Exploring the Performance of Low-Cost Part Scanning in Wire Arc Additive Manufacturing

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Anukkah Burleson

December 2025

Acknowledgments

I am deeply grateful to all the friends, family, colleagues, and mentors who have supported me throughout my educational career and this research journey. Your encouragement, guidance, and belief in me have made even the most challenging moments feel possible. I want to thank my graduate research advisor, Dr. Bradley Jared, for his guidance through this project and my graduate studies. I am also thankful to my research group for their collaboration, advice, and constant willingness to help when needed. I especially want to thank Tommy Tucker, Badri Narayanan, and the rest of the team at Lincoln Electric for their invaluable support, feedback, and generosity throughout this work.

Abstract

Wire arc additive manufacturing (WAAM) is a metal additive process which utilizes traditional welding equipment in conjunction with robotic manipulators to create metal parts for a variety of applications. The WAAM process requires material deposition which involves rapid material melting and solidification. Heating and cooling cycles are extensive and produce internal material stresses in turn creating distortion. Distortion can become extensive and commonly produces nonconforming parts, reducing yield, and increasing manufacturing time and cost. To address these challenges, 3D scanning can be employed as an in-situ monitoring technique to detect distortion as it develops by gathering accurate scans in-between layers. Surface measurements using 3D scanning are desired with uncertainties below 1 mm. Existing scanning solutions already exist, but are expensive and difficult to co-locate in welding cells where excessive operating temperatures, weld spatter, and airborne soot often occur. Therefore, a low-cost solution is necessary. Combining an inexpensive depth sensor, a robotic manipulator already utilized for WAAM processes and data processing scripts creates a promising solution for low-cost 3D scanning. Open3D integration is utilized alongside aligned image frames with robot position data. Performance experiments were conducted to access the system's capability and limitations using test artifacts of different shapes and sizes. Robot scan velocity was varied to test effects on surface reconstruction. Experiments scanning test artifacts shows the low-cost system is capable of producing meshes accurate to 1 mm. Test artifacts also demonstrated limitations of the system when evaluating parts with low-texture and sharp edges being prone to poor scan quality. Experiments were conducted during WAAM processes and demonstrate the ability of the system to be implemented in-situ to monitor part geometry.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Wire Arc Additive Manufacturing

Wire arc additive manufacturing (WAAM) is a metal additive process that builds components layer by layer through the controlled deposition of molten wire [4]. Once a part is fabricated it can be post-processed using conventional machining techniques to achieved the desired final geometry and surface finish. WAAM processes most commonly leverage traditional welding equipment along with a robotic manipulator enabling the production of complex and larger scale metal parts with reduced material waste. Material deposition involves rapid melting and solidification of the material resulting in repeated thermal cycling. Internal stresses within the part occur due to the thermal cycling. These stress often lead to geometric distortion of the part, a common challenge within WAAM processes. For large components, deposition can extend over several days and numerous print layers. As a result, part distortion can accumulate across layers, potential leading to non-confirming parts, increasing cycle times and costs. Distortion is considered a core problem in welding-based additive manufacturing [5]. An example of an aluminum WAAM part with a subsequently distorted base plate can be seen in Figure 1.1.

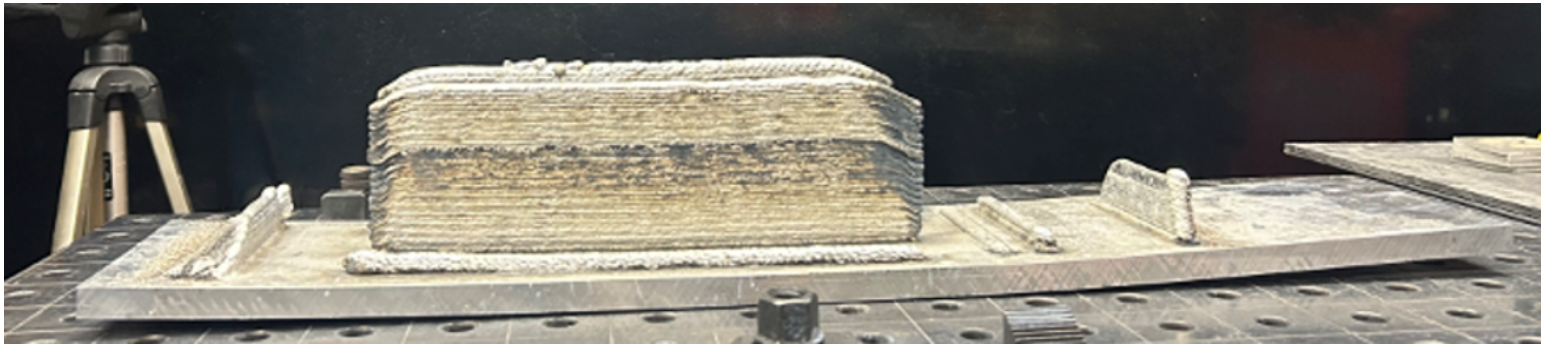


Figure 1.1: Example of distortion for a WAAM part.

Monitoring WAAM processes in-situ creates several challenges, primarily due to the harsh operating environment. WAAM cells expose equipment to elevated temperatures, spatter soot, and other airborne debris generated during deposition. Such a dynamic environment pose significant risks to sensitive or high-value sensors. Contamination, interference or physical damage can occur to monitoring equipment. Significant heat created from the arc during deposition can also degrade or warp monitoring equipment. Equipment housings and electronic components are all subject to damage from the thermal load. These combined factors make it difficult to maintain accuracy and longevity of sensitive equipment creating a strong motivation for a low-cost solution.

1.2 3D Scanning

3D scanning is a process of measuring real-world geometries to collect a three-dimensional representation of an object's shape and appearance. The resulting scan data is used to generate three-dimensional models, typically represented by either polygon meshes or point clouds. A single scan produces a single surface point cloud, which rarely captures a complete representation of the entire part geometry. Therefore multiple scan from many different viewpoints are required, creating multiple point clouds. Individual point clouds are transformed into a common coordinate system through a process known as registration or alignment. The sequence of steps from a raw data set to a complete model is know as the 3D scanning pipeline. There are several types of available scanning technology including laser-based scanners, structured light scanners, photogrammetry systems, and structured depth cameras [6]. Laser based scanners determine surface geometry by time-of-flight or triangulation of reflected laser beams. Structured light scanners project a known light pattern on the observed surface. Deformation of the pattern is observed by another camera to determine surface slopes and depth. Photogrammetry systems create a 3D image through the stitching of 2D images based on feature matching algorithms. Structured depth cameras combine color imaging with depth sensing, using either infrared patterns or stereo-vision methods to capture both red-green-blue (RGB) color and depth data simultaneously. A stereo vision structured depth camera was selected for the work herein. 3D scanners can be

used to monitor part geometry during WAAM processes by creating successive 3D meshes of the part layers are deposited. Scanning between layers, such as during process dwell times, allows for monitoring without significantly interrupting the process cycle time. However, traditional 3D scanning systems are expensive and often rely on external fiducial markers for alignment for image stitching. Both the high cost and the dependence on physical markers presents challenges in harsh WAAM manufacturing environments. Limitations of traditional scanning technology motivates a low-cost, marker less scanning systems capable of providing a sufficient accuracy for distortion monitoring

1.3 Current Objectives

The aim of this research was to develop a low-cost 3D scanning system capable of morphology measurement of WAAM parts. By utilizing readily available hardware, the system remains cost effective and creates an almost disposable solution for harsh manufacturing environments. Previous research was focused on creating a working system, including scanning functionality and mesh generation. The work described herein contributed to this previous work in the development of spatial and time synchronization.

After previous research created successful mesh generation, the next steps were determining the accuracy, repeatability, and limitations. The primary goals were to validate the integration of the scanning system into WAAM processes and to quantify the accuracy of the generated surface meshes. An accuracy target of 1 mm was established to demonstrates the system usefulness for distortion detection. Additional parameters such as scanning velocity, part geometry, and surface finish were also evaluated to better understand their influence on scan quality and reconstruction accuracy.

Chapter 2

Previous Work

As WAAM becomes increasingly accepted within industry as a viable manufacturing technique, ongoing research aims to better understand the causes, prevention, and detection of part distortion. Some research efforts are focused on predicting distortion before it occurs by creating models to simulate thermomechanical-related defects. A finite element analysis (FEA) can be useful in detect models and conditions where distortion is likely to occur [7]. While useful, FEAs are computationally expensive and can take several hours to process. An artificial neural networks (ANN) can be employed to predict welding distortion and geometric accuracy for multilayer WAAM structures. ANNs can be paired with FEAs to improve limitations caused by computational expense [8]. Along with predictive modeling, researchers have also explored methods for accessing geometric accuray post process in completed WAAM components. One study established a repeatable method to analyze the geometric accuracy of WAAM parts, more specifically multi-layer circular walls produced using Cold Metal Transfer (CMT) [9]. Specimens were manufactured with varying parameters and 3D meshes were generated using a high-resolution GOM ATOS Core 200 Structured light-scanner. The resulting meshes were analyzed in Rhinoceros 3D and Grasshopper software to extract metrics including average bead height and width, lateral waviness, and a torch on/off region profile. While the study demonstrated the usefulness of mesh generation to create a comprehensive description of the WAAM processes, the process was performed post-process using a stationary GOM ATOS scanner, limiting its applicability for real-time process correction. To address these limitations, researchers are exploring in-situ

monitoring approaches as well. One study developed a closed-loop surface correction system for WAAM . The method scans each layer with a laser topography sensor and processes the data via a depth-image algorithm to automatically perform milling or re-deposition to maintain uniform geometry [10]. The study demonstrates a crucial step towards self-correcting additive manufacturing, but the laser sensor is costly and only suitable for small test coupons, not large WAAM structures. While these studies successfully demonstrate the effectiveness of 3D monitoring during WAAM process, several challenges still exist. Another study examined using digitial image correlation in-situ for WAAM processes. DIC provides a contactless measurement method useful for in-situ monitoring of WAAM processes [11]. However, it requires applying a speckle pattern on the part creating an interruptive step in the additive manufacturing process. Given the growing emphasis on in-situ sensing and real-time geometric evaluation, an understanding of the accuracy and limitations of 3D scanning systems becomes critical. Research surrounding the accuracy of 3D scanners has been conducted before. One study reviewed accuracy of four different scanner ranging from low to high cost [3]. The scanners were evaluated using calibrated artifacts and confirming to current ISO and VDI/VDE standards for optical coordinate meteorology (OCM). Inspiration for the work conducted here was drawn from the experimental methods presented in the referenced study, particularly regarding the evaluation of 3D scanning accuracy. One major challenge implementing 3D scanning as a in-situ monitoring technique is the environment. As previously stated, the environment is hazardous for monitoring equipment. The use of low-cost and readily available equipment allows for a affordable and nearly disposable solution enabling use in harsh WAAM environments without significant financial cost. Previous research conducted by Koller was key in creating the current working low-cost scanning system [1]. The groundwork for the 3D scanning pipeline and communication between the robot and camera were generated. The work described here contributed to Koller's work in the development of spatial and time synchronization. Prior research on 3D scanning system's accuracy, scanning integration into WAAM processes, and low-cost scanning solutions provided the foundation for the approach adopted here.

Chapter 3

System Overview

3.1 Hardware

To create the low-cost 3D scanning system, an inexpensive depth sensor is paired with a robotic manipulator. The robotic manipulator is already utilized for WAAM deposition, which makes the addition of 3D scanning relatively inexpensive since no additional robotic hardware is necessary. Pairing the robot and the depth sensor enables coordinated operation as the robot provides real-time positional data while the sensor captures 3D geometry of the part surface. Together, these components, along with a camera flash whose use will be explained later, create a low-cost 3D scanning system. Figure 3.1 presents a labeled diagram of the complete system setup.

3.1.1 Robotic Manipulator

The robotic manipulator used for these experiments is a KUKA KR-6. The KR-6 is a 6-axis robotic manipulator with a 6 kg payload capacity and a 1.6 m maximum reach. The robot is accompanied by a KUKA KR C2 controller which provides trajectory control and digital I/O interfaces. The controller houses a Beckhoff EL2809 EtherCAT digital terminal which provides 16 channels of 24 V DC outputs rated at 0.5 A. A welding torch is attached to the end effector of the robot for the purpose of WAAM fabrication. For these experiments, an

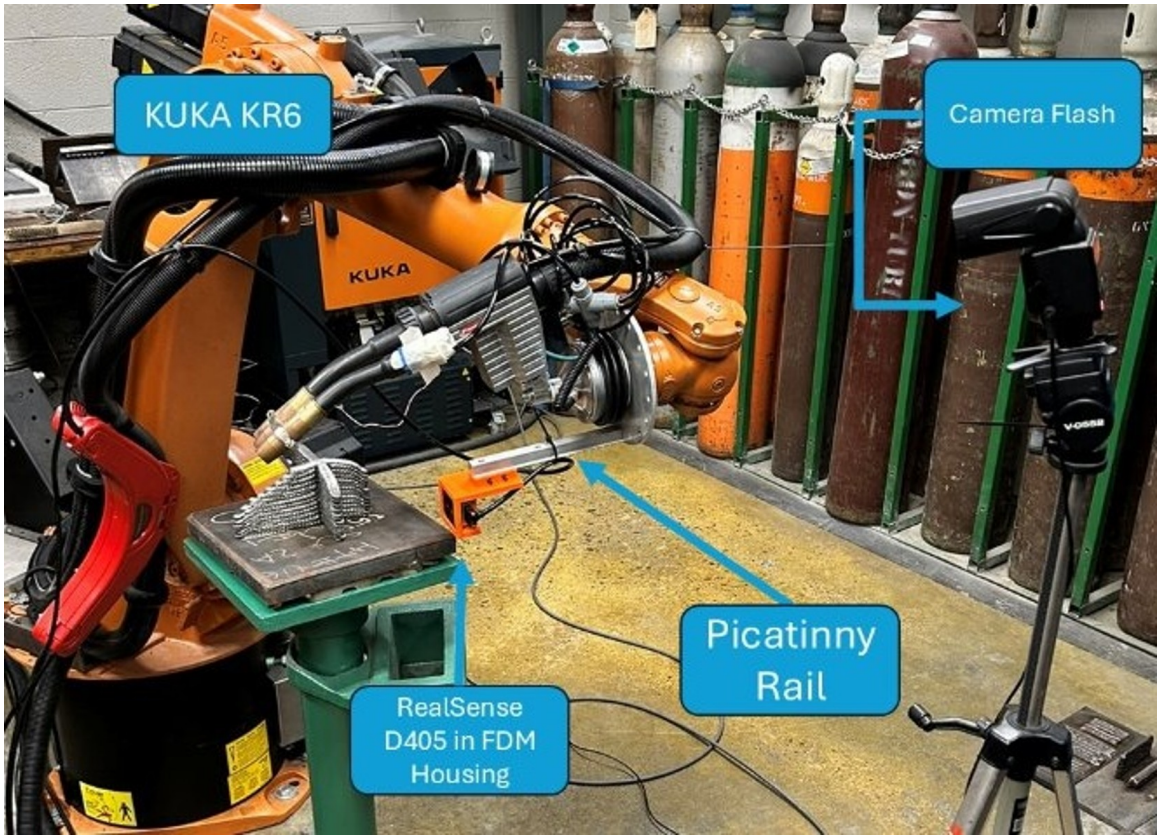
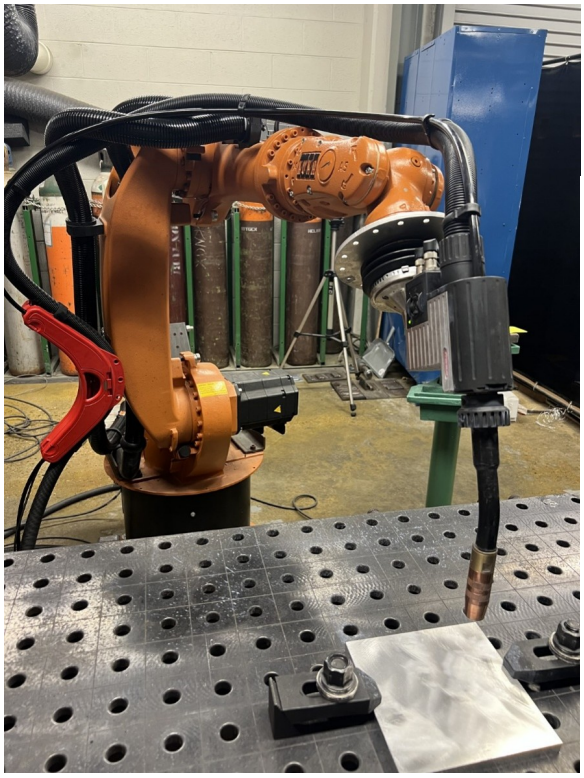


