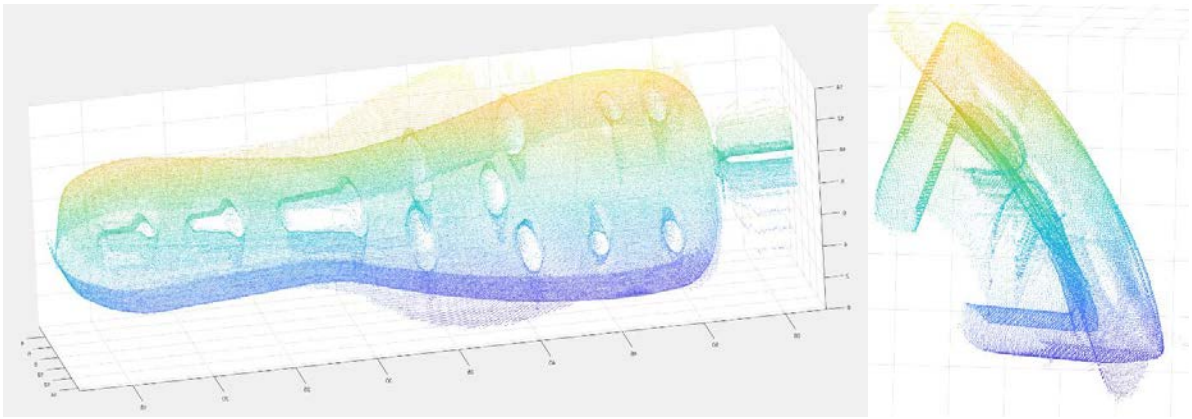


Figure 50 - Result of applying transformation matrix obtained with cropped data to full data

Hybrid Stitching Method

The Hybrid Stitching test utilizes a combination of the Crop Stitching Method and the Inlier Stitching Method. Specifically, the only data points that are removed from the raw data sets is the base where a cylindrical support rod can be seen in previous figures. Through experimentation, the inlier ratio that yielded the best stich for this method was 60%. The resulting stitch for this method can be seen in Figure 51. RMS results can be found in Table 3.

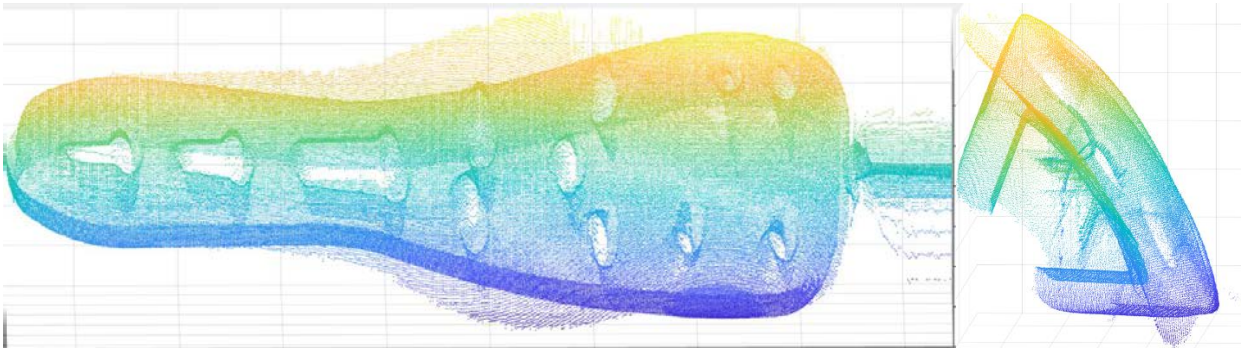


Figure 51 - Stitching result for utilization of the Hybrid method

Determining Overall 3D Scanner Speed

Since speed is a pinnacle to the successful operation of the 3D scanner, a series of experiments were devised and executed to determine the overall speed of the device. This includes determining the time required for acquiring, transferring, and processing data. As mentioned previously in the Design section, the LJ-V7060 sensor was substituted for the LJ-V7080. Before testing can begin, it is critical to understand the implications of using the lower resolution sensor. Figure 52 best displays the issue with using scaled data from the LJ-V7080. The image on the right side of Figure 52 is a magnified version of the left image such that individual points can be seen with a horizontal spacing of 50µm and a vertical spacing of 5µm.

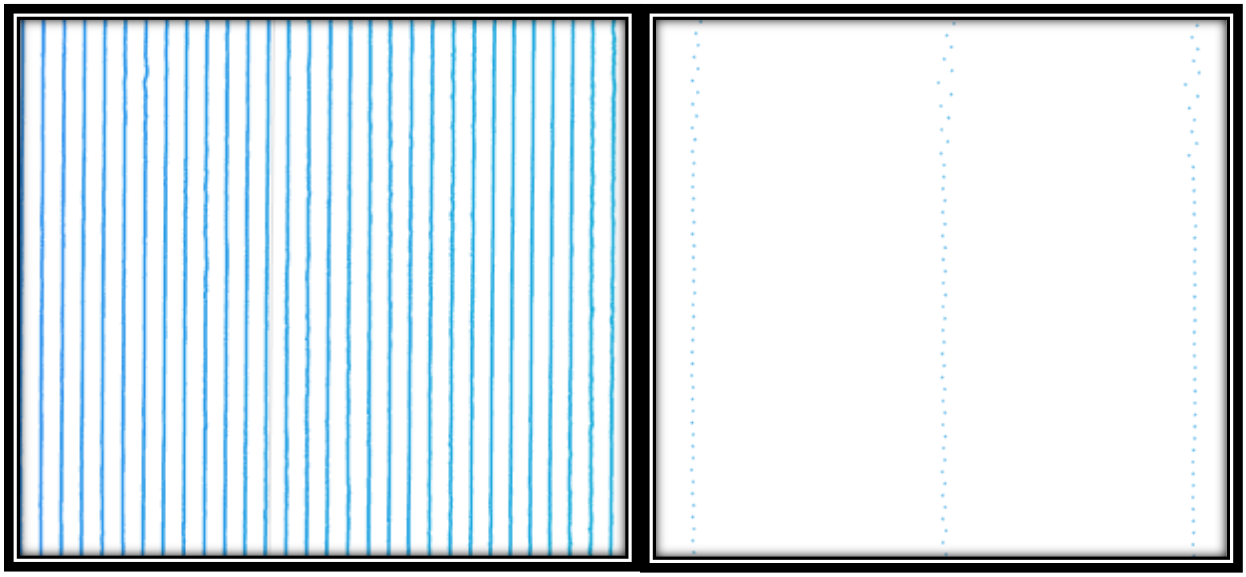


Figure 52 – Rendering of point cloud data showing difference in horizontal and vertical accuracy

Since the vertical resolution of the data is set through the interaction of the mechanical system and the operational frequency of the sensor, the data's vertical points are separated by 5µm as intended. However, due to the capabilities of the sensor, the horizontal resolution is only 50µm. The issue with these data points is simply due to the file being too small. If the goal of these tests is to determine if current algorithms and computing technology are powerful enough to meet the computational time requirements, a file that accurately represents real data at the desired point cloud density must be used. This was done by manually editing the acquired (.csv) data sets through duplication of the data by a factor of 10. This resulted in an increase in file size from

50MB to 500MB. Figure 53 shows the point cloud density as the result of data manipulation, where the image on the right is a magnified version of the left image.

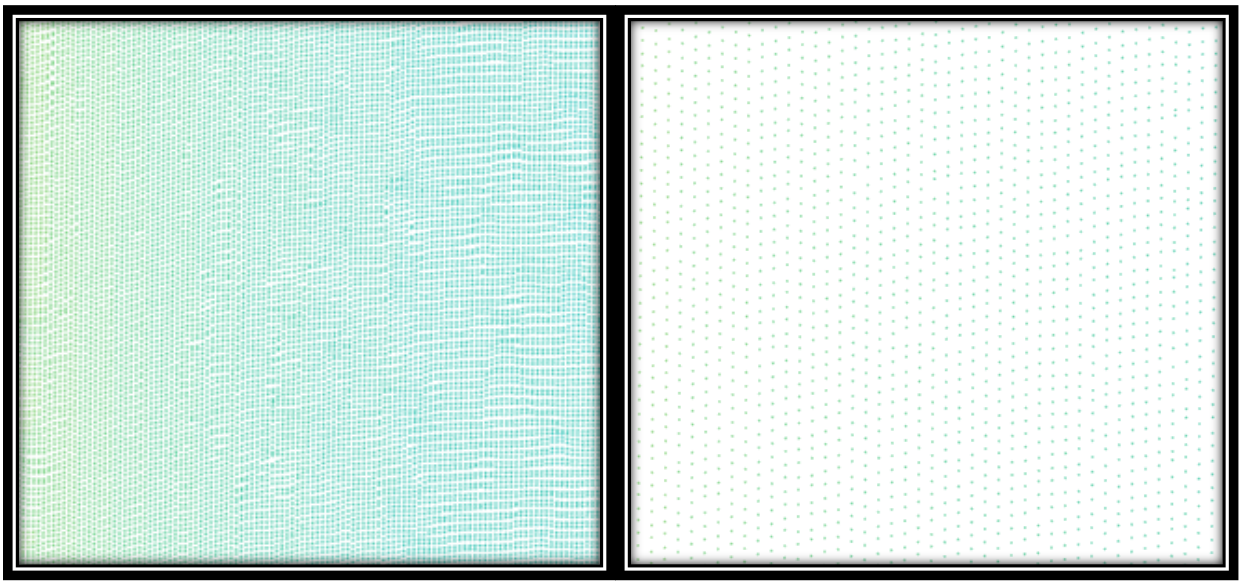


Figure 53 - Rendering of point cloud data after manipulation with point spacing of 5µm

Speed of Data Acquisition

Data acquisition refers to the process of actuating the sensor over an objects surface and then converting the acquired information into a digital file. The time allocated to this process is dependent on the actuation speed of the mechanical apparatus and the height of the object being scanned. Given that the desired spacing between points is 5µm and the operational frequency of the sensor is 1000Hz, the velocity at which the sensor is propagated at can be calculated using Equation 22

$$V_{Sensor} = X_{Spacing} \times Hz_{Sensor} \quad eq.22$$

From here, Equation 23 can be used to calculate the time required to scan the surface of an object. Given that the mechanical apparatus moves at 5mm/s from Equation 22, the object's height will be set to 10cm, resulting in a required scan time (20 seconds for the current configuration of the device). It is important to note that the KEYENCE LJ-V sensors can be safely overclocked up to

4000Hz. If this were to be implemented, the time required to scan the object could be reduced to 5 seconds.