Figure 3.1: Robotic scanning hardware set-up [1].

Intel RealSense D405 was also attached to the robot end effector to allow the camera to be maneuvered in space around the object being scanned.

KUKA WorkVisual is used to configure one of the EL2809 output terminals for the purpose of triggering the camera flash. KUKA's Robot Sensor Interface (RSI) was utilized to stream and then record position data from the robot during measurements based on the defined tool and base coordinate frames. RSI was also used to trigger when the camera starts and stops recording. Data from RSI is sent as .xml packets to a host computer at a rate of 250 Hz and recorded using a Python script. See Section 3.2.2 for details. Figure 3.2 provides images of the KUKA KR-6 and the corresponding controller located in the WAAM development cell.



(a) KUKA KR-6 Manipulator

(b) KUKA KR C2 Controller

Figure 3.2: KUKA manipulator and controller in the WAAM development cell.

3.1.2 Intel RealSense D405

An Intel RealSense D405 camera is utlized as a depth sensor attached to the end effector of the robotic manipulator. The D405 uses two separate cameras to provide stereo vision depth measurements. Stereo vision calculates depth by using information from two separate camera images captured from slightly different angles at the exact same instant in time [12]. Figure 3.3 provides a demonstrates of the concept of stereo vision. Two camera's view the object from different viewpoints. The difference in the perceived position from each frame is used to determine the object's depth. The approach mimics human binocular vision, where depth perception is derived from the difference in images between the left and right eye. The camera operates within a working range of 7 – 50 cm and provides both color and depth images. Figure 3.4 shows an example color and depth image from the D405. Multiple resolutions are provided including 1280 x 720, 848 x 480, 640 x 480, 640 x 360, 480 x 270, and 424 x 240 pixels. Depth and color frames can be recorded along with time stamps at frame rates up to 90 frame per second (FPS). For experimentation, a rate of 30 FPS and resolution of 1280 x 720 pixels was used based on suggestions from Intel to produce the most stable depth values [13].

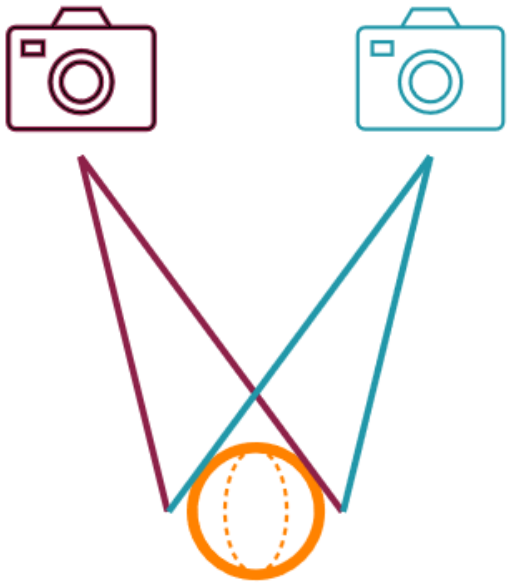


Figure 3.3: Concept of depth estimation using stereo vision.

(a) Color Image **(b)** Depth Image

Figure 3.4: Example color and depth image from D405.

The D405 is a small and compact sensor with a size of 42 x 42 x 23 mm and a weight of only 60 grams [14]. It's aluminum housing features two M3 mounting points on the back and one $\frac{1}{4}$ - 20 UNC thread mount on the bottom. The M3 mounting points were utilized to mount the camera inside a fused deposition model (FDM) mounting bracket specifically designed for the D405. Figure 3.6 provides an image of the D405 in the FDM mount on the robot. A USB 3.1 micro-b port was used to transfer frame data from the camera to a dedicated data acquisition (DAQ) computer. The D405 costs around 275 USD making it an inexpensive depth sensor.

3.1.3 Camera Flash

A camera flash was introduced to the measurement system to facilitate temporal alignment of camera frames with the robot position data. The flash used was a Neewer NW-670 which was triggered via its internal PC sync circuit. The PC sync circuit activates through the shorting of the internal circuit within the Neewer NW-670 flash through a sync jack located on the side of the flash. The trigger signal was generated by the KUKA KR C2 controller and then relayed to the flash through an output channel on the Beckhoff EL2809. To provide protection for the controller and to trigger the flash with limited latency, a simple external circuit consisting of a 2200 ohm, 0.5 watt resistor, LTV-816 optocoupler was constructed to short the internal PC sync circuit. The external circuit uses the 24VDC 0.5 A output from the KR C2 controller to trigger the camera flash while effectively isolating the controller from any high voltage in the internal PC sync circuit. The cost of the camera flash was 100

USD, therefore, adding minimal expense to the system. While the flash unit allows a range of output light intensities and pulse durations, for this application, it was simply set to full output and aimed directly at the area being scanned.

3.2 Software

3.2.1 Kuka Robot Language

KUKA Robot Language (KRL) is a proprietary programming language developed by KUKA for controlling their industrial robot systems. KRL is syntactically like Pascal and designed specifically for defining motion and logic control. A KRL code consists of two different files with the same name. The two different files are a permanent data file with extension .dat and a movement command file with the extension .src. The data file stores data associated with the program such as robot position, tool and base definitions, and variable declarations. Source files contain the main program logic which is executed by the robot controller. KRL has the ability to perform basic mathematical calculations within programs such as addition, subtraction, multiplication, and division as well as trigonometric functions (e.g. sine, cosine, tangent). These functions are useful for tasks such as establishing motion offsets or parameterizing path plans.

For scanning, the program begins by moving the robot to the home position. Once completed, the RSI stream is initialized and camera recording is activated. A wait function provides a 1 second pause before the output channel in the Beckhoff terminal is pulsed to activate the camera flash. Once the flash is triggered, the programmed robot movement will occur. Robot movements can be programmed through many different offline programs, but for these experiments movement points were programmed manually using the controller pendant. Some offline edits were performed to edit variables and implement equations necessary to define all movement points using KUKA WorkVisual.

KRL provides several different types of robot movements, but only three main movements were used during experiments. The first type of movement was a point to point (PTP) movement which moves all robot joints simultaneously to reach the target position as quickly

as possible. PTP moves are not linear and produce discontinuous motion, reducing scan quality. Because of the nature of PTP movements, they were not used for scan paths; however, a KRL program requirement is to start and end with a PTP movement. To meet this requirement, a PTP move was added as a home position to the beginning and end of the script outside of any RSI streaming and camera recording. Therefore, no PTP movement occurs during data recording. The other two movements used during experimentation were Linear Movements (LIN) and Circular Movements (CIR). Linear Movements move the tool along a straight path in a cartesian space. A Circular Movement moves the tool along a circular arc defined by an auxiliary (center of radius) and an end point to fully the define the circular arc. Both movements were utilized for path planning for scans due to their smooth motion. Once the robot scanning motion concluded, the camera recording was stopped by switching a variable using KRL. The robot then returned to the home position again as the RSI stream ended.

3.2.2 Python Code

Python scripts were used extensively during work for the system and can be broken down into 3 main functions: recording, alignment, and surface reconstruction. Three primary Python libraries were integral to the success of the measurement system. The first, pyrealsense2, was responsible for configuring and recording data from the Intel RealSense D405 camera [15]. The second, OpenCV, was used for image file management during recording [16]. OpenCV was also used for spatial and time calibration. Finally, Open3D was used for camera data processing and integration to perform surface reconstruction of scanned objects [17].

Recording

The first python script is for recording robot and camera data during scanning. The script is responsible for initializing the D405 camera connection and configuration. The script also prepares to receive and then record robot data from RSI in a .csv file. Once the robot program begins, recording starts, and camera image frames are collected along with RSI position data. Color and depth frames are both stored along with a .json file used to record important camera settings. Color and depth frames are named with the timestamp taken from the metadata during recording and stored as .jpg and .png respectively. Camera

intrinsics are one critical parameter set since they change based on camera resolution and are unique to each different camera. To ensure the correct camera intrinsics are being used in surface reconstruction, the recording script fetches the intrinsics from the camera metadata each time the recording script is ran. The camera intrinsics are saved to a .json file to be read later by other Python scripts.

Alignment

The second python script performs data alignment. It ensures that data collected from the Intel RealSense camera and the KUKA robot are synchronized and aligned in both time and space. The alignment step is critical for an accurate surface reconstruction. The script correctly sequences the camera frames with their corresponding robot positions. Robot position data is defined as the location of the defined tool in relation to the defined base. For measurements, the tool is defined as the camera. Robot position data is given in Cartesian coordinates X, Y, Z with rotation angles A, B, C. For the KR6, an A rotation corresponds to rotation about the Z-axis. B rotation corresponds to Y-axis rotation while C rotation corresponds to X-axis rotation. The camera flash is used to find the exact camera frame where movement begins, allowing for an offset to be found between the timestamps of the robot and the camera. From here a trajectory .log file is created, giving each frame a corresponding position.

Surface Reconstruction

The third and final Python script performs surface reconstruction using synchronized RGB-D frames and robot poses. To begin, a color frame and a depth frame are combined together to produce an RGB-D image. The RGB-D images are paired with their corresponding positions which were found in the second Python script, alignment. Robot positions are inverted in the integration script to match Open3D's desired format of the coordinate system relative to the camera. From here, having found all the necessary intrinsic and extrinsic for the camera in previous Python scripts (Recording and Alignment), Open3D's reconstruction begins.

Open3D accomplishes reconstruction through the use of Truncated Signed Distance Function (TSDF) [18] integration to create a voxel block grid. A voxel block grid is a data structure used to represent 3D scenes where the data structure is globally sparse and locally dense. 3D space is coarsely divided into voxel blocks creating a grid. Voxel blocks

contain dense voxels and are organized by 3D coordinates. A voxel is a three-dimensional equivalent of a pixel, representing a volume element [19]. Several parameters can be adjusted in Open3D to control the creation of the voxel block grid within the reconstruction process. These parameters are block resolution, block count, and voxel size. Block resolution is the number of voxels along each axis within a single voxel block. Block count is the total number of allocated blocks in the entire voxel block grid. If the number of allocated blocks is small, the voxel block grid may miss important data, but if allocated blocks is too large computational energy is wasted. Finally, the voxel size determines the physical length of each voxel edge in space (measured in meters). Smaller voxel sizes result in higher resolution and finer detail, while larger voxel sizes produce lower resolution but faster processing times.

For surface reconstructions discussed herein, voxel size was set to 0.001 meters. A voxel of this size was found to produce a good resolution for scans, but not be computationally expensive. Each voxel is assigned a TSDF and weight value during integration. A TSDF value falls between -1 and 1 with 0 being defined as the zero crossing where the surface is located. A positive TSDF indicates the voxel is above the surface while a negative TSDF value indicates a voxel below the surface. A weight value represents the count or number of times a specific voxel was observed in the data set. A larger weight value indicates a higher confidence in the voxel's location. After TSDF integration, the surface corresponding to the zero crossings can be extracted. The Marching Cubes algorithm [20] is used to convert the voxel grid into a smooth triangle mesh, which is exported as a .ply file.

Previous research [1] performed the integration step using Open3D's legacy application programming interface (API). Recent research carried out integration using Open3D's newer Tensor-based API which provides built-in support for extracting TSDF and weight values directly from scan data. TSDF and weight values were pulled and stored in a .csv file to be examined further. Tensor API also allow for the adjustment of more parameters within the integration process such as block resolution and block count which was unavailable in the legacy API.

3.3 System Calibration

For the system to function properly and to achieve accurate surface recreations, precise alignment between the recorded camera frames and the corresponding robot positions must be achieved. Without this alignment, the positional data is uncorrelated with the captured images, rending robot information unusable for reconstruction.

3.3.1 Camera Calibration

While the D405 is calibrated by the manufacturer, it is important to understand its potential error sources and to periodically check its calibration accuracy. The D405 uses metrics like a health check score, Depth Root mean square (RMS) error, and Z - accuracy to evaluate the accuracy and precision of the camera. The first two listed metrics, health check score and depth RMS error, focus on the camera's ability to see objects are report back their position with low noise mostly focusing on the precision or relative error of the system. Health check score is a value indicating the extent which calibration deviates from ideal. The health check score is provided after performing during on-chip self-calibration using the RealSense View SDK [2]. On-chip self-calibration requires no visual target, like a checkerboard, but instead needs to view a dense scene. Scene density can be achieved from either a natural scene with density created by multiple objects or an artificial noise pattern as seen in Figure 3.5. An artificial noise pattern was selected herein. The noise pattern is positioned in the camera's field of view (FOV) and on-chip calibration is ran. A lower health check score is better, with any value above 0.25 requiring further calibration. Depth RMS error measured as the standard deviation of a flat plane and is the noise of the plane fit appearing as bumps on the surface. Documentation lists the D405 as having an RMS error of $\leq 1\%$ [13]. The last listed metric, Z - Accuracy, focuses on the accuracy of the depth readings. Z - accuracy compares the depth reading from the camera to a ground truth distance quantifying error along the Z axis. Accuracy in Z – axis can affect scaling and size dimensions on meshes. Intel RealSense documentation states the D405's Z – accuracy is expected to be within +/- 2% [13]. The Depth Quality Tool (DQT) from RealSense was used to evaluate depth RMS error and Z - accuracy error. To do so, the camera was positioned with the camera facing the same noise

pattern used for on-chip calibration. The distance between the camera and the noise pattern was found and entered into the DQT as a ground truth. The DQT software then reported a Z - accuracy and Depth RMS error accuracy percentage.

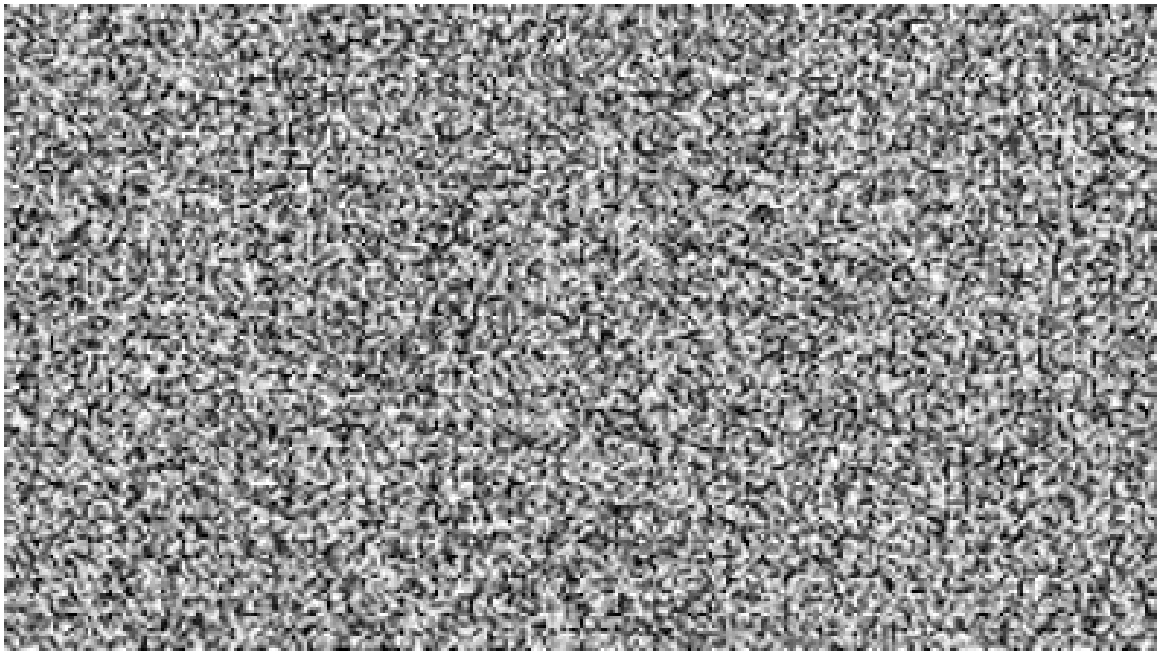


Figure 3.5: Noise pattern provided by RealSense [2].

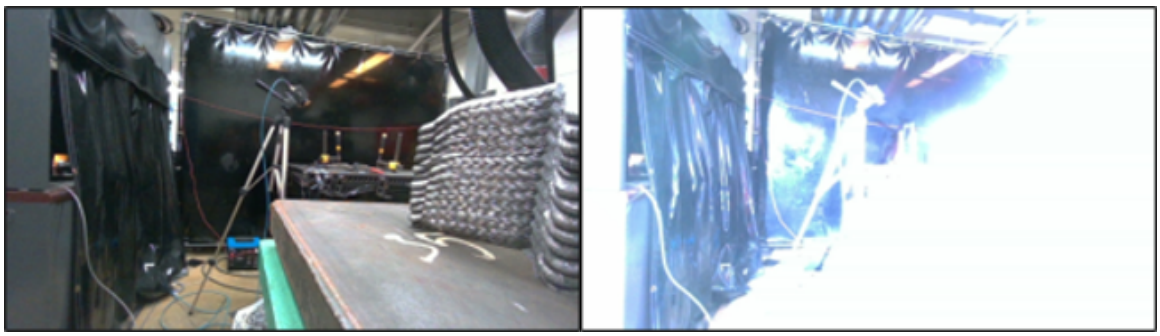
The importance of camera calibration was experienced firsthand during the research. Scans began using the first camera which will be referenced as camera A throughout. Camera A was used until measurement inconsistencies were found during golf ball artifact experiments outlined in Chapter 4. Originally scans using Camera A showed deviations in the diameter of the golf ball artifact of roughly -1.5 mm, but during testing this number changed to a much higher diameter deviation of -6.5 mm indicating a shrinkage of 15% in the overall mesh diameter. The On-Chip health score was determined on Camera A to be 0.01, well within the acceptable range (< 0.25). The DQT was used to evauluate Camera A's depth RMS error and Z - accuracy which were found to be 0.23% and 18.55% respectively. While the camera's RMS error was within the acceptable range, it's Z – accuracy is much larger than the manufacturer's target of +/- 2%. At this point, Camera A was swapped out for the second Intel RealSense D405 camera, which will be referred to as Camera B. Camera B's health score was -0.03, well within the range recommended by RealSense. Using DQT, Camera B's RMS error was 0.07% and its Z – accuracy was 0.11%. Both the RMS error

and the Z – accuracy were found to be within the recommended range. Camera B was then used to redo measurements of the same golf ball geometry as before and deviations were found to be on the order of 1.5 mm, which was expected. Thus, observed changes in scan quality were caused by Camera A becoming uncalibrated. While the source of this change is unknown, the camera had been used in the WAAM cell during extensive printing and was thus subject to both temperature, vibration and potentially impact disturbances. The rapid pivot to a new camera highlights the utility of the work described herein. Camera A could also be re-calibrated through the use of a Dynamic Calibration provided by RealSense, but for the remaining experiments Camera B was implemented instead.

Figure 3.6: Intel D405 in FDM mount.

3.3.2 Time Synchronization

Synchronizing robot position data with camera depth measurements presented a challenge due to difference in data rates and computer timestamps. Robot position data is received as a data packet every 4 milliseconds using an IPOC time stamp. IPOC is the KUKA specific RSI timestamp for the data packet being received in milliseconds. The camera timestamps are based of exact exposure time in the sensor clock in microseconds. Camera frames are being record at a rate of 30 frames per seconds (FPS). To align the position data with the camera frames, an external stimulus is introduced to provide a reliable time reference point. A camera flash was selected to provide the stimulus since it can be easily triggered by the robot and recorded within the camera field of view. An example of a flash occurring within images from the D405 can be seen in Figure 3.7. The camera flash is triggered by the KUKA I/O via the flash's internal PC sync circuit. The flash is captured and identified using OpenCV. Flash is detected by comparing pixels between corresponding frames, quantifying the number of changed pixels between each frame. When the flash occurs, pixel intensities change across the camera field of view. A threshold of 90% of total pixels changed was chosen to identify a flash event. A high threshold ensured the flash was being detected instead of any outside noise. The RSI data shows when the variable "FLASH" was changed from 0 to 1, indicating when the controller triggered the flash. Figure 3.8 shows an RSI output stream during which flash triggering has occurred.



(a) Before Flash **(b)** During Flash

Figure 3.7: D405 RealSense images during flash triggering [1].

By using the RSI data timestamp and the identified flash camera frame timestamp, the time offset between the two data streams is calculated and utilized to temporally synchronize

# Flash	# Ipoc
0	1653721047
0	1653721051
0	1653721055
1	1653721059
1	1653721063
1	1653721067

Figure 3.8: KUKA RSI stream with IPOC timestamp and flash variable change from 0 to 1 [1].

position and camera data. It is possible for a scenario to occur where an exact match of position data cannot be found for a saved camera frame. To accommodate this scenario, linear interpolation is used between closest available sets of position data for the camera frame. Though a scenario like this is not a likely source of error in the current set-up. A ratio of 8.3 robot position data packets are received for every 1 camera frame [1]. With the position data packets being received at a much higher rate, it is unlikely a exact match can not be found.

Figure 3.10 provides an example of surface measurements of an FDM printed propellor geometry both before and after time synchronization is implemented. Figure 3.9 shows an image of the FDM printed propeller used for surface reconstruction. Qualitative improvements in the surface reconstructions can be seen along the edges of the propellor part after time synchronization. Noise is reduced with proper time synchronization since frames are better aligned. These observations demonstrate that time synchronization is a critical step for obtaining reliable surface measurements when combining robot position data with depth frames.

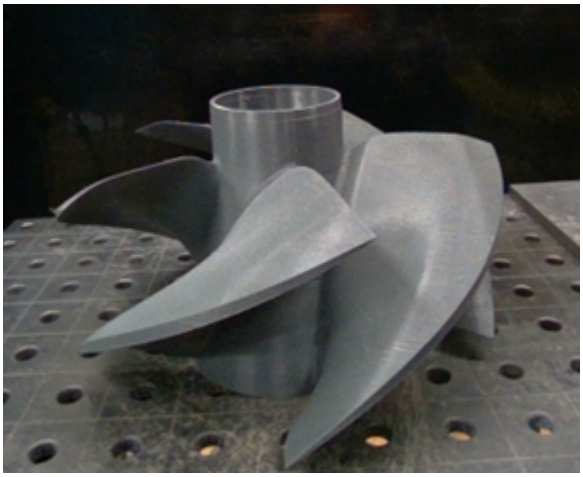
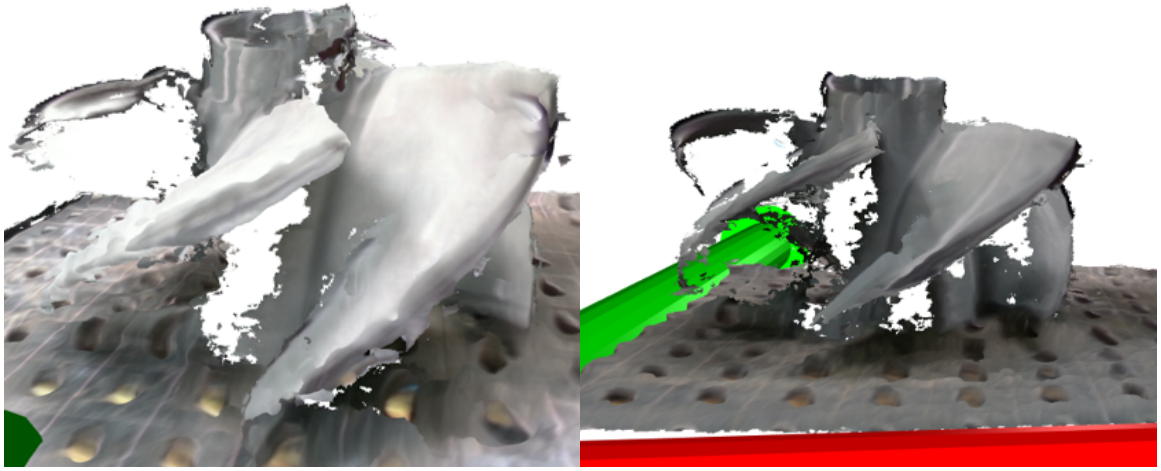


Figure 3.9: FDM printed propellor geometry.



(a) No time synchronization **(b)** With time synchronization

Figure 3.10: Surface meshes of an FDM propellor before and after the implementation of temporal synchronization.

3.3.3 Spatial Synchronization

RSI robot position data is exported relative to the active tool center point (TCP) of the robot, so for accurate camera positioning, the location of the camera on the robot must be determined to define its position as a tool on the robot. The TCP defines the location of the tool's origin relative to its wrist joint location. In KUKA robots, the position data received from the RSI stream specifies the TCP coordinates with respect to the programmed base. Therefore, an accurate TCP is essential to determine the camera's position at the time of each image relative to this base. Standard robot TCP calibration methods require physically touching a known point to calculate the origin of the tool. Since the camera's origin is located at the camera's image plane, an optical method of determining the TCP was implemented using a checkerboard reference pattern and OpenCV.

The checkerboard, consisting of an 11 × 8 grid of black and white squares each measuring 20 mm per side, was printed on an 8.5 × 11" sheet of paper and mounted to a glass plane to ensure flatness. The origin of the checkerboard is defined by OpenCV as the upper left inner corner of the pattern in the camera's image frame. A representative checkerboard pattern along with the defined origin can be seen in Figure 3.11

The checkerboard was placed on a pedestal near the robot, Figure 3.14. To align the origin of the checkerboard with the origin of the robot base a "dummy" tool must first be created. Any tool can be used if it can be set accurately through traditional touch-based TCP calibration methods. The torch on the robot was calibrated as this tool using KUKA's built in XYZ 4-point method. The torch was touched off on a calibration spire from 4 distinct positions and angles producing an accurate TCP calibration. An image of the torch tip probing the spire is located on Figure 3.12. After calibrating the torch, a new base was created on the robot. A base was created through KUKA's 3 - point method. To begin, the robot was jogged where the tip of the torch is touching the checkerboard in the upper left inner corner. The origin of the new base is set as this location. The robot was then jogged to touch the bottom left inner corner of the pattern, setting the positive Y axis. Finally, the positive X axis was set by touching the torch to the upper right inner corner of the pattern. Figure 3.13 shows the orientation of the coordinate system set as the base on the robot. By

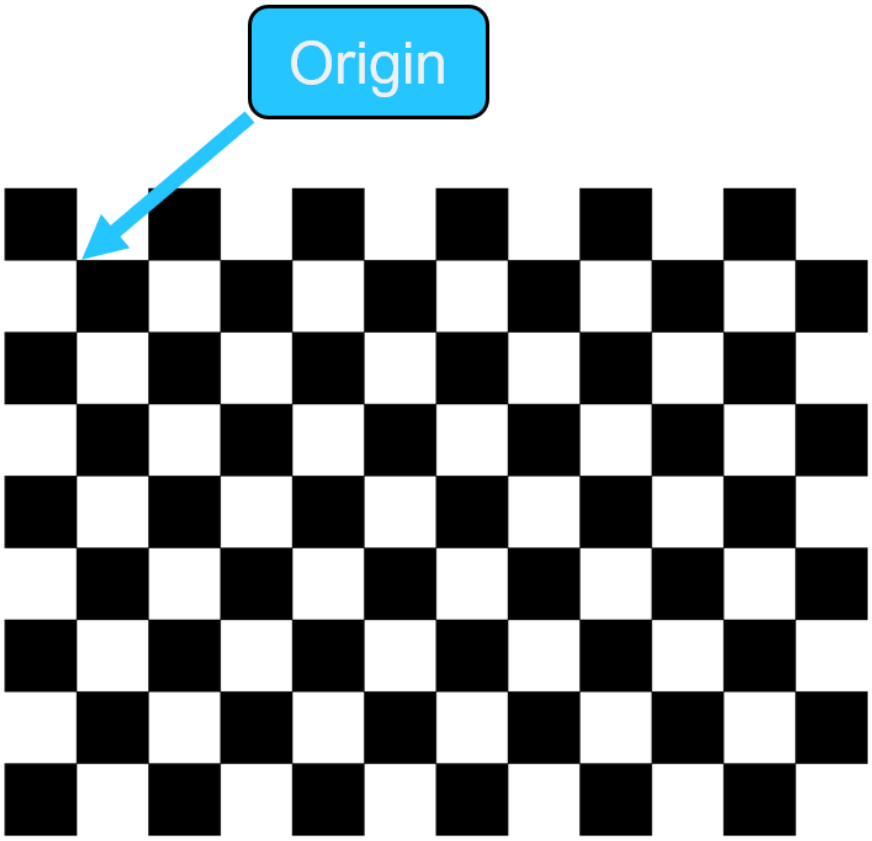


Figure 3.11: Example checkerboard pattern with labeled origin.

setting the robot base coordinate system this way, it matches the coordinate system OpenCV will identify on the checkerboard pattern. It was critical to align the base of the origin with the origin of the checkerboard because otherwise the position data being recorded during TCP calibration would not be recorded in the same coordinate frame as the camera pose estimation. After base calibration, the robot was then positioned above the checkerboard so that the full checkerboard pattern was visible within the camera's field of view. At a single instant both an image and the robot position were recorded. Robot position was recorded with the active tool frame set to null frame, reporting the position of the wrist in relation to the robot base (checkerboard pattern). An example robot, checkerboard, and camera position during camera TCP calibration can be seen in Figure 3.14.