Speed of Data Transfer

Data transfer refers to the process of moving data from one data processing device to another. Specifically, this refers to the rate at which sensor data points are transferred from the LJ-V7001 Controller to the Host Computer. Throughout the prototyping phase of this project, the transfer of data was accomplished via USB 2.0. Ideally, USB 2.0 has a maximum data transfer rate of 60MBps. This means that if 3, 500MB point cloud data sets were to be transmitted over USB 2.0, it would take 25 seconds under ideal circumstances. While this data transfer rate is unacceptable for meeting the time requirements, there are other methods to transfer large quantities of data quickly. The LJ-V7001 Controller does have the capability to transmit data over a Gigabit Ethernet with a bandwidth of 125MBps which would ideally transfer the same amount of data in 12 seconds.

Speed of Data Processing

One of the main hurdles in meeting the time requirements is in the processing of the sensors data. The processing of data can be broken down into two algorithms. The first is the sorting algorithm mentioned in the Generating a Point Cloud section in the Design portion of this paper. The second is the iterative close point algorithm described in detail though out this paper.

Formatting Algorithm

The goal of the formatting algorithm is to convert the raw data from the sensor's controller into properly formatted data for use in later processing. The following test encompasses the implementation of two separate algorithms. The first is an algorithm written using MATLAB, while the second algorithm was written using C# with utilization of .NET framework. For this test, the two algorithms are required to format the raw 500MB data file previously described into a properly formatted point cloud.

The MATLAB algorithm speed test required the algorithm to use the modified data .CSV file type to import the data and required MATLAB to save the resulting output as a .CSV file. The results for this test can be seen in Figure 54. It is interesting to note how inefficient the MATLAB environment is at saving .CSV files as this took over 9 minutes to complete.

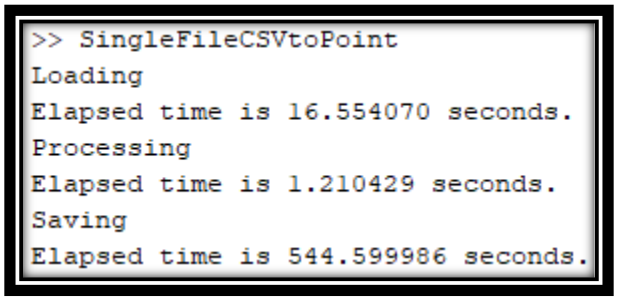


Figure 54 - MATLAB Command Line output for sorting data from a (.CSV) file to a (.CSV) file

The next test for MATLAB required the algorithm to use the same .CSV file to import data, however the MATLAB algorithm then saves the resulting output as a .MAT file. The results for this test can be seen in Figure 55.

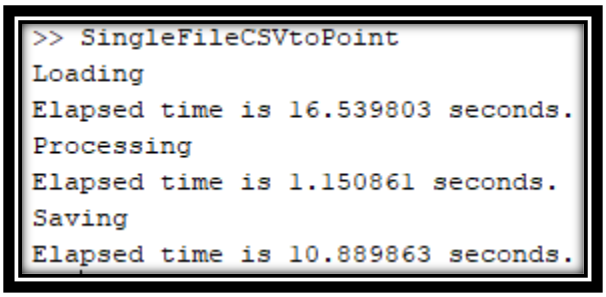


Figure 55 - MATLAB Command Line output for sorting data from a (.CSV) file to a (.MAT) file

The final test for MATLAB required the algorithm to import data in a .MAT file type and save the output as a .MAT file. The results for this test can be seen in Figure 56.

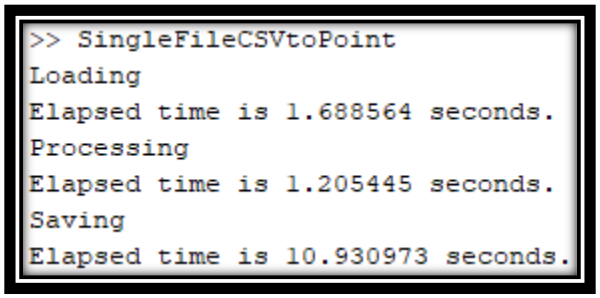


Figure 56 - MATLAB Command Line output for sorting data from a (.MAT) file to a (.MAT) file

Regarding the speed of the C# program, two tests were run. The first used the modified data from a .CSV file, while the data points processed by the algorithm are saved to a .CSV file. The second test used the same input file method however the output of the program was saved to a .TXT file. The results for both tests were identical and can be seen in Figure 57.

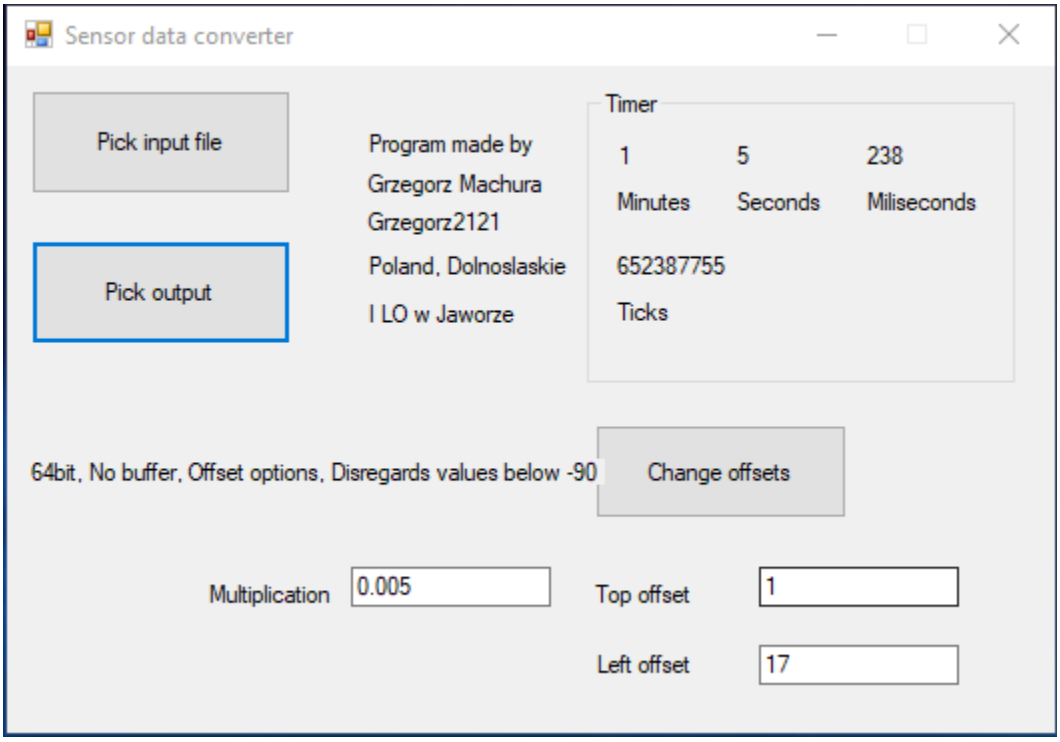


Figure 57 - C# program GUI with Timer results in top right of window

Stitching Algorithm

To determine the speed of the stitching algorithm, the modified point cloud file was not used. This is because the current implementation of the ICP algorithm within the MATLAB environment recommends noncompeting of data by a factor of 10 to reduce the computational intensity. For this reason, the point cloud was not down sampled such that use of the original data acquired at 50µm would reflect downscaling of data acquired at 5µm. In addition, the modified data points that reflect 5µm density of points was not used due to the geometric configuration of points, resulting from the modification process as seen in Figure 58.