A python script, utilizing OpenCV, is used to calculate the transformation matrix of the checkerboard relative to the camera. An example matrix can be seen in Equation 3.1. Where $\mathbf{T}_{\text{cb\_to\_cam}}$ denotes the homogeneous transformation matrix for the checkerboard relative to the camera. $\mathbf{R}_{\text{cb\_to\_cam}}$ is the corresponding 3 x 3 rotation matrix , and *tvec* is the 3 x 1 translation vector.

$$T_{cb\_to\_cam} = \begin{bmatrix} R_{cb\_to\_cam} & tvec \\ 0 & 1 \end{bmatrix} \tag{3.1}$$

To find $\mathbf{T}_{\text{cam\_to\_cb}}$, the homogeneous transformation matrix of the camera pose relative to the checkerboard, the matrix must be inverted. The rotation matrix is inverted by taking its transpose yielding $\mathbf{R}_{\text{cam\_to\_cb}}$. The corresponding translation vector is obtained by multiplying the transposed rotation matrix by the negative original translation vector. The operations are shown in Equations 3.2 and 3.3. Here, $\mathbf{R}_{\text{cam\_to\_cb}}$ is the corresponding 3 x 3 rotation matrix and $\mathbf{t}_{\text{cam\_to\_cb}}$ is the 3 x 1 translation vector for the transformation matrix of the camera pose relative to the checkerboard

$$R_{cam\_to\_cb} = R_{cb\_to\_cam}^{T} \tag{3.2}$$

$$t_{cam\_to\_cb} = -R_{cam\_to\_cb} * tvec \tag{3.3}$$

Figure 3.12: Demonstration of 4-Point touch method with the calibration. spire

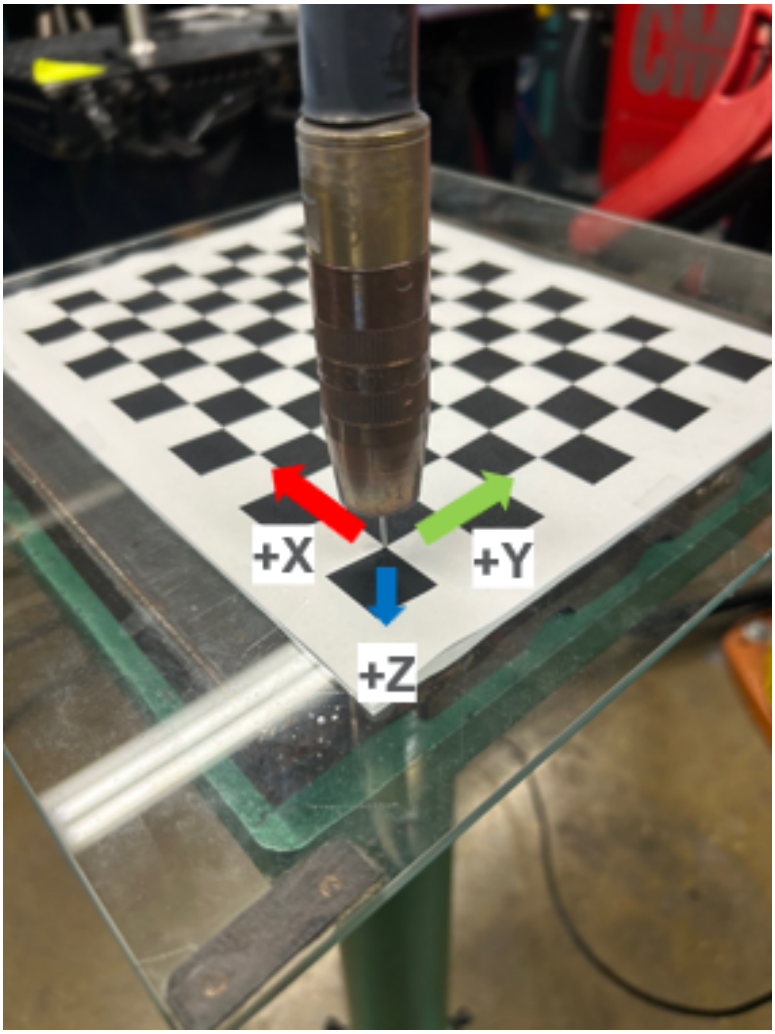


Figure 3.13: Checkerboard base coordinate system.

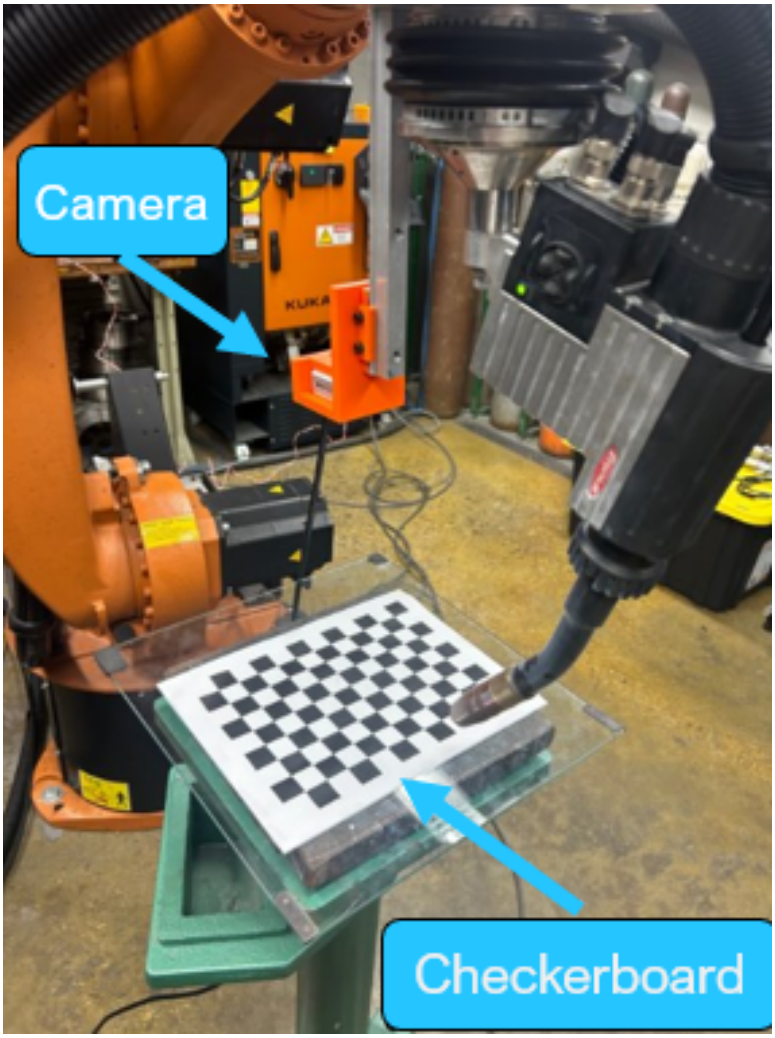


Figure 3.14: Position of the camera and robot for data collection when calibrating the camera TCP.

Once inverted the transformation matrix now reflects the camera position relative to the base as , $\mathbf{T}_{\text{cam\_to\_cb}}$ and is represented by Equation 3.4.

$$T_{cam\_to\_cb} = \begin{bmatrix} R_{cam\_to\_cb} & t_{cam\_to\_cb} \\ 0 & 1 \end{bmatrix} \tag{3.4}$$

The robot position recorded with no tool frame while taking the camera image is turned into transformation matrix for the wrist pose in checkerboard frame, $\mathbf{T}_{\text{wrist\_to\_cb}}$, where $\mathbf{R}_{\text{cam\_to\_cb}}$ is the corresponding 3 x 3 rotation matrix and $\mathbf{tvec}_{\text{wrist}}$ is the 3 x 1 translation vector. The transformation matrix is represented in Equation 3.5.

$$T_{wrist\_to\_cb} = \begin{bmatrix} R_{wrist\_to\_cb} & tvec_{wrist} \\ 0 & 1 \end{bmatrix} \tag{3.5}$$

Equation 3.5 must be inverted to represent the checkerboard pose in the wrist frame. Similarly as inverting matrix $\mathbf{T}_{\text{cb\_to\_cam}}$, the rotation matrix, $\mathbf{R}_{\text{wrist\_to\_cb}}$, is inverted by taking it's transpose yielding $\mathbf{R}_{\text{cb\_to\_wrist}}$. The corresponding translation vector is obtained by multiplying the transposed rotation matrix by the negative original translation vector yielding $\mathbf{t}_{\text{cb\_to\_wrist}}$. The operations are shown in Equations 3.6 and 3.7. Once inverted the transformation matrix now reflects the checkerboard pose relative to the wrist base as, $\mathbf{T}_{\text{cam\_to\_cb}}$ and is represented by Equation 3.8.

$$R_{cb\_to\_wrist} = R^{T}_{wrist\_to\_cb} \tag{3.6}$$

$$t_{cb\_to\_wrist} = -R_{wrist\_to\_cb} * tvec_{wrist} \tag{3.7}$$

$$T_{cb\_to\_wrist} = \begin{bmatrix} R_{cb\_to\_wrist} & t_{cb\_to\_wrist} \\ 0 & 1 \end{bmatrix} \tag{3.8}$$

Multiplying $\mathbf{T}_{\text{cb\_to\_wrist}}$ by $\mathbf{T}_{\text{cam\_to\_cb}}$, cancels the checkerboard frame, yielding direct transformation matrix $\mathbf{T}_{\text{cam\_to\_wrist}}$ representing the camera pose relative to the robot wrist

which can also be referred to as the tool center point of the camera. The final transformation calculation is represented in Equation 3.9.

$$T_{cam\_to\_wrist} = T_{cb\_to\_wrist} * T_{cam\_to\_cb} \tag{3.9}$$

Figure 3.15 shows an FDM printed propeller geometry measured using the robotic scanning system after now performing both time and spatial synchronization. Compared to Figure 3.10a, surface detail and quality show significant improvement over a measurement without any synchronization. When comparing the mesh generated with both time and spatial synchronization to Figure 3.10b with only time synchronization, improvements are seen in the weld table and propellor geometry indicating both spatial and time synchronization are critical for accurate surface reconstruction. The improvement is most evident on the weld table, where calibration yields a more precise and accurate representation of the flat surface geometry.

Figure 3.15: FDM propeller mesh after time and spatial synchronization.

Chapter 4

Experiments

4.1 Reconstruction and Quality Analysis

4.1.1 Artifact Selection

While the system has proven effective in generating meshes that visually appear to be high in quality, a more quantitative and analytical approach to assessing scan accuracy was desired. A common method of determining accuracy in optical coordinate metrology is using a calibrated artifact. A calibrated artifact is a physical reference object with precisely known shape and dimensions. The initial strategy was to collect measurements from an off-the-shelf calibrated artifact designed for 3D scanning. The artifact consists of spheres with varying diameters and separation distances. The off-the-shelf artifact was eventually abandoned, however, due to limitations of the low-cost system, see Section 5.1.

In lieu of the off-the-shelf artifact, alternate reference geometries where explored. Since off-the-shelf calibrated artifacts commonly utilize spheres, they were explored first. A spherical geometry allows for a best fit diameter to be easily calculated from measurement data. Such a fit then facilities the quantification of sphere diameter and center location. Thus, efforts focused on identifying a reference sphere artifact with greater surface texture to enhance scan accuracy. Consequently, a golf ball was proposed as an alternative. Golf balls have dimples on the surface to reduce aerodynamic drag and increase the distance they travel. These dimples create surface contrast that aid in depth measurement for the D405

camera. The United States Golf Association (USGA) mandates that golf balls must have a minimum diameter of 1.680 inches or 42.67 mm measured from the outside of the dimples for the ball to be considered a legal ball [21]. The equipment rule is to ensure that a player does not use a smaller ball which would travel longer distances after impact. A Titleist Pro V1X golf ball was ultimately selected to be used as an artifact for testing the accuracy of the low-cost system. Although manufacturers keep detailed dimensional tolerances proprietary to protect their production processes, golf ball are generally engineered to be as close as possible to the minimum allowed diameter to maximize driving distance. The diameter of the select golf ball artifact was verified by measuring across the dimples with digital calipers. Size was found to be 42.67 mm.