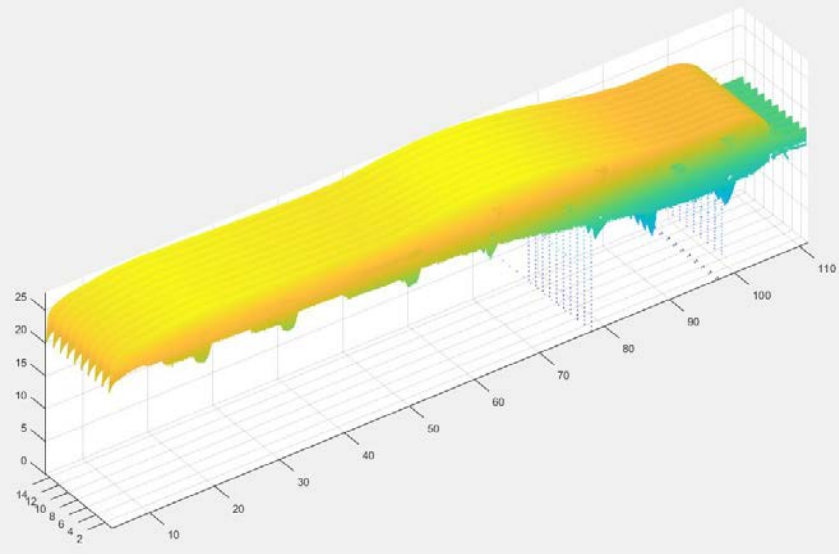


Figure 58 - Rendering of Modified point cloud data

To determine the time the ICP algorithm took to stitch data together, the 'tic/toc' function in MATLAB was used. The results for testing the Brute Force, Cropping, and Inlier methods can be seen in Table 3 below.

Method	Time	RMS
Brute Force	27.07	0.8693
Inlier	29.55	0.2310
Crop Stich	3.58	0.2952
Hybrid	69.65	0.1647

Table 6 - Time and RMS results from implementation of various ICP techniques

CONCLUSIONS

This prototype represents a section (1/3) of an idealized system for industrial purposes and was used to validate its application through a bench top implementation. A 3D scanning apparatus capable of high resolution and high speed was successfully prototyped. While the ideal sensor was not used for prototyping, the overall working principal for this technique of 3D scanning was shown to be a valid alternative to conventional methods.

Device Resolution

Mechanically speaking, the device was built with machining tolerances down to ±100µm. However, a machining error pertaining to drive belt alignment occurred and the linear rail alignment was offset by 1mm. However, as indicated by the Linear Rail Alignment calculations in the Testing section, this error has a negligible effect on the vertical mechanical resolution of the device. The overall resolution of the 3D scanning apparatus is mainly dependent on the quality of the triangulation sensor. Practically speaking, the 3D scanning apparatus only controls the vertical resolution of the scan, while the horizontal and depth components are sensor dependent. This being the case, only the vertical resolution was tested. The results from vertical resolution testing and calibration demonstrate that the mechanical apparatus is capable of constant 5mm/s actuation to enable the required 5µm scan intervals. While the exact accuracy and repeatability of device actuation were not quantified, the results from this test showed that any geometric data acquired from the 3D scanning apparatus would be representative of the objects true geometry for proof of concept purposes. Regarding the resolutions that are solely sensor dependent, a later study is necessary to verify the exact sensors specifications provided by the manufacturer.

Point Cloud Stitching

Stitching together two nontrivial point clouds quickly escalated into a highly complicated endeavor. Given the built-in tools MATLAB provides for implementing the ICP, the stitching of two data sets obtained by the 3D scanning apparatus was accomplished. However, the accuracy of the stitching algorithm is not acceptable to meet the initial requirements set at the beginning of this thesis. There may be several reasons for this. One possible explanation for inaccuracy of the

stitching algorithm may be inherent to the algorithm. An additional reason for the inaccuracy of the stitch may be due to a nonlinearity associated with the depth perception of the sensor. However, this is only speculation and needs to be further examined in future work.

Device Speed

The importance of speed must be stressed again when grasping the technological niche that this device attempts to fit. The feat of acquiring, transferring, and processing 500MB of data per sensor in under a minute is an extremely difficult task. The raw computing power necessary for such a task required a full bandwidth saturation with the latest computational hardware available. Regarding the specific times involved, Figure 59 best shows the current speed of the device and theoretical speed broken into the times required for acquisition, transfer, sorting, and stitching.

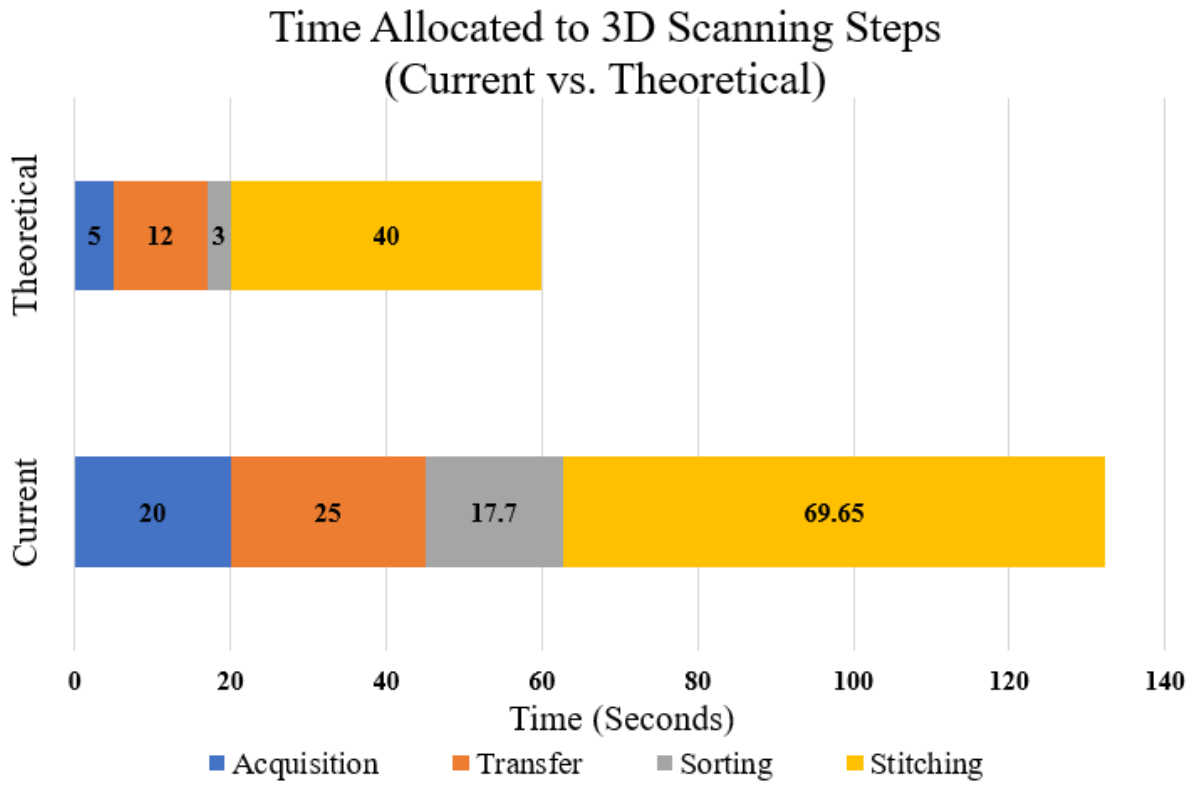


Figure 59 – Component time allocation steps for current device and theoretical minimum

Acquisition

As the testing section indicates, acquisition time (blue) is limited by the operational frequency of the sensor and thus the linear velocity of actuation. Given a generic scanning length of 10cm, the required time to scan an object is 20 seconds. However, by overclocking the sensor to 4000Hz, the acquisition velocity is quadrupled and thus the new theoretical time for acquisition shrinks to 5 seconds.

Transfer

Regarding the transfer of data (orange), the testing section indicated that transferring the data to the processing computer over USB 2.0 consumes 25 seconds. Theoretically speaking, if Gigabit Ethernet were to be used, the time would shrink to 12 seconds. The time of transfer could be reduced further if the latest transfer methods were implemented within the KEYENCE hardware such as Thunderbolt. However, this would require a large investment in R&D and will be discussed further in the Future Work section.

Sorting

Through testing, the overall computational speed of MATLAB's applied LAPACK BLAS libraries are far superior to any conventional looping method as seen in Figures 54-57. When it comes to the reading and writing of data, MATLAB fails in comparison to the basic efficiency of a standard programing language. The process of sorting the raw data into the proper format via MATLAB (grey) consumes approximately 17.5 seconds when data must be read via .CSV. This time could be further reduced to 3 seconds by eliminating MATLAB's need to read the input data from a .CSV file and directly feeding MATLAB the raw data in a .MAT file type. Again, this change would require collaboration with KEYENCE.

Stitching

Regarding the time required for stitching (yellow) the resulting data varies depending on the method. The best stitching result occurred in the Hybrid method with a total time for computation

of 69.65 seconds. However, the Crop Stitching method took a mere 3.58 seconds to complete. This can mainly be attributed to a significant decrease in data needed to be processed by the ICP algorithm due to cropping of most of the data set. Overall the ICP algorithms do need additional work and will be discussed in the Future Work section. However, it is important to note that the variable that most affects ICP stitching time is the accuracy of the initial transformation matrix. The accuracy of this matrix is solely dependent on the pre-known positions of the scanned object in relation to the sensor or the pre-known positions of the sensors in relation to each other. Given these unknowns, a theoretical time for the ICP algorithm cannot be determined. However, the theoretical times for acquisition, transfer, and sorting leave the ICP algorithm with an available 40 seconds of computational time.

In its current state the device does not meet the required 60 seconds to complete its task, and requires an overall time 132.15 seconds to acquire, transfer, and process data. Given the recommended hardware and software changes discussed within this thesis, the system is theoretically capable of meeting the time requirement.

FUTURE WORK

Implementing GPU Computation

One potential method for improving the overall computational speed of either algorithm is through the implementation of GPU computing. The optimal hardware for implementation would be a CUDA-enabled NVIDIA GPU. Specifically, CUDA refers to the application programming interface (API) proprietary to modern NVIDIA GPU's. The optimal solution for linear algebra computations could utilize a new GPU technology called "Tensor Cores". This technology has shown computational performance roughly (9x) greater than conventional GPU cores. "Each Tensor Core provides a 4x4x4 matrix processing array which preforms the operation D = A*B +C, where A, B, C and D are 4x4 matrices as Figure 60 shows." (Jeremy Appleyard, 2017)

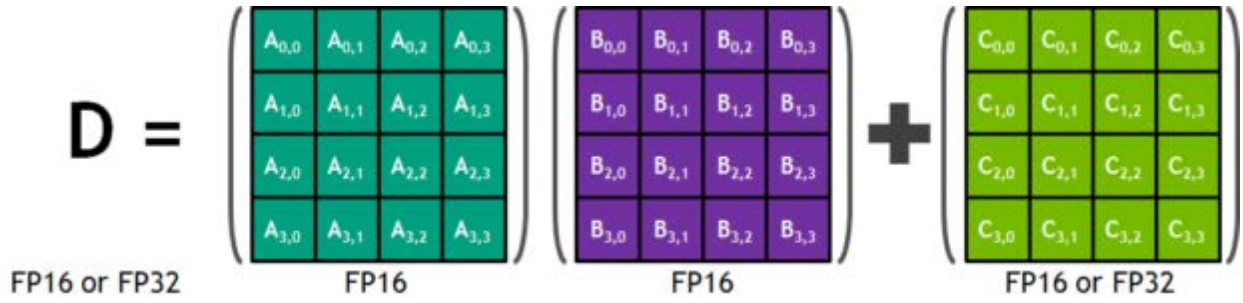


Figure 60 - Tensor Core 4x4x4 matrix multiply and accumulate

This technology could be implemented utilizing MATLAB's Parallel Computing Toolbox. Specifically, this toolbox allows users to "solve computationally and data-intensive processors using multicore processors, GPUs, and computer clusters." (MathWorks, 2018)

Replacing Drive Belt with Lead Screw

Regarding the mechanical design of the 3D scanning apparatus, it may be advantageous to further refine the overall design. While the belt driven system was shown to be a reliable method for actuation, further research and development should explore the possibility of replacing the belt driven system through a high precision threaded rod for actuation. This would reduce the number of moving parts in the device and cut down on the overall footprint of the apparatus. However, it is important to note that this may introduce more vibrational noise directly to the sensor head as

the vibrational noise generated by the stepper motor will be transmitted through a different medium. Further calculations and testing of these effects will be needed before moving forward. Future work for the mechanical system also includes integration of three sensors in the delta configurations as described in the Design section. Such integration will allow the system to simultaneously scan all exterior surfaces of a desired object. Implementation of the full design will require additional electronics; thus, extensive integration will be needed.

Integrating and Automating the System

In its current state, the system requires a user to manually trigger actuation of the sensor and manual triggering of data capture. This system will need to be automated such that timing of these actions will no longer rely on manual user operation. Furthermore, a dedicated computer will be needed such that data transfer and processing can occur on the same machine.

ICP Algorithm

Further refining the process of stitching point clouds is needed to achieve an overall point cloud map with an accuracy of 5μm. A detailed study on the non-linearity's of collected depth data may resolve the issues encountered in the Testing section. However, the error may be due to a flawed algorithm. To solve this, further development and research of the computerized intelligence behind the ICP algorithms that solve the Procrustes problem. An entire Ph.D. dissertation could potentially be dedicated towards perfecting this aspect of the Thesis.

KEYENCE LJ-V System Improvements

One of the key pitfalls in processing the data is the time MATLAB requires to import .CSV data. If KEYENCE were to produce a software patch for the exportation of data out of the LJ-V7001 Controller in .mat format, loading times would be significantly decreased as indicated by testing. Licensing issues may require partnership with KEYENCE and MATLAB. Additionally, if KEYENCE were to implement support for faster transfer, upload times to MATLAB would drop from 25 seconds with USB 2.0, to 2.5 seconds for USB 3.1, and 1.25 seconds for Thunderbolt. Such a modification would require substantial funding for implementation.

REFERENCES

Afanche Technologies, Inc. (2018, April 27). *Tutorial: Point Cloud Data Processing*. Retrieved from Afanche Technologies : https://www.afanche.com/tutorial-point-cloud-data-processing

AMCI. (2018, March). *Industrial automation resources*. Retrieved from Advanced Micro Controls INC: https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/stepper-vs-servo/

Artec3D. (2018, April 25). *3D Scanners.* Retrieved from Artec3D: https://www.artec3d.com/portable-3d-scanners/artec-spider#specifications

B.S. Verma, I. I. (2008). Impact of computers in radiography: The advent of digital radiography, Part-2. *Indian Journal of Radiology and Imaging*.

Besl, P. J. (1992). A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 239-256.

Borini, S. (2009, August 19). *How does BLAS get such extreme preformance?* Retrieved from Stack Overflow: https://stackoverflow.com/questions/1303182/how-does-blas-get-such-extreme-performance

Chen, Y. a. (1992). Object Modelling by Registration of Multiple Range Images. *Image Vision Computing. Butterworth-Heinemann,* 145-155.

Ciscle, A. (n.d.). Head CT scan.

COORD3 Metrology. (2014, Nov 11). *50 years of coordinate measuring machine industry developments and history*. Retrieved from coord3-cmm: http://www.coord3-cmm.com/50-years-of-coordinate-measuring-machine-industry-developments-and-history/

Daniels, F. R. (1921). *Machinery Index to Vol. XXVIII.* New York: The Industrial Press.

David Fofi, T. S. (2004). *A comparative survey on invisible structured light.* Le Creusot, France: IUT Le Creusot.

Fellicious, C. (2016, March 30). Is MATLAB a programming language like C & C++? Thiruvananthapuram, Kerala, India.

Higinio Mora, J. M.-P.-G.-G. (2016, October 21). *Computational Analysis of Distance Operators for the Iterative Closest Point Algorithm*. Retrieved from Journals PLOS One: http://journals.plos.org/plosone/article/authors?id=10.1371/journal.pone.0164694

Jeremy Appleyard, S. Y. (2017, October 17). *Programming Tensor Cores in CUDA 9*. Retrieved from NIVIDIA Developer Blog: https://devblogs.nvidia.com/programming-tensor-cores-cuda-9/

Keyence. (2018, January 23). *White Papers: Keyence Corporation.* Retrieved from Keyence Web site: https://www.keyence.com/mykeyence/?ptn=001

Laser Design, Inc. (2018). *rx-desktom-scanner*. Retrieved from Laser Design: https://www.laserdesign.com/products/rx-desktom-scanner/

libpointmatcher. (2018). *Supported file types and importing/exporting point coulds*. Retrieved from libpointmatcher: https://libpointmatcher.readthedocs.io/en/latest/ImportExport/#descmaptable

Machura, G. (2018, March 31). Method for Converting Image Datat to Point CLoud Data Format Using C# in Comparison to MATHLAB. Jawor, Lower Silesian Voivodeship, Poland.

MathWorks. (2018). *Call LAPAC and BLAS Functions*. Retrieved from MathWorks.com: https://www.mathworks.com/help/matlab/matlab_external/calling-lapack-and-blas-functions-from-mex-files.html

MathWorks. (2018). *Computer Vision System Toolbox/Functinos/pcregrigid*. Retrieved from mathworks: https://www.mathworks.com/help/vision/ref/pcregrigid.html#mw_b4945ac5-fbbb-4509-ac68-2dd28e0198a2

MathWorks. (2018, April 1). *Parallel Computing Toolbox*. Retrieved from MathWorks.com: https://www.mathworks.com/products/parallel-computing.html

Mayo Clinic. (2015, March 25). *CT Scan*. Retrieved from Mayo Clinic: https://www.mayoclinic.org/tests-procedures/ct-scan/about/pac-20393675

Mitutoyo. (2018, Febuary 21). *White Papers: Mitutoyo America Coropration.* Retrieved from Mitutoyo America Coropration Web site: https://ecatalog.mitutoyo.com/LEGEX-5007009001200-SERIES-356-Ultra-high-Accuracy-CNC-CMM-C1505.aspx

Pomerleau, F. &. (2015, May). *A Review of Point Cloud Registration Algorithms for Mobile Robotics.* Retrieved from Research Gate: https://www.researchgate.net/publication/277558596_A_Review_of_Point_Cloud_Registration_Algorithms_for_Mobile_Robotics

Roberto Marani, V. R. (2016, January 12). *A Modified Iterative Closest Point Algorithm for 3D Point Cloud Registration.* Retrieved from Wiley Online Library: https://onlinelibrary.wiley.com/doi/full/10.1111/mice.12184

Schönemann, P. H. (1966). *A Generalized Solution of the Orthogonal Procrustes Problem.* Phychometric Labratory University of North Carolina.