While a golf ball has a relatively precise diameter, this value is fixed and does not provide a size range across which the measurement system could be evaluated. Thus, other measurement artifacts have been explored. Several artifacts were designed featuring different types of primitive geometry which included spheres, cylinders, cubes, and fins. These artifacts were designed to be of varying sizes and distances apart from each other. Figures 4.1 - 4.4 show models of the artifacts used in experimentation. Tables 4.1 - 4.4 show the corresponding nominal dimensions for each artifact. These artifacts were 3D printed using an FDM printer. Though 3D printing is not the most accurate method of manufacturing, it was explored for iteration of artifact designs. Feature dimensions were verified through measurement with digital calipers. Tables 4.5 - 4.9 demonstrate the actual dimensions, measured with calipers, of the artifacts after manufacturing. White, marbled PLA filament with black speckles was used for the printed artifacts. Such filament produced surfaces a finish very similar to marbled sculptures. The finish mimicked a noise pattern being projected on the artifacts. The FDM manufactured artifacts can be seen in Figure 4.5.

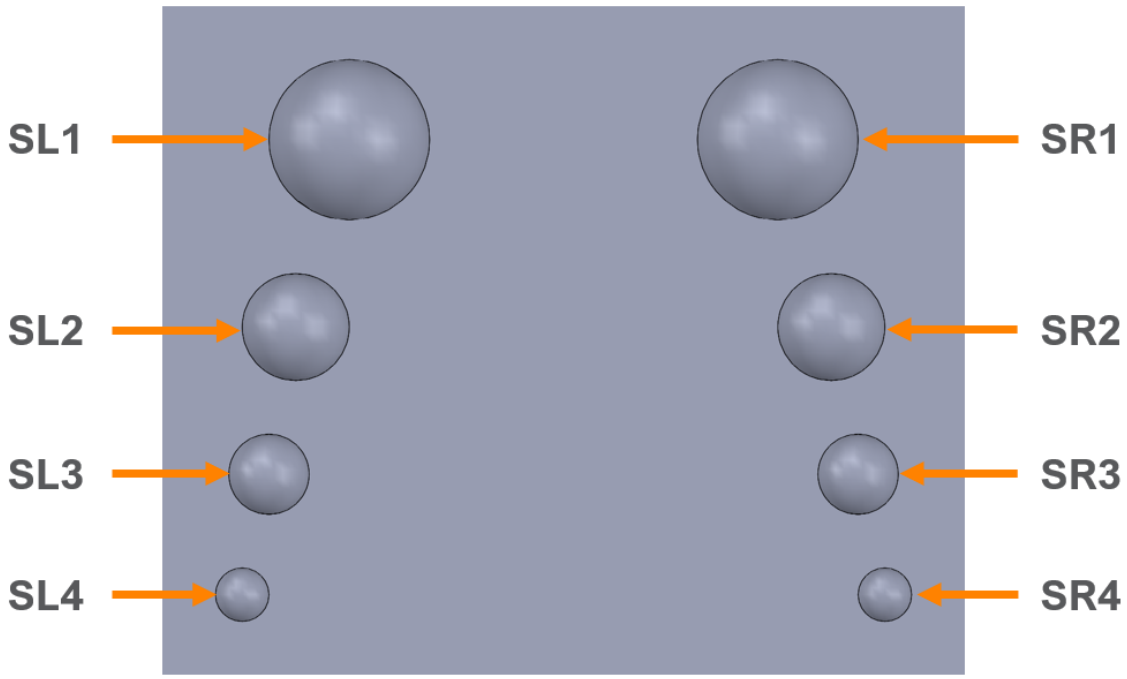


Figure 4.1: Top down view of sphere artifact with labeled spheres.

Table 4.1: Nominal Sphere Artifact Dimensions

Sphere Pairs	Nominal Diameter L,R (mm)	Nominal Center to Center Distance (mm)
SL1-SR1	30	80
SL2-SR2	20	100
SL3-SR3	15	110
SL4-SR4	10	120

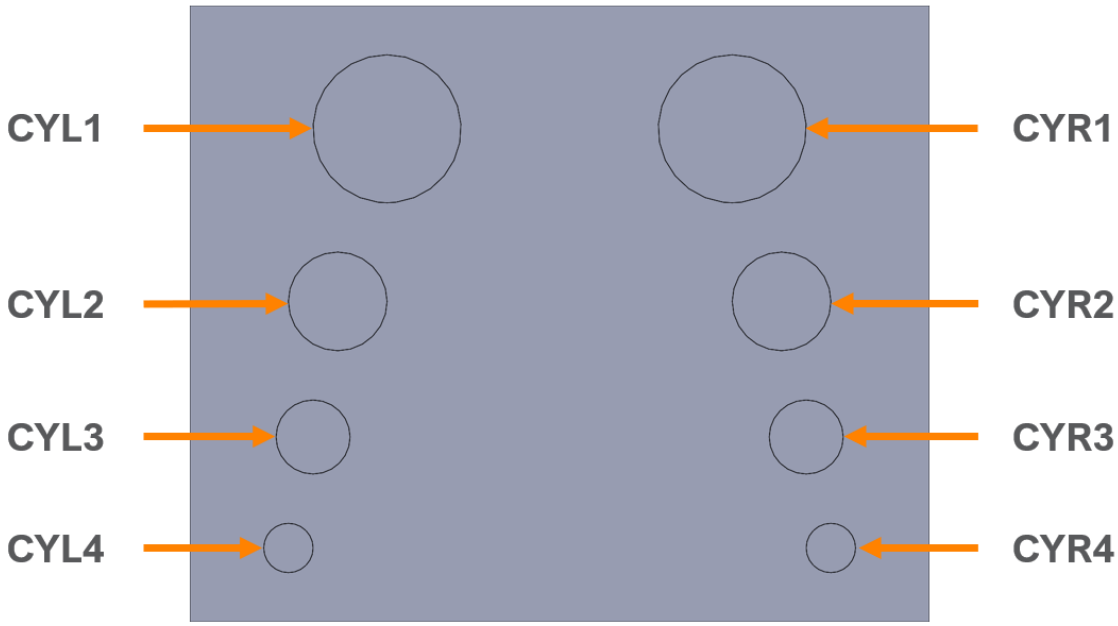


Figure 4.2: Top down view of cylinder artifact with labeled cylinders.

Table 4.2: Nominal Cylinder Artifact Dimensions

Cylinder Pairs	Nominal Diameter L,R (mm)	Cylinder Height L,R (mm)	Nominal Center to Center Distance (mm)
CYL1-CYR1	30	30	70
CYL2-CYR2	20	20	90
CYL3-CYR3	15	15	100
CYL4-CYR4	10	10	110

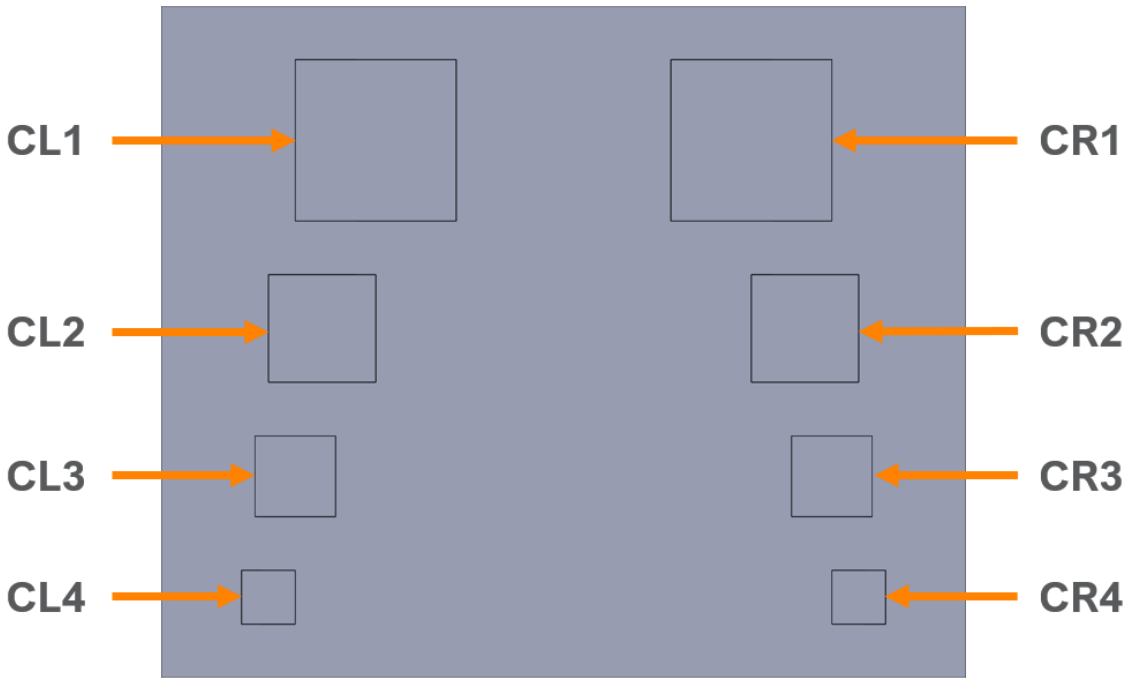


Figure 4.3: Top down view of cube artifact with labeled cubes.

Table 4.3: Nominal Cube Artifact Dimensions

Cube Pairs	Nominal Edge Length L,R (mm)	Nominal Center to Center Distance (mm)
CL1-CR1	30	70
CL2-CR2	20	90
CL3-CR3	15	100
CL4-CR4	10	110

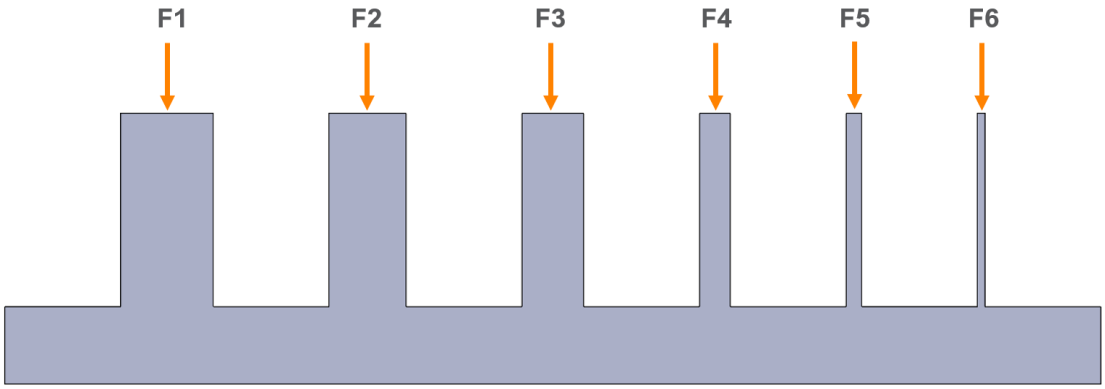


Figure 4.4: Front view of fin artifact with labeled fins.

Table 4.4: Nominal Fin Artifact Dimensions

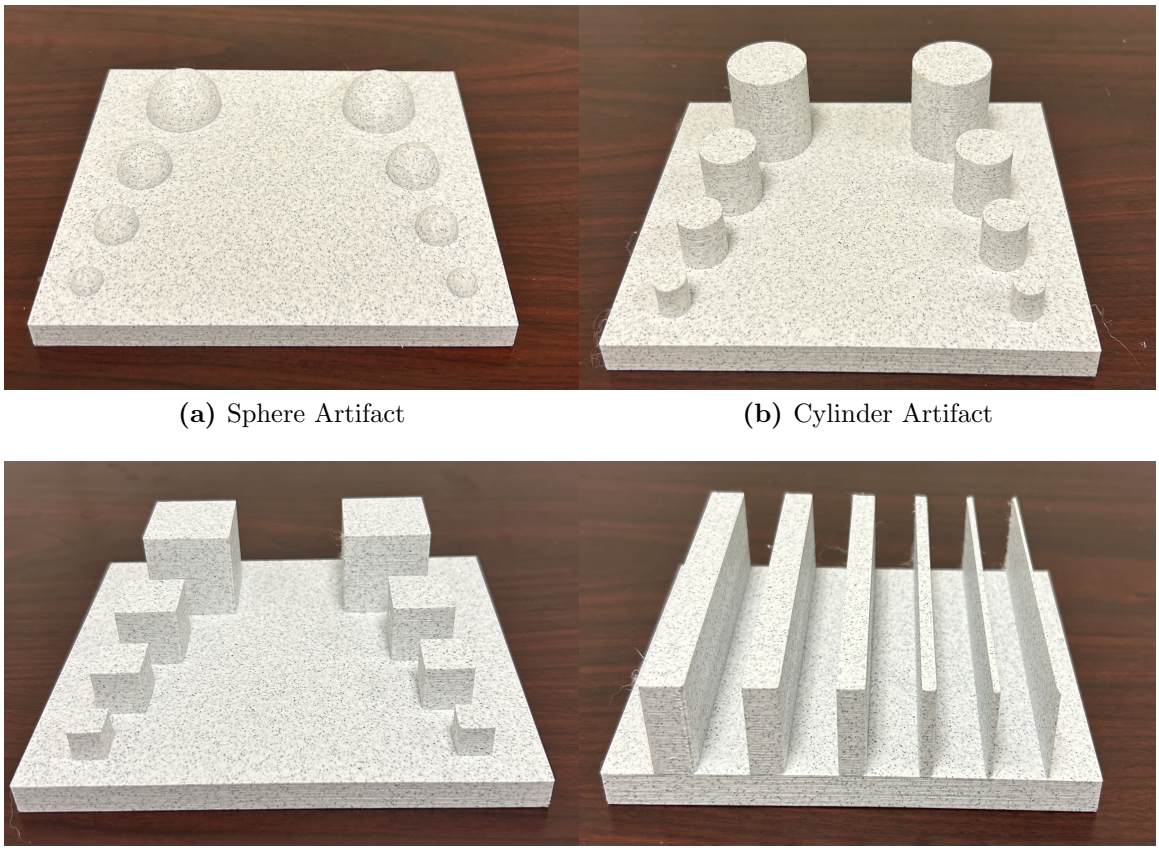
Fin Number	Nominal Thickness (mm)
F1	12
F2	10
F3	8
F4	4
F5	2
F6	1

Table 4.5: Actual Sphere Artifact Dimensions

Sphere Pairs	Diameter L (mm)	Diameter R (mm)
SL1-SR1	29.75	29.71
SL2-SR2	19.68	19.67
SL3-SR3	14.68	14.61
SL4-SR4	9.72	9.71

Table 4.6: Actual Cylinder Artifact Dimensions

Cylinder Pairs	Diameter L (mm)	Height L (mm)	Diameter R (mm)	Height R (mm)
CYL1-CYR1	29.73	30.00	29.75	29.95
CYL2-CYR2	19.92	19.96	19.87	19.98
CYL3-CYR3	14.91	14.97	14.98	15.00
CYL4-CYR4	9.87	10.06	9.89	9.93



(a) Sphere Artifact **(b)** Cylinder Artifact

(c) Cube Artifact **(d)** Fin Artifact

Figure 4.5: FDM primitive geometry artifacts.

Table 4.7: Actual Left Side Cube Artifact Dimensions

Cube	Length L (mm)	Width L (mm)	Height L (mm)
CL1	30.06	30.16	30.06
CL2	20.11	20.26	20.01
CL3	15.24	15.55	15.02
CL4	10.32	10.13	9.94

Table 4.8: Actual Right Side Cube Artifact Dimensions

Cube	Length R (mm)	Width R (mm)	Height R (mm)
CR1	30.05	30.08	29.29
CR2	20.13	20.12	20.04
CR3	15.15	15.42	15.03
CR4	10.21	10.15	10.00

Table 4.9: Actual Fin Artifact Dimensions

Fin	Thickness (mm)
F1	11.97
F2	9.99
F3	8.01
F4	4.02
F5	2.02
F6	1.08

4.1.2 Scanning Strategy

Each artifact was scanned while moving through 5 different positions with each position being 45 degrees away from the last. The torch was removed for these experiments to aid with positioning of the camera. The camera was kept at a 45-degree angle downwards in the horizontal plane, always facing towards the artifact while it was maneuvering around the part in a half circle in the XY plane. Figure 4.6 demonstrates the relative angles the camera paused at during scanning. The D405 was kept 15 cm away from the center of the artifact, well within the working range of the camera (7-50 cm). At each position the camera was paused facing the part for 5 seconds to ensure adequate frame collection at each angle. The artifacts were placed onto a level pedestal with a printed noise pattern beneath them. The noise pattern helps the camera with depth perception as the clean pedestal table can cause false depth readings for the camera. Figure 4.7 shows the robot in the first position of five to scan the golf ball. It is important to note that due to limitations in the camera's field of view and the robot's reach, a small section on the back side of the golf ball was not scanned during measurements. The bottom section of the ball is naturally also unavailable since it is sitting on the pedestal surface.

For the golf ball artifact experiments, the artifact was scanned ten different times for each programmed robot velocity with the goal of observing how scan velocity affects surface reconstruction. Scan velocity was selected based on limitations of the robot. The KUKA KR-6 has a maximum velocity of 200 cm/s, which is an unreasonable velocity to expect during the short travel distances required for artifact scanning. To find the maximum reasonable

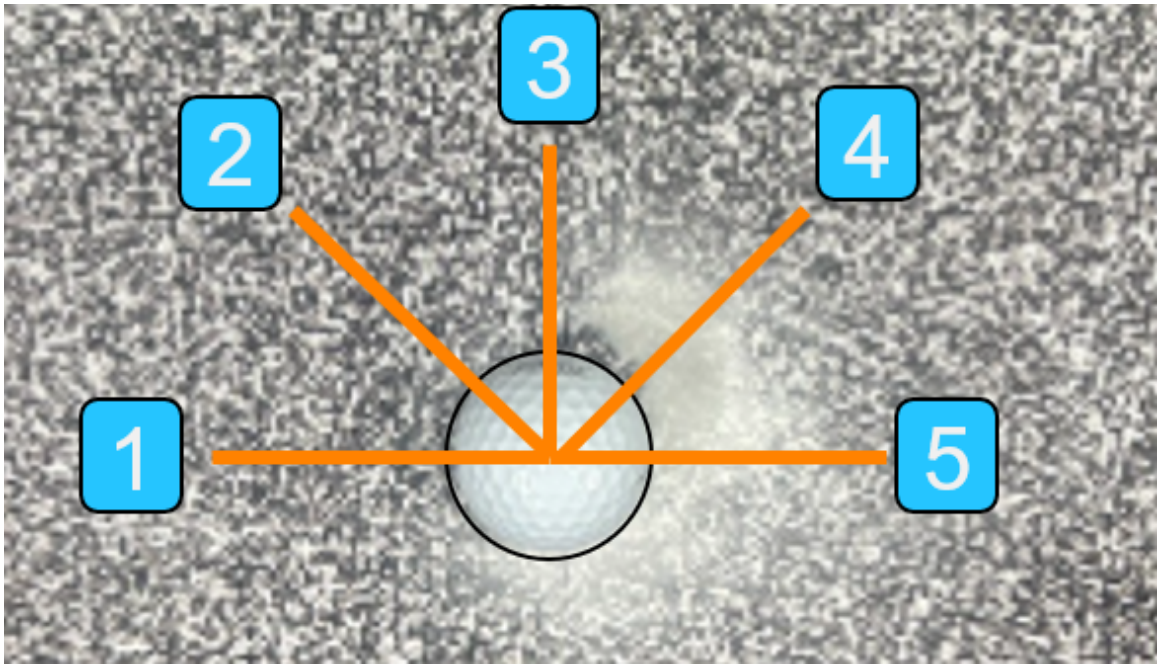


Figure 4.6: Programmed scan positions for measurement of a golf ball artifact.

velocity, the robot was programmed to move at 100 cm/s along the desired path. The robot velocity profile was then calculated based on the RSI position data and plotted in Figure 4.8. It can be observed that the robot achieves a velocity of only 17.5 cm/s before the 5 second pause indicating the fastest speed of travel for the current scan path. Thus, the different scan velocities for the golf ball artifact were 0.5, 1, 5, 10 and 15 cm/s. The scan velocity for each of the 5 programmed speed is plotted in Figure 4.9.

For the primitive 3D-printed artifacts, a similar measurement approach applied to the golf ball artifact was utilized. The robot velocity was set to 1 cm/s and each artifact was scanned one time. The goal of these scans was to observe how changes in artifact geometry effect scan quality.

4.1.3 Examining TSDF and Weight Values

Open3D's Tensor API allows for the TSDF and weight values to be extracted from a voxel block grid and stored for further analysis. For the golf ball artifacts, the number of voxels along with the average TSDF and weight values were recorded to a .csv file to evaluate the effects of robot velocities on reconstruction values. Overall averages of voxel count,

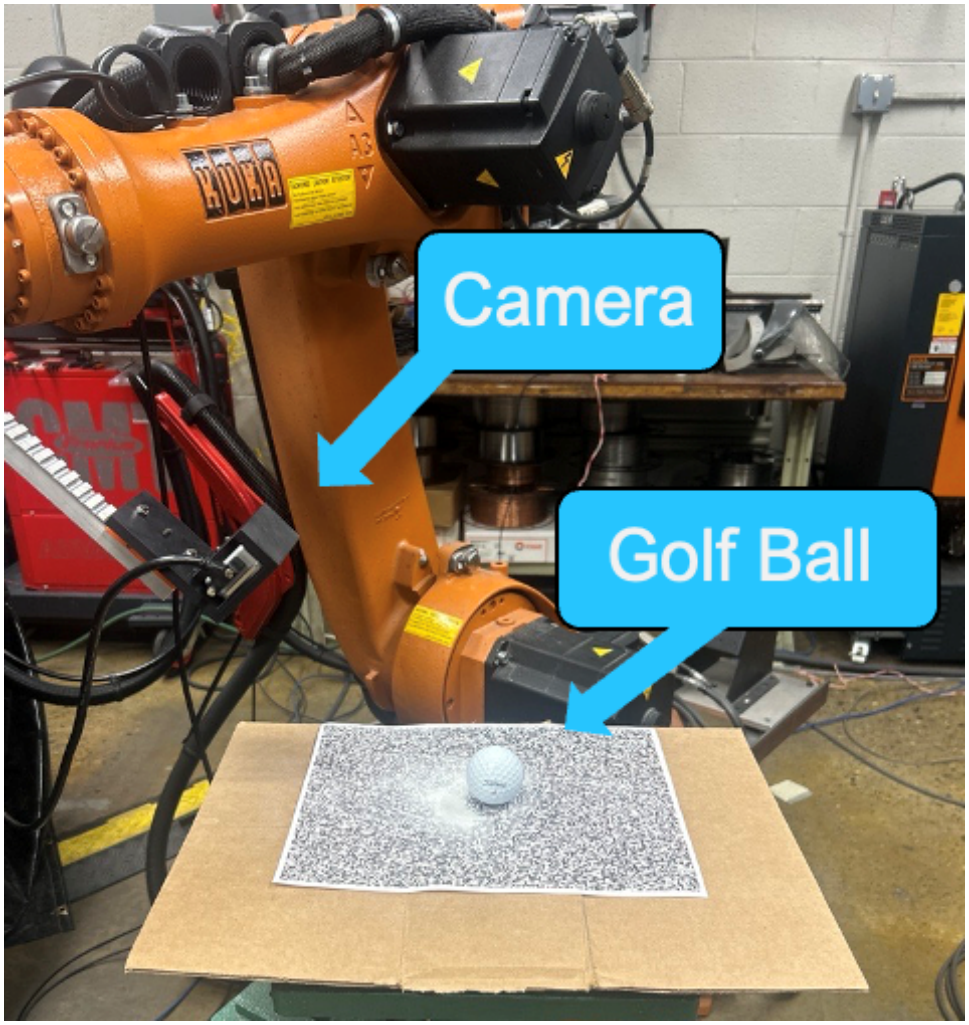


Figure 4.7: Robot in position 1 of the golf ball artifact scan path.

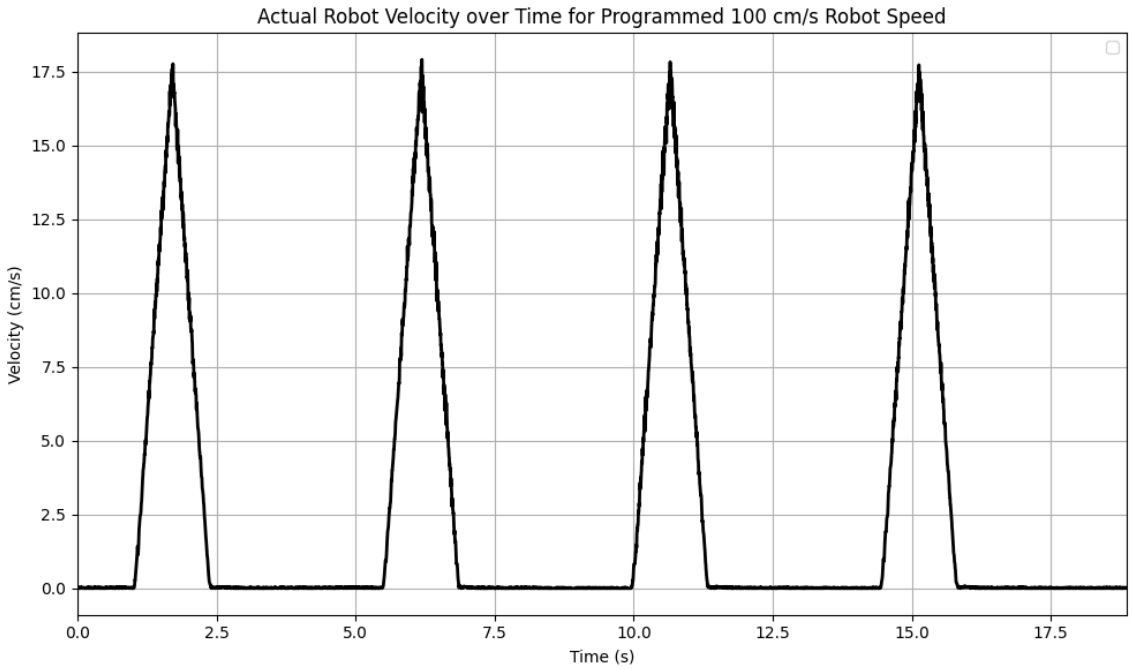


Figure 4.8: Recorded Robot velocity during scan duration for programmed 100 cm/s robot speed.

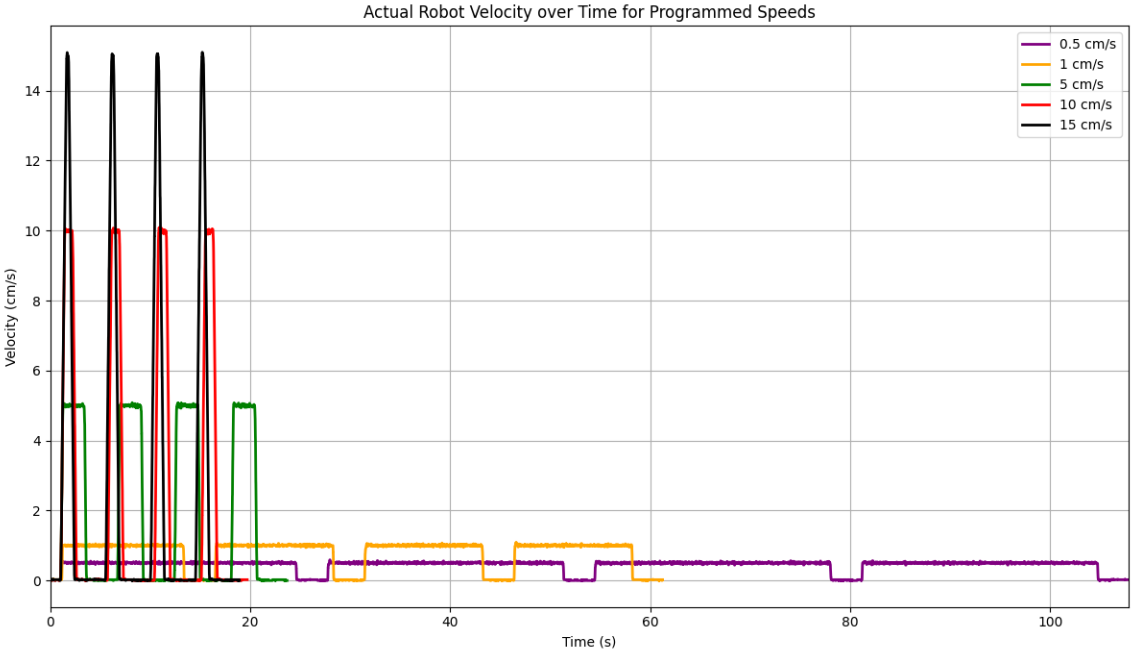


Figure 4.9: Recorded Robot velocity during scan duration for all programmed scan speeds.

TSDF value, and weight value were calculated across the ten scan speeds, along with the corresponding standard deviation. As discussed in Section 3.2.2, lower average TSDF values indicate a more precise surface reconstruction. Conversely, a higher average weight indicates voxels were observed multiple times demonstrating a high confidence in the assigned TSDF value [22].