Starret. (2018, March 25). *metrology product detail*. Retrieved from Starrett.com: http://www.starrett.com/metrology/product-detail/98-12

Walter, M. (2016, August 29). *How Accurate is Microstepping Really?* Retrieved from HACKADAY: https://hackaday.com/2016/08/29/how-accurate-is-microstepping-really/

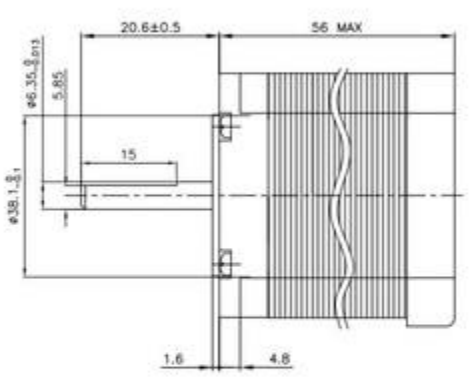
Zhang, Z. (1998). *A Flexible New Technique for Ccamera Calibration.* Redmond: Microsoft Research.

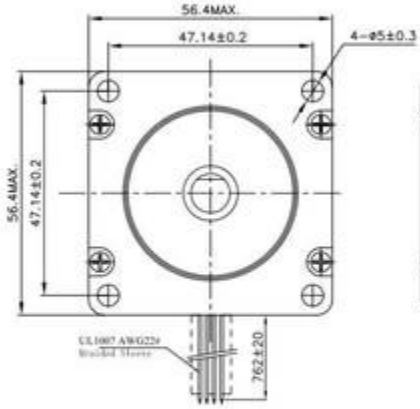
APENDIX A

Model			LJ-V7080
Mounting conditions			Diffuse reflection
Reference distance			80 mm 3.15"
Measurement range	Z-axis (height)		±23 mm 0.91" (F.S.=46 mm 1.81")
	X-axis (width)	NEAR side	25 mm 0.98"
		Reference distance	32 mm 1.26"
		Far side	39 mm 1.54"
Light source	Type		Blue semiconductor laser
	Wavelength		405 nm (visible beam)
	Laser class		Class 2 Laser Product (IEC60825-1, FDA(CDRH) Part 1040.10*1)
	Output		4.8 mW
Spot size (reference distance)			Approx. 48 mm × 48 µm 1.89" × 0.001890"
Repeatability	Z-axis (height)		0.5 µm 0.000020"*2*3
	X-axis (width)		10 µm 0.000394"*2*4
Linearity	Z-axis (height)		±0.1% of F.S.*5
Profile data interval	X-axis (width)		50 µm 0.002"
Sampling cycle (trigger interval)			Top speed: 16 µs (high-speed mode), Top speed: 32 µs (advanced function mode)*6
Temperature characteristics			0.01% of F.S./°C
Environmental resistance	Enclosure rating		IP67 (IEC60529)*7
	Ambient light		Incandescent lamp: 10,000 lux max.*8
	Ambient temperature		0 to +45 °C 32 to 113 °F*9
	Relative humidity		20 to 85 % RH (No condensation)
	Vibration resistance		10 to 57 Hz, Double amplitude 1.5 mm 0.06", 3 hours in each of the X, Y, and Z directions
	Shock resistance		15 G/6 ms
Material			Aluminum
Weight			Approx. 400 g

*1 The laser classification for FDA(CDRH) is implemented based on IEC60825-1 in accordance with the requirements of Laser Notice No. 50.
*2 This value is from a case in which measurement has been performed with a reference distance with 4,096 times of averaging.
*3 The measurement targets are KEYENCE standard targets. This value is from a case in which the average height of the default setting area has been measured in height mode. All other settings are default.
*4 The measurement target is a pin gauge. This value is from a case in which the position of the intersection between the rounded surface of the pin gauge and the edge level has been measured in position mode. All other settings are default.
*5 The measurement targets are KEYENCE standard targets. The profile data is from a case in which measurement has been performed with 64 times of smoothing and 8 times of averaging. All other settings are default.
*6 For high-speed mode, when the measurement area is at its minimum, binning is ON, image capture mode is set to standard, and parallel image capture is ON.
All other settings are default. For advanced function mode, when the measurement area is at its minimum, binning is ON and image capture mode is set to standard. All other settings are default.
7 This value is from a case in which the sensor head cable (CB-B) or extension cable (CB-B*E) has been connected.
*8 This is the illuminance for the light-receiving surface of the sensor head during white paper measurement when light has been shined onto the white paper.
*9 The sensor head must be mounted on a metal plate for use.

Product Name	Hybrid Stepper Motor MT-2303HS280AW-OB				RevC
Made by	P.T	Approve	Y.L	Date	3/17/2016

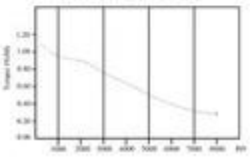




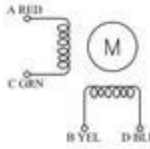
Laser Marking

Items	Specs	Items	Specs
Phase Number	2 phases	Step Angle	1.8°±5%
Driving Voltage	24-48 VDC (Recommended)	Rated Current	DC 2.8A / Phase
Resistance(20℃)	1.1±10% Ω / phase	Inductance	3.0±20% mH / Phase
Holding Torque	≥ 12.6 Kg-cm(175 Oz.in)	Detent Torque	400 g-cm REF.
Rotate Direction	ABCDA CW	MAX Starting PPS	2500 PPS
MAX slewing PPS	5000 PPS	Insulation Resistance	≥100 MΩ(DC 500V)
HI POT	AC600V/1mA/1S	Insulation Class	Class B
Rotor Inertia	300 g.cm²	Weight	0.72 Kg REF.