4.1.4 Evaluating Artifacts

To evaluate the golf ball scans, two different analysis methods were implemented. For both methods, the meshes were first imported into MeshLab, an open-source software for processing and editing 3D triangular meshes [23]. MeshLab was used to trim up unnecessary parts of the mesh such as the plate and any background noise. The mesh was scaled by 1000x to convert it from meters to millimeters. The mesh was then exported from MeshLab and imported into a Python script for a best-fit analysis. The best-fit method used a least squares method to calculate a best-fit diameter to the trimmed golf ball mesh data. The calculated best-fit diameter was then recorded for each scan for each different speed.

The second method used functions built into MeshLab. After the golf ball mesh was trimmed and transformed to the correct scale, a control sphere was created within MeshLab with a diameter of 42.67 mm, matching the standard golf ball size. The measured mesh was aligned to the control sphere using an iterative closest point (ICP) method. Once aligned a comparison to the control sphere was made and minimum and maximum deviations were recorded into an Excel file for each scan. The mesh was colorized based on deviation from the control sphere with green representing aligning with surface, red indicating deviation inside the control sphere, and blue indicating deviations outside the control sphere. The purpose of this analysis was to quantify deviations from the control within the mesh data prior to applying a best-fit alignment, enabling a more detailed evaluation of the raw mesh characteristics by showing a full error map over the entire surface. For the primitive geometry artifacts, evaluation was focused on visual inspections of the generated mesh. The Meshes were imported into MeshLab, where unnecessary parts of the mesh were trimmed away and each mesh was scaled up by 1000x for meters to millimeters unit conversion. The resulting surface mesh was observed for noticeable discrepancies compared to the real-life model such

as missing regions or irregular surface features. This assessment focused on how different geometric features influence the overall scan quality,

4.2 In-Situ Scanning

Ultimately the goal of this research is to implement the low-cost measurement system into a WAAM processes to use for measuring part distortion. To test this system capability, an experiment was conducted by scanning during WAAM processes to mimic an industry setting where this system could be beneficial. A large (18" x 12" x 6.5") aluminum "Power T" was constructed for another research project within the lab shown in Figure 4.11. The part had a 3-bead wide wall consisting of 45 layers and was completed over five days of printing. The camera remained on the robot with the torch for the entire build process. To protect the camera and the FDM printed mount, aluminum foil was wrapped around the camera to protect from spatter and mitigate some of the heat from welding, Figure 4.10. A welding lens was also cut to size and placed in front of the camera to protect its lens from spatter. The welding lens remained fixed throughout all scans. The robot's trajectory was manually programmed, combining circular and linear movements to closely follow the contours of the build. The camera was consistently oriented with the camera lens directed towards the part and it's position was maintained within the required working distance of the camera (7-50 cm). A scan of the the first layer was conducted, with a scan then collected every 5 layers resulting in a total of 10 surface measurements. These layers were subsequently processed, and a Python script stacked them sequentially, effectively simulating the additive build process.

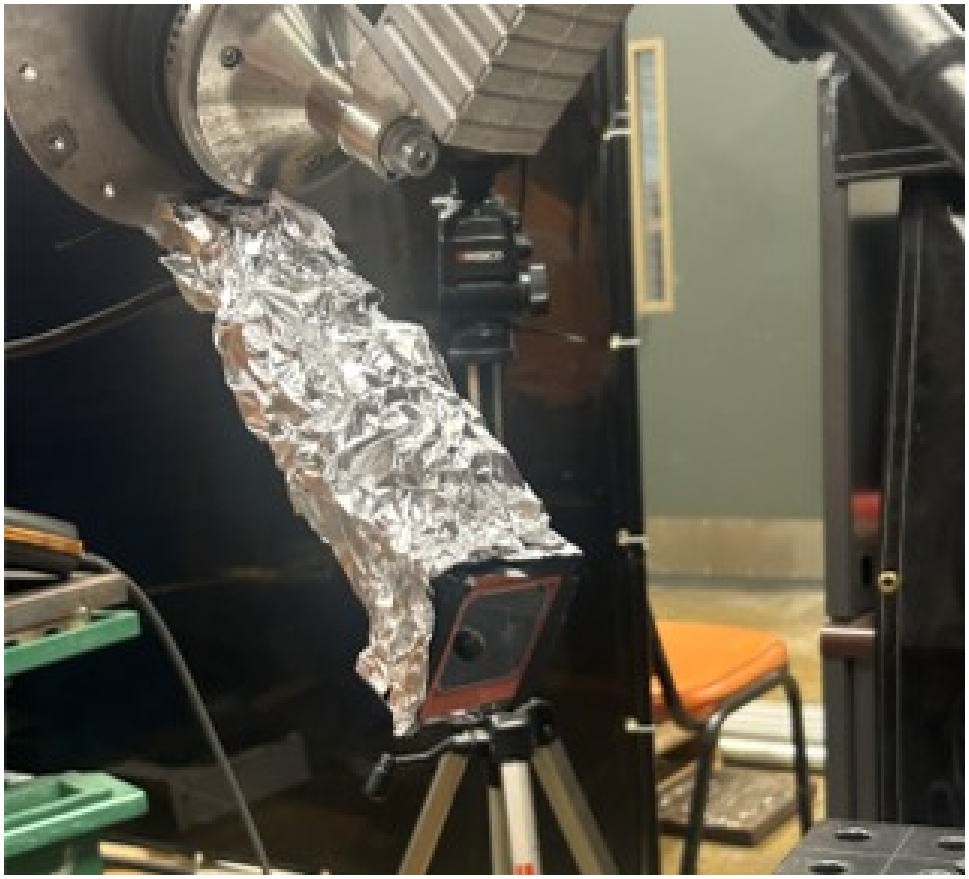


Figure 4.10: D405 set-up for part measurement during a WAAM build.

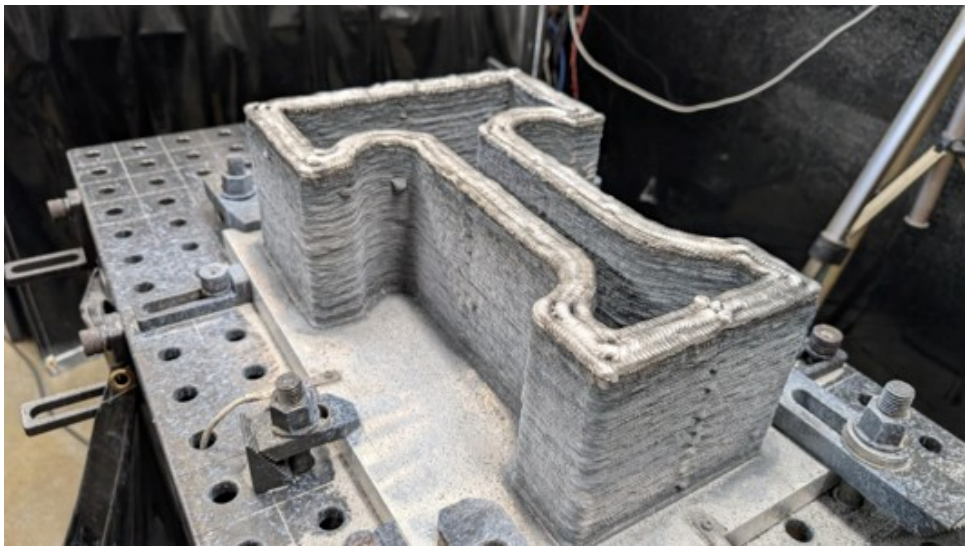


Figure 4.11: Aluminum WAAM Power T.

Chapter 5

System Performance

5.1 Initial Artifact Results

Initially an off-the-shelf artifact was selected to be used to perform measurements. This strategy was eventually abandoned due to limitations of the low-cost system. An Intel RealSense D405 does not use an IR projector to aid with depth perception, therefore creating a limitation when scanning low textured items. Low surface texture does not pose a significant challenge when measuring WAAM-manufactured components, as the inherent layer lines and surface debris enhance depth perception for the D405 sensor. In contrast, low texture is commonly observed in off-the-shelf calibrated artifacts due to their highly uniform manufacturing processes. For traditional scanners, low-texture is remedied by a spray or coating applied to the surface of the artifact. The GOM PSA400 was coated with a titanium dioxide powder, which for traditional scanners would reduce reflectivity and significantly improve reconstruction quality. The GOM PSA400 coated with the titanium dioxide powder can be seen in Figure 5.1. For the low-cost system, the titanium dioxide coating did not create a texture capable of improving depth calculation. An example scan using the low-cost system on the GOM PSA400 artifact can be seen in Figure 5.2.

Figure 5.1: GOM PSA400 calibration artifact sprayed with titanium dioxide powder [3].

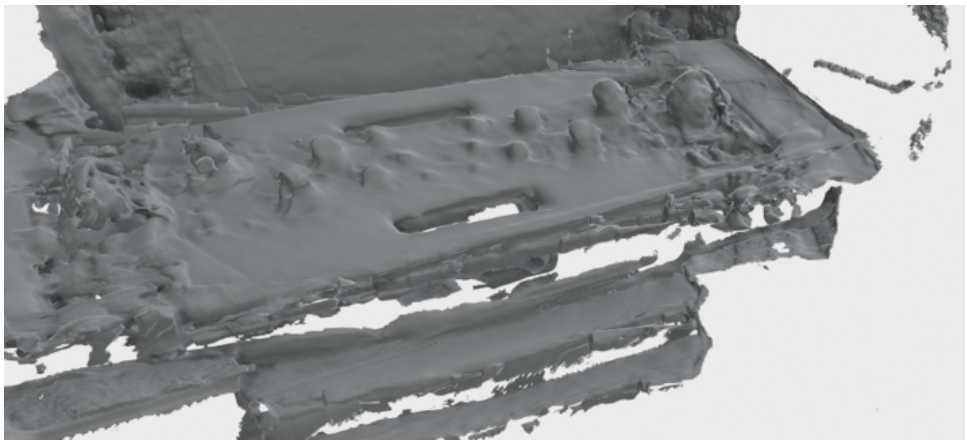


Figure 5.2: Surface reconstruction of the GOM PSA400 artifact

.

5.2 Influence of Scan Speed

The first system performance characteristic which was explored was the impact of robot scan speed on measurement results. For this work, the ProV1x golf ball was scanned at five distinct robot speeds, with ten scans conducted at each speed. The intent was to investigate how scanning speed influences surface reconstruction and the accuracy of scanner measurements.

5.2.1 Scan Data

Initially, the quantity of camera frames captured during each scan was analyzed. As can be seen from Figure 5.3 the expected trend of reducing frames with speed is shown. Reducing the number of frames also significantly decreased processing time during operations such as alignment and integration. Processing time on average for 0.5 cm/s is 140 seconds while for 15 cm/s it is only 22 seconds.

Another trend was the distance traversed by the robot between frames. For each scan, the positional data of consecutive frames was calculated to determine the average inter-frame travel distance. Over increasing velocities, Figure 5.4, the travel distance between frames increases, as expected. Such distance is important as it may create measurement errors due to motion blur. Feature data may also be missed due to the camera moving past a part feature before it collects enough image data to adequately define surface voxels.

Camera frames and robot motion couple to produce the voxels with necessary for surface reconstruction. As anticipated, the average voxel count exhibited a decreasing trend with increasing robot speed, Figure 5.5, mirroring the general trend observed for frame count. The only exception to the trend was between 10 and 15 cm/s where a slight increase in voxel count occurred for the faster speed. The observed increase is small, and may be due to the camera traversing larger distances between frames, resulting in reduced overlap between consecutive depth maps. The TSDF integration may treat these non-overlapping frames as new voxels, inflating the voxel count even if the surface coverage is similar. Error bars were included on Figure 5.5 as voxel count was the only observed trend in this section to have a standard deviation large enough to be visible on a plot.

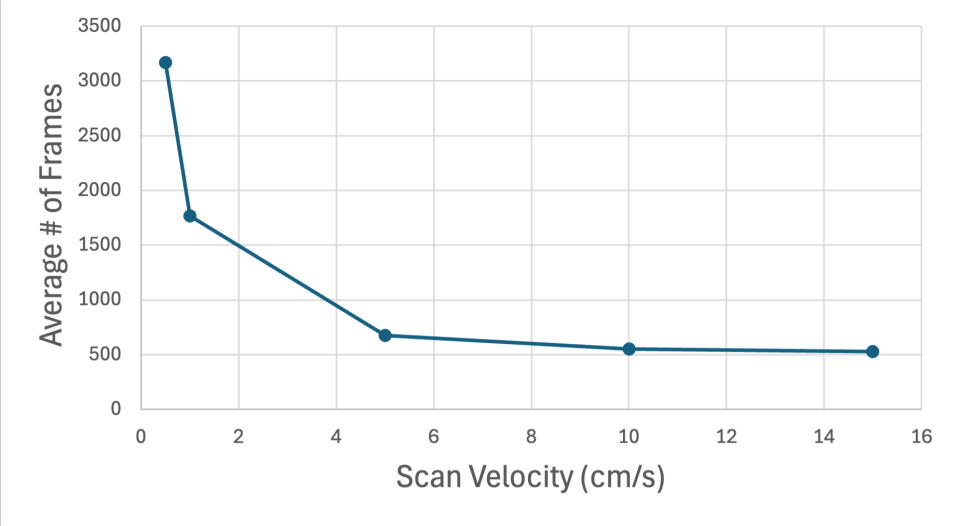


Figure 5.3: Effect of scan velocity on average frame count.

Due to decreasing voxel count and distance traveled between frames increasing while scan speed increased, it was expected the average TSDF value would also increase because each voxel would receive fewer depth observations. The fewer depth observations would cause the TSDF value to become less stable when it is integrated over fewer frames, causing its value to drift farther from zero. For scan velocities of 1 cm/s and higher, this trend can be observed in Figure 5.6. The average TSDF value dropped, however, between the scan speeds of 0.5 and 1 cm/s. A plausible explanation for the elevated TSDF values at 0.5 cm/s is that slower scanning speeds collect more camera data and produce a broader range of TSDF values due to there being more time to capture the background scene. Finally, weight values were observed to decrease as the robot measurement speed increased, Figure 5.7. Such a trend was expected since both frame and voxel counts decrease with increasing speed and reduce the number of surface observations available per voxel. Such a decrease suggests lower confidence in surface measurement locations, which will couple to the measurement errors discussed in the next section.