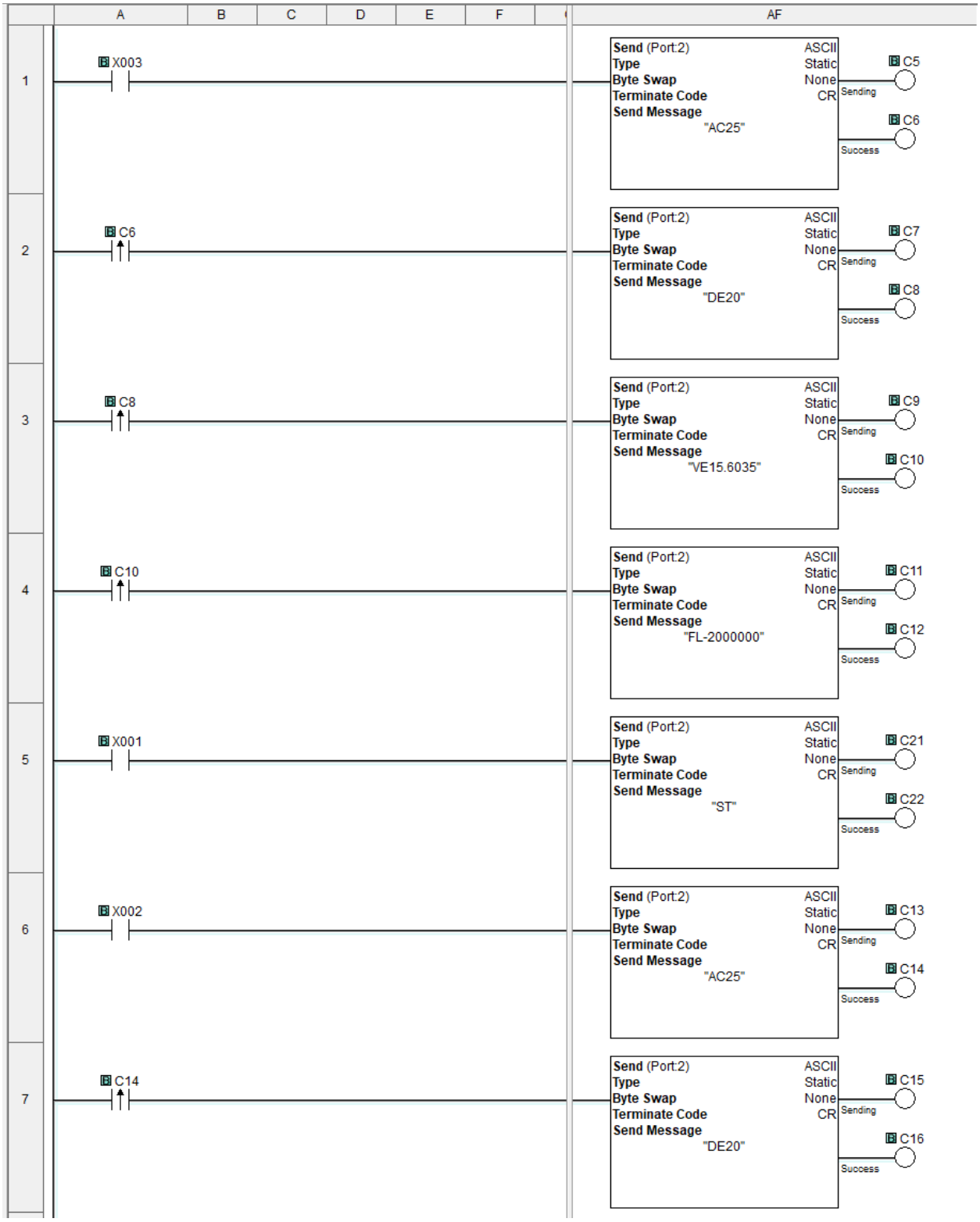
TORQUE CURVES



Wiring Diagram



APPENDIX B


A
B
C
D
E
F
AF
1
X003
Send (Port:2)
Type
Byte Swap
Terminate Code
Send Message
"AC25"
ASCII
Static
None
CR
C5
Sending
C6
Success
2
C6
Send (Port:2)
Type
Byte Swap
Terminate Code
Send Message
"DE20"
ASCII
Static
None
CR
C7
Sending
C8
Success
3
C8
Send (Port:2)
Type
Byte Swap
Terminate Code
Send Message
"VE15.6035"
ASCII
Static
None
CR
C9
Sending
C10
Success
4
C10
Send (Port:2)
Type
Byte Swap
Terminate Code
Send Message
"FL-2000000"
ASCII
Static
None
CR
C11
Sending
C12
Success
5
X001
Send (Port:2)
Type
Byte Swap
Terminate Code
Send Message
"ST"
ASCII
Static
None
CR
C21
Sending
C22
Success
6
X002
Send (Port:2)
Type
Byte Swap
Terminate Code
Send Message
"AC25"
ASCII
Static
None
CR
C13
Sending
C14
Success
7
C14
Send (Port:2)
Type
Byte Swap
Terminate Code
Send Message
"DE20"
ASCII
Static
None
CR
C15
Sending
C16
Success

APPENDIX C

...epos\Sensor_data_coverter\Sensor_data_coverter\Form1.cs 1

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10 using System.IO;
11 using System.Threading;
12 using System.Diagnostics;
13
14 namespace Sensor_data_coverter
15 {
16     public partial class Form1 : Form
17     {
18         /* CODE made by Grzegorz Machura (Grzegorz2121, Poland, Dolnyśląsk, I
             LO w Jaworze) */
19
20         /* public class point
21          {
22             public double X;
23             public double Y;
24             public string Z;
25          }*/
26
27
28         //List<point> points_list = new List<point>();
29
30         int left_offset = 0;
31         int top_offset = 0;
32
33         const float limit = -90;
34
35         double m_factor;
36
37         string input_path;
38         Stopwatch stopwatch = new Stopwatch();
39
40         public Form1()
41         {
42             InitializeComponent();
43         }
44
45
46         private void Pick_file_Click(object sender, EventArgs e)
47         {
48             openFileDialog1.ShowDialog();
49         }
50
51         private void openFileDialog1_FileOk(object sender, CancelEventArgs e)
52         {
53             input_path = openFileDialog1.FileName;
54         }
55
```

```
56          private void Pick_output_Click(object sender, EventArgs e)
57          {
58              saveFileDialog1.ShowDialog();
59          }
60
61
62          public double find_lowest_value(string path)
63          {
64              StreamReader sr = new StreamReader(path);
65              double lowest=0;
66
67              int X_coord = 0;
68              int Y_coord = 0;
69              string input_line = "";
70
71              for (int i = 0; i < top_offset; i++)
72              {
73                  sr.ReadLine();
74              }
75
76              while ((input_line = sr.ReadLine()) != null)
77              {
78
79                  input_line = input_line.Replace(" ", string.Empty);
80
81                  string[] points = input_line.Split(',');
82
83                  for (int i = left_offset; i < points.Length; i++)
84                  {
85                      if (Convert.ToDouble(points[i].Replace(".", ",")) < limit)
86                      {
87
88                      }
89                      else
90                      {
91                        double temp = Convert.ToDouble(points[i].Replace(".",
                      ","));
92                          if(temp<lowest)
93                          {
94                              lowest = temp;
95                          }
96                      }
97
98                      X_coord++;
99                  }
100
101                 X_coord = 0;
102                 Y_coord++;
103
104             }
105
106             return lowest;
107         }
108
109         private void saveFileDialog1_FileOk(object sender, CancelEventArgs e)
110         {
```

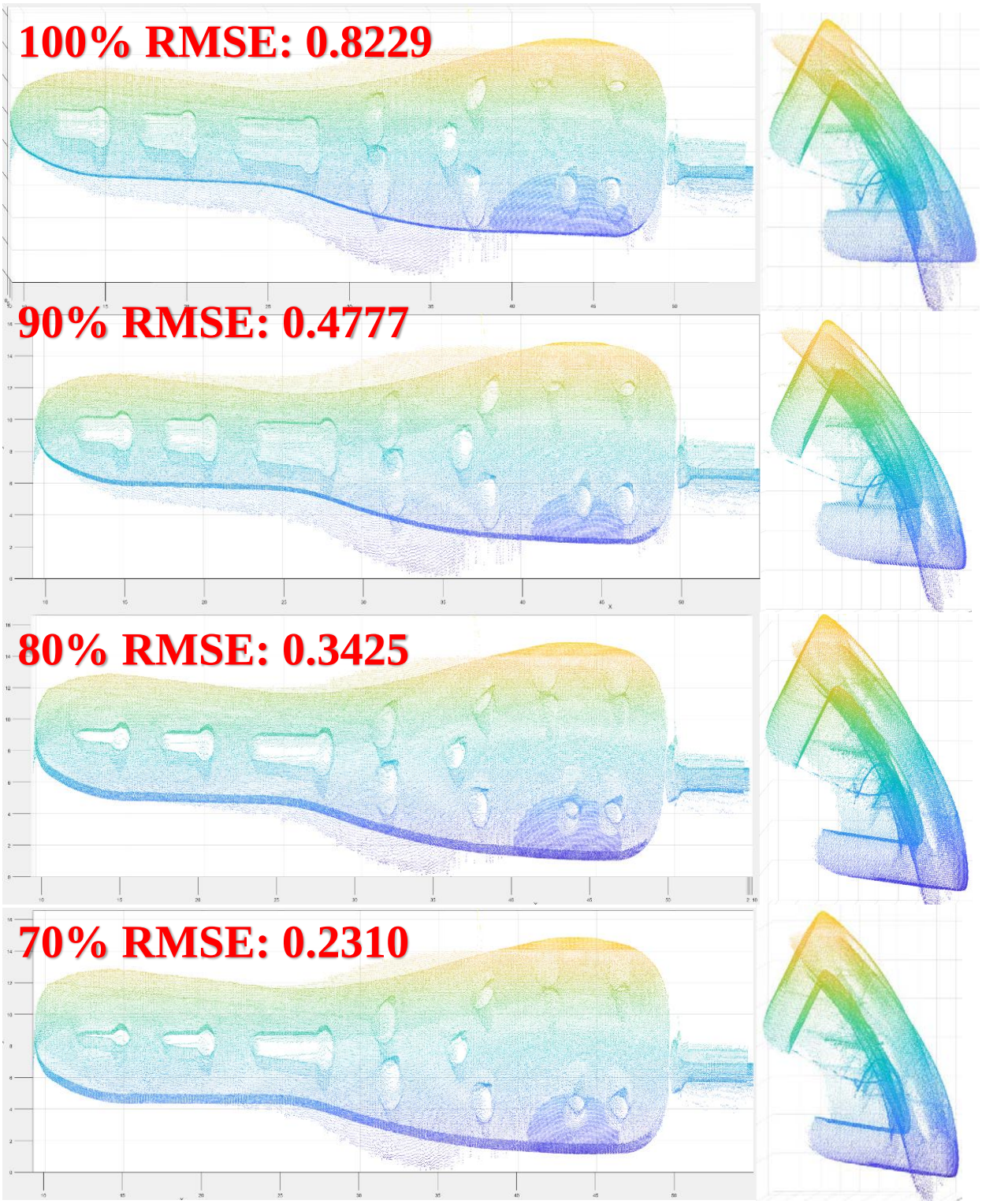
...epos\Sensor_data_coverter\Sensor_data_coverter\Form1.cs 3

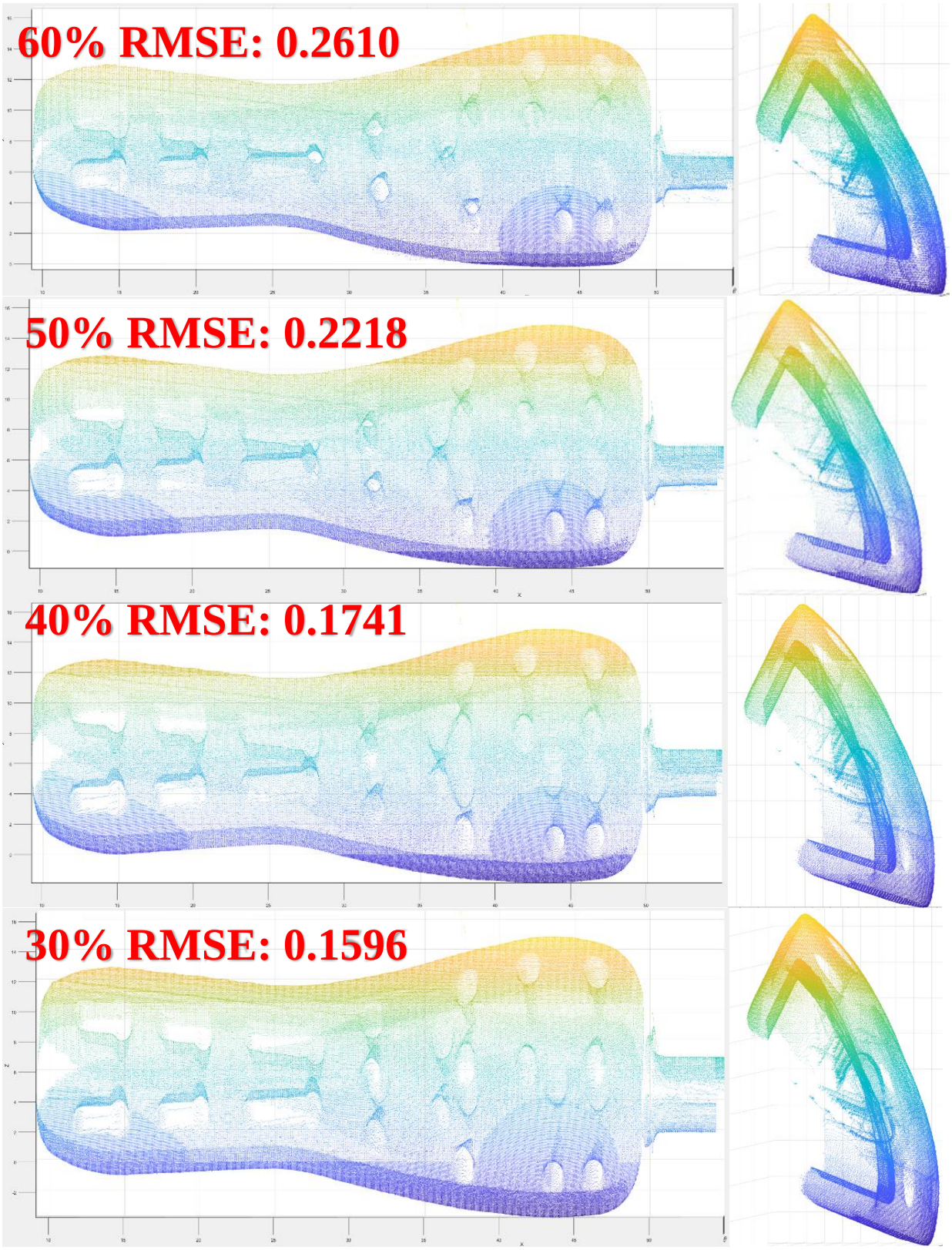
```
111             ticks_label.Text = "00:00";
112             miliseconds_label.Text = "00:00";
113             seconds_label.Text = "00:00";
114             minutes_label.Text = "00:00";
115             stopwatch.Start();
116
117             string output_path = saveFileDialog1.FileName;
118
119             StreamReader stream_reader = new StreamReader(input_path);
120             StreamWriter stream_writer = new StreamWriter(output_path);
121
122             int X_coord=0;
123             int Y_coord=0;
124             string input_line = "";
125
126             double lowest = find_lowest_value(input_path);
127
128             for(int i = 0 ; i<top_offset ; i++ )
129             {
130                 stream_reader.ReadLine();
131             }
132
133             while ((input_line = stream_reader.ReadLine())!=null)
134             {
135
136                  input_line = input_line.Replace(" ", string.Empty);
137
138                  string[] points = input_line.Split(',');
139
140                  for(int i =  left_offset; i < points.Length ; i++)
141                  {
142                     if(Convert.ToDouble(points[i].Replace(".", ","))<limit)
143                     {
144
145                     }
146                     else
147                     {
148                        /* point p = new point();
149                         p.X = Convert.ToDouble(X_coord * m_factor);
150                         p.Y = Convert.ToDouble(Y_coord * m_factor);
151                         p.Z = points[i];
152                         points_list.Add(p);*/
153                         stream_writer.WriteLine((X_coord * m_factor).ToString
                    ().Replace(",",".") + "," + (Y_coord * m_factor).ToString
                    ().Replace(",", ".") + "," + ((Convert.ToDouble(points
                    [i].Replace(".", ","))-lowest).ToString().Replace
                    (",","."));
154                     }
155
156                     X_coord++;
157                  }
158
159                 X_coord = 0;
160                 Y_coord++;
161
162             }
```

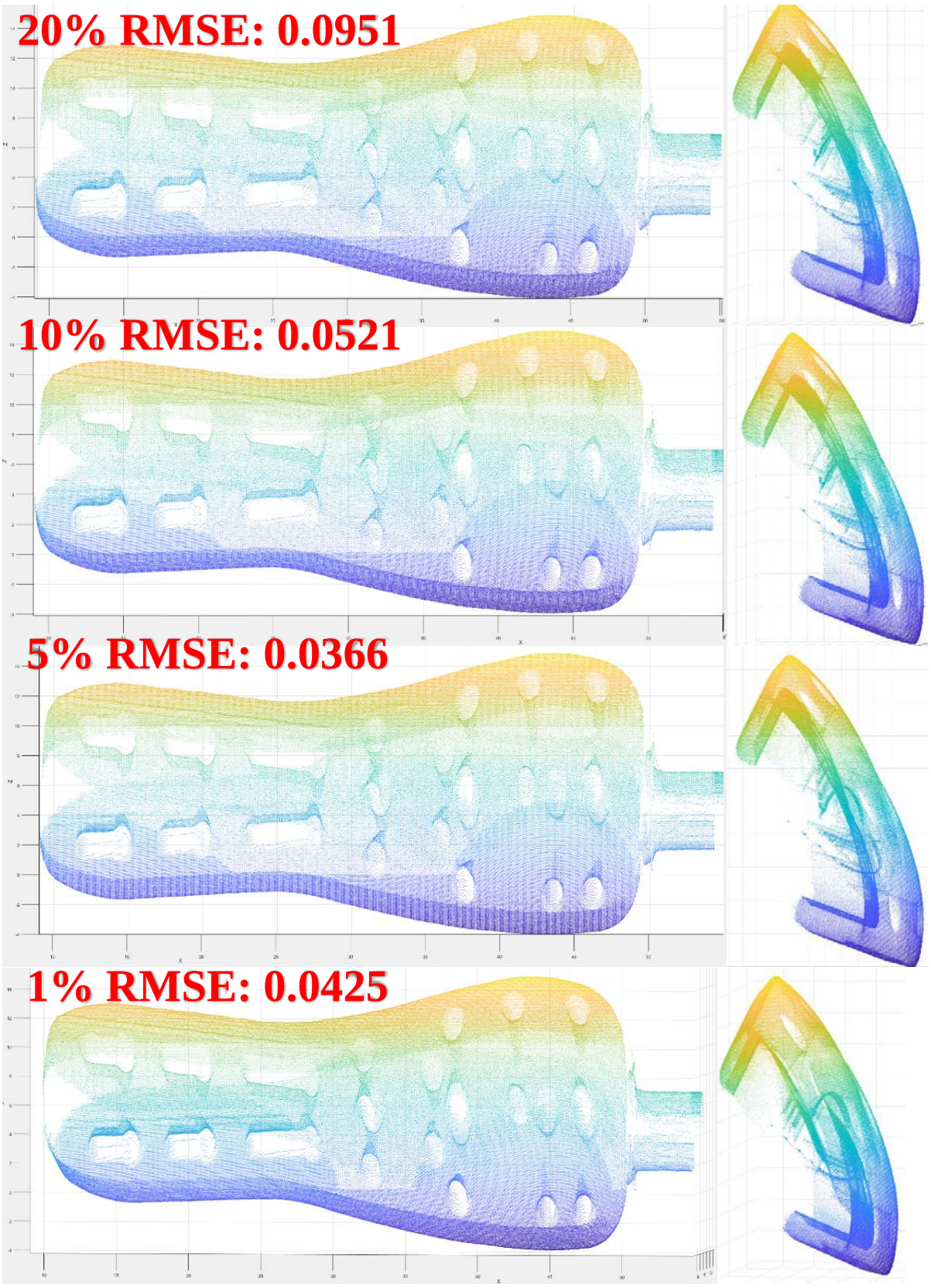
```
163
164
165             input_path = "";
166             output_path = "";
167             X_coord=0;
168             Y_coord=0;
169
170             stream_writer.Flush();
171
172             stream_reader.Close();
173             stream_writer.Close();
174
175             stopwatch.Stop();
176             ticks_label.Text = stopwatch.Elapsed.Ticks.ToString();
177             miliseconds_label.Text = stopwatch.Elapsed.Milliseconds.ToString
                  ();
178             seconds_label.Text = stopwatch.Elapsed.Seconds.ToString();
179             minutes_label.Text = stopwatch.Elapsed.Minutes.ToString();
180
181         }
182
183         private void Form1_Load(object sender, EventArgs e)
184         {
185
186         }
187
188         private void change_offsets_button_Click(object sender, EventArgs e)
189         {
190             try
191             {
192                 top_offset = Convert.ToInt16(top_offset_box.Text);
193             }
194             catch
195             {
196                 top_offset = 0;
197             }
198
199             try
200             {
201                 left_offset = Convert.ToInt16(left_offset_box.Text);
202             }
203             catch
204             {
205                 left_offset = 0;
206             }
207
208             try
209             {
210                 m_factor = Convert.ToDouble(multiplication_box.Text.Replace
                      (".", ","));
211             }
212             catch
213             {
214                 m_factor = 1;
215             }
216
```

```
217         }
218
219         private void top_offset_box_KeyPress(object sender, KeyPressEventArgs
              e)
220         {
221             if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
222             {
223                 e.Handled = true;
224             }
225         }
226
227         private void left_offset_box_KeyPress(object sender, KeyPressEventArgs
              e)
228         {
229             if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
230             {
231                 e.Handled = true;
232             }
233         }
234
235         private void multiplication_box_KeyPress(object sender,
              KeyPressEventArgs e)
236         {
237             if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && !
                  char.IsPunctuation(e.KeyChar))
238             {
239                 e.Handled = true;
240             }
241         }
242         /* CODE made by Grzegorz Machura (Grzegorz2121, Poland, Dolnyśląsk, I
              LO w Jaworze) */
243     }
244 }
245
```

APPENDIX D




60% RMSE: 0.2610
50% RMSE: 0.2218
40% RMSE: 0.1741
30% RMSE: 0.1596


20% RMSE: 0.0951
10% RMSE: 0.0521
5% RMSE: 0.0366
1% RMSE: 0.0425

APPENDIX E

4/1/18 9:45 PM C:\Users\Nichol...\HighEffiencyCSVtoPcCSV.m 1 of 1

```
tic
M = csvread('Run1.csv',17,20);
toc
mn = min(min(M));
Mn = abs(mn);
M = M + Mn;
toc
M(M < -90) = NaN;
x = 0.005; y = 0.005;
[rows,cols] = find(~isnan(M));
NA = [rows*x, cols*y, M(~isnan(M))];
toc
csvwrite('SortRun1.csv',NA);
toc

tic
M = csvread('Run2.csv',17,20);
toc
mn = min(min(M));
Mn = abs(mn);
M = M + Mn;
toc
M(M < -90) = NaN;
x = 0.005; y = 0.005;
[rows,cols] = find(~isnan(M));
NA = [rows*x, cols*y, M(~isnan(M))];
toc
csvwrite('SortRun2.csv',NA);
toc

tic
M = csvread('Run3.csv',17,20);
toc
mn = min(min(M));
Mn = abs(mn);
M = M + Mn;
toc
M(M < -90) = NaN;
x = 0.005; y = 0.005;
[rows,cols] = find(~isnan(M));
NA = [rows*x, cols*y, M(~isnan(M))];
toc
csvwrite('SortRun3.csv',NA);
toc
```

```
%Load Crop and Reformat data from sensor 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
M = csvread('Run2.csv',17,20);
M(M < -90) = NaN;
mn = min(min(M));
Mn = abs(mn);
M = M + Mn;
x = 0.05; y = 0.05; z = 1;
[rows,cols] = find(~isnan(M));
NA = [rows*x, cols*y, M(~isnan(M))*z];
NB = [rows*x, cols*y, M(~isnan(M))*z];
B = NA;
%Duplicate output and Crop for post prosessing %%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m,~] = size(NB);
NB(NB([1:m],1) > 48) = NaN;
NB(NB([1:m],1) < 35) = NaN;
NB(NB([1:m],2) > 7) = NaN;
NB(NB([1:m],3) < 4) = NaN;
NB(1,1) = 9.3;
NB(1,2) = 0.65;
NB(1,3) = 0;

subplot(2,2,1);
pcshow(NB);
Crop2 = NB;

%csvwrite('SortRun2.csv',NA);
tic
%Load Crop and Reformat data from sensor 3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = csvread('Run3.csv',17,20);
M(M < -90) = NaN;
mn = min(min(M));
Mn = abs(mn);
M = M + Mn;

x = 0.05; y = 0.05;
[rows,cols] = find(~isnan(M));
NA = [rows*x, cols*y, M(~isnan(M))];
NB = [rows*x, cols*y, M(~isnan(M))];
C = NA;
%Duplicate output and Crop for post prosessing %%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m,~] = size(NB);
NB(NB([1:m],1) < 38) = NaN;
NB(NB([1:m],1) > 50) = NaN;
NB(NB([1:m],2) < 5) = NaN;
NB(NB([1:m],3) < 2) = NaN;
NB(1,1) = 12.05;
NB(1,2) = 2.15;
NB(1,3) = 0;
subplot(2,2,2);
pcshow(NB);
```

```
Crop3 = NB;
%Convert Matrixs to Pointcloud identity %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B = pointCloud(B)
C = pointCloud(C)
CC = pointCloud(Crop2)
BB = pointCloud(Crop3)
%subplot(2,2,3);
pcshow(Crop2);
%subplot(2,2,4);
pcshow(Crop3);



%Get Transfomation Matrix

%gridSize = 0.1;
%BB = pcdownsample(Crop2, 'gridAverage', gridSize);
%CC = pcdownsample(Crop3, 'gridAverage', gridSize);
%a = pi;
%R = [1 0 0 0; 0 cos(a) -sin(a) 0; 0 sin(a) cos(a) 0; 0 0 0 1];
%RR = affine3d(R);
a = (pi);
R = [1 0 0 0; 0 cos(a) -sin(a) 0; 0 sin(a) cos(a) 0; 0 0 0 1];
RR = affine3d(R);
[tform,movingReg,rmse] = pcregrigid(CC, BB, 'Metric','pointToPoint','Extrapolate',↙
false,'MaxIterations',250000000,'InitialTransform', RR );
CT = pctransform(CC,tform);
mergeSize = 0.1;
FINAL1 = pcmerge(BB,CT,mergeSize);
subplot(2,2,3);
pcshow(FINAL1);
rmse
BT = pctransform(B,tform);
mergeSize = 0.1;
FINAL2 = pcmerge(C,BT,mergeSize);
subplot(2,2,4);
pcshow(FINAL2)
toc
```

```
%Load Crop and Reformat data from sensor 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
M = csvread('Run2.csv',17,20);
M(M < -90) = NaN;
mn = min(min(M));
Mn = abs(mn);
M = M + Mn;
x = 0.05; y = 0.05; z = 1;
[rows,cols] = find(~isnan(M));
NA = [rows*x, cols*y, M(~isnan(M))*z];
NB = [rows*x, cols*y, M(~isnan(M))*z];
B = NA;
%Duplicate output and Crop for post prosessing %%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m,~] = size(NB);
NB(NB([1:m],1) > 48) = NaN;
%NB(NB([1:m],1) < 35) = NaN;
%NB(NB([1:m],2) > 7) = NaN;
%NB(NB([1:m],3) < 4) = NaN;
%NB(1,1) = 9.3;
%NB(1,2) = 0.65;
%NB(1,3) = 0;

subplot(2,2,1);
pcshow(NB);
Crop2 = NB;

%csvwrite('SortRun2.csv',NA);
tic
%Load Crop and Reformat data from sensor 3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = csvread('Run3.csv',17,20);
M(M < -90) = NaN;
mn = min(min(M));
Mn = abs(mn);
M = M + Mn;

x = 0.05; y = 0.05;
[rows,cols] = find(~isnan(M));
NA = [rows*x, cols*y, M(~isnan(M))];
NB = [rows*x, cols*y, M(~isnan(M))];
C = NA;
%Duplicate output and Crop for post prosessing %%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m,~] = size(NB);
%NB(NB([1:m],1) < 38) = NaN;
NB(NB([1:m],1) > 50) = NaN;
%NB(NB([1:m],2) < 5) = NaN;
%NB(NB([1:m],3) < 2) = NaN;
%NB(1,1) = 12.05;
%NB(1,2) = 2.15;
%NB(1,3) = 0;
subplot(2,2,2);
pcshow(NB);
```

```
Crop3 = NB;
%Convert Matrixs to Pointcloud identity %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B = pointCloud(B)
C = pointCloud(C)
CC = pointCloud(Crop2)
BB = pointCloud(Crop3)
%subplot(2,2,3);
pcshow(Crop2);
%subplot(2,2,4);
pcshow(Crop3);



%Get Transfomation Matrix

%gridSize = 0.1;
%BB = pcdownsample(Crop2, 'gridAverage', gridSize);
%CC = pcdownsample(Crop3, 'gridAverage', gridSize);
%a = pi;
%R = [1 0 0 0; 0 cos(a) -sin(a) 0; 0 sin(a) cos(a) 0; 0 0 0 1];
%RR = affine3d(R);
a = (pi);
R = [1 0 0 0; 0 cos(a) -sin(a) 0; 0 sin(a) cos(a) 0; 0 0 0 1];
RR = affine3d(R);
[tform,movingReg,rmse] = pcregrigid(CC, BB,'InitialTransform',↙
RR,'Metric','pointToPoint','Extrapolate',false,'InlierRatio',.6,'MaxIterations',1000000)
CT = pctransform(CC,tform);
mergeSize = 0.1;
FINAL1 = pcmerge(BB,CT,mergeSize);
subplot(2,2,3);
pcshow(FINAL1);
rmse
BT = pctransform(B,tform);
mergeSize = 0.1;
FINAL2 = pcmerge(C,BT,mergeSize);
subplot(2,2,4);
pcshow(FINAL2)
toc
```