Visual observation of changes in the golf ball surface meshes with robot scan speed correlate well to the trends discussed here. Figure 5.8 shows the resulting surface meshes for the slowest and fastest robot scan speeds, 0.5 and 15 cm/s respectively. The fastest speed surface mesh has degraded to an ovoid shape rather than a sphere at the slower speed. At

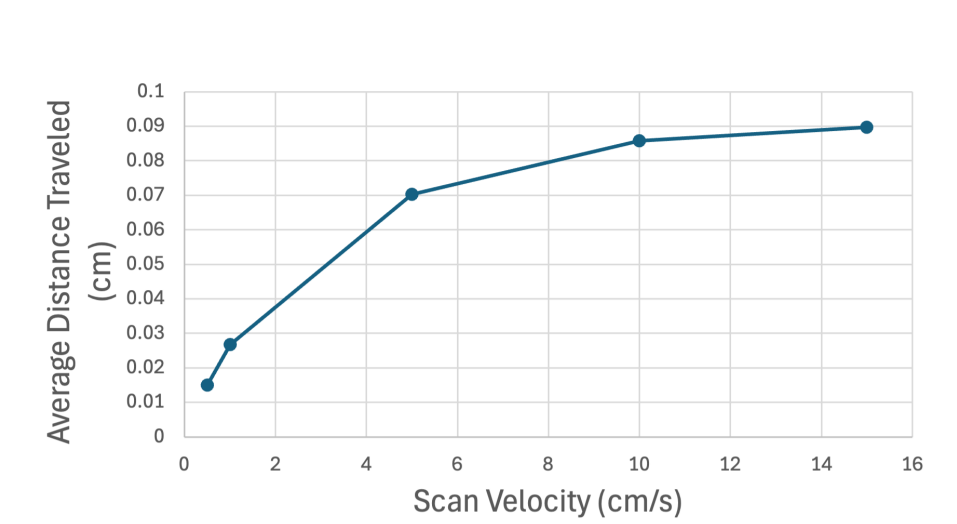


Figure 5.4: Effect of scan velocity on average robot travel distance between captured frames.

slower velocities, the surface appears smooth and uniform, whereas at higher velocities, the mesh exhibits noticeable deviations and surface irregularities.

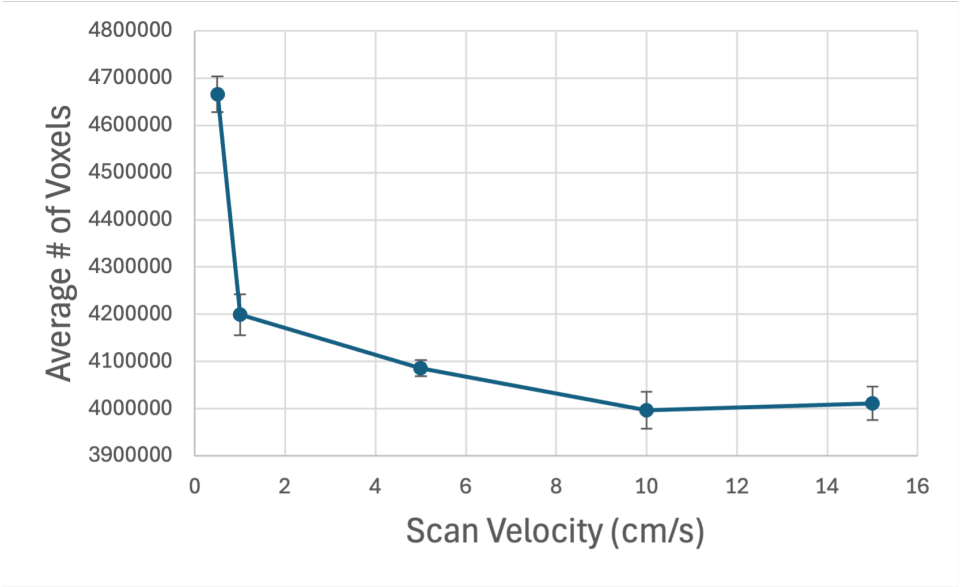


Figure 5.5: Effect of scan velocity on average voxel count.

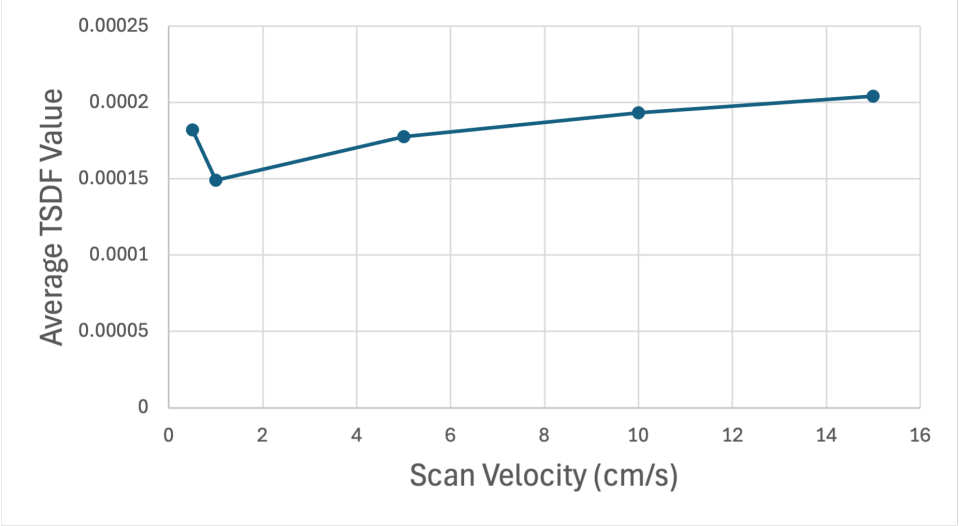


Figure 5.6: Effect of scan velocity on the average TSDF value.

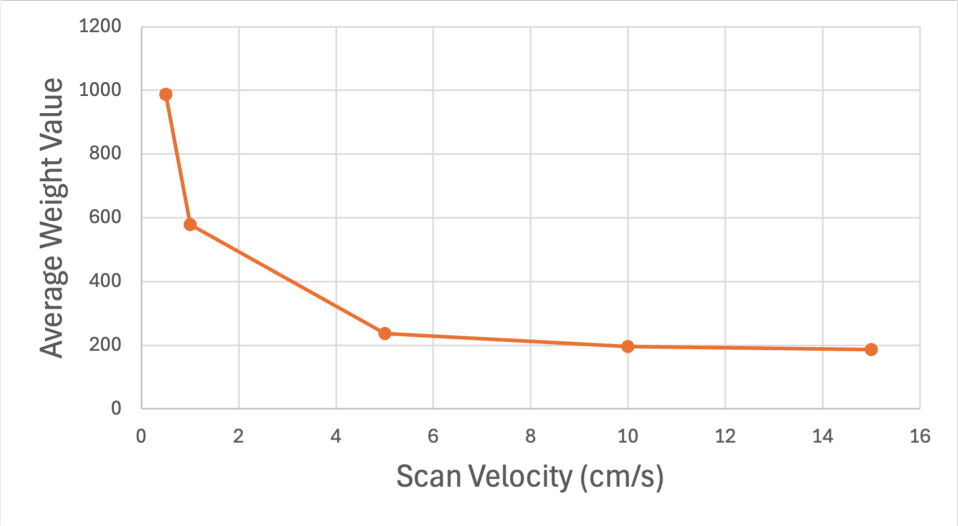
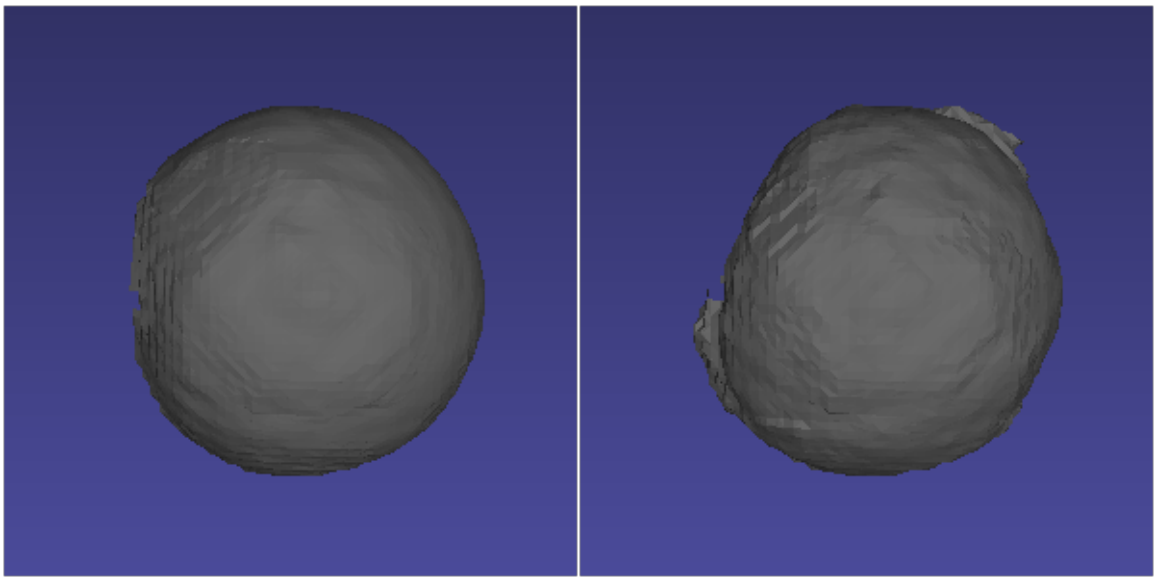


Figure 5.7: Effect of scan velocity on average weight value.



(a) 0.5 cm/s scan velocity **(b)** 15 cm/s scan velocity

Figure 5.8: Surface reconstruction of the golf ball artifact at slowest and fastest scan velocities.

5.2.2 Measurement Accuracy

The mesh data from each scan was used to create a best fit diameter which was then compared to the standard golf ball size of 42.67 mm. An absolute best fit error was recorded for each scan and averaged for each robot speed. The plotted errors can be seen in Figure 5.9. The overall trend shows an increased error with increased speed. At 10 cm/s, however, the average error decreased, but with the largest standard deviation. Given the overall trend of rising error with increasing velocity and the elevated standard deviation, the comparatively small error observed at 10 cm/s is likely not representative of the trend and is more likely to have been created from noise due to the increasing robot measurement speed. At 15 cm/s, the error increases significantly, although still only 1.3 mm. Along with the best-fit data, MeshLab was used as to calculate surface mesh deviation histograms. Since the method is slower than the best-fit calculation, the scan with the lowest best fit error was selected from each set of scan speed measurements. The comparisons for each scan velocity can be seen in Figures 5.10 through 5.14. The histogram in each figure represents the distribution of surface deviations for the corresponding scan. Positive values indicate a surface larger than, or outside, the control sphere while negative values indicate a surface below, or inside, the control sphere. As expected, the 0.5 cm/s velocity scan has the lowest deviation range with a minimum error of -1.04 mm and a maximum error of 0.884 mm. The slower speed histogram shows a bell curve distribution around the 0 indicating an accurate surface reconstruction. 15 cm/s has the highest deviation with a minimum value of -4.97 mm and a maximum of 2.91 mm. The faster speed histogram shows a small spike around the 0 error then a flatter distribution surrounding suggesting that while the mesh is reconstructed somewhat accurately there is increased noise and a decrease in precision during reconstruction.

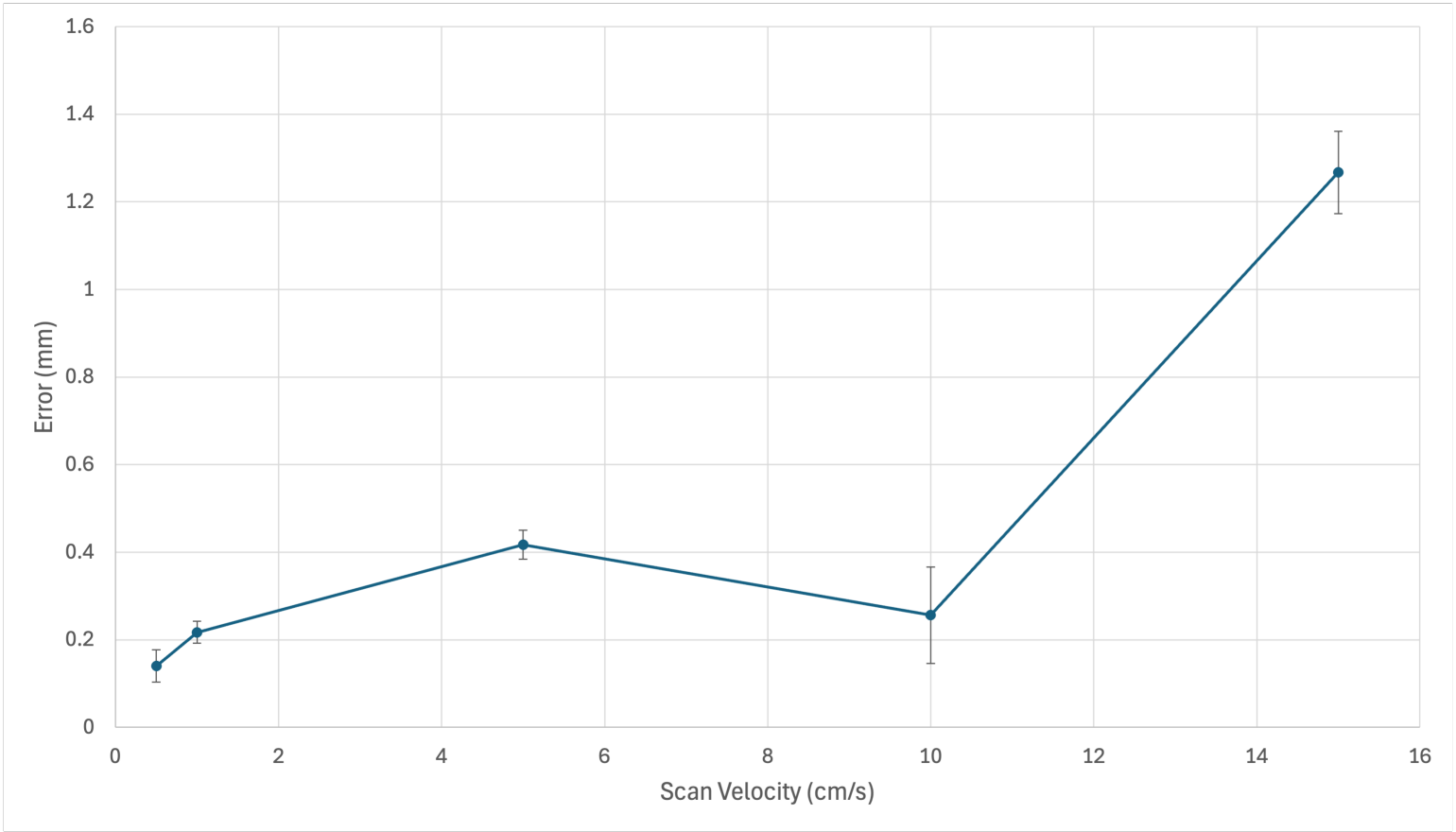


Figure 5.9: Effect of scan velocity on scan error.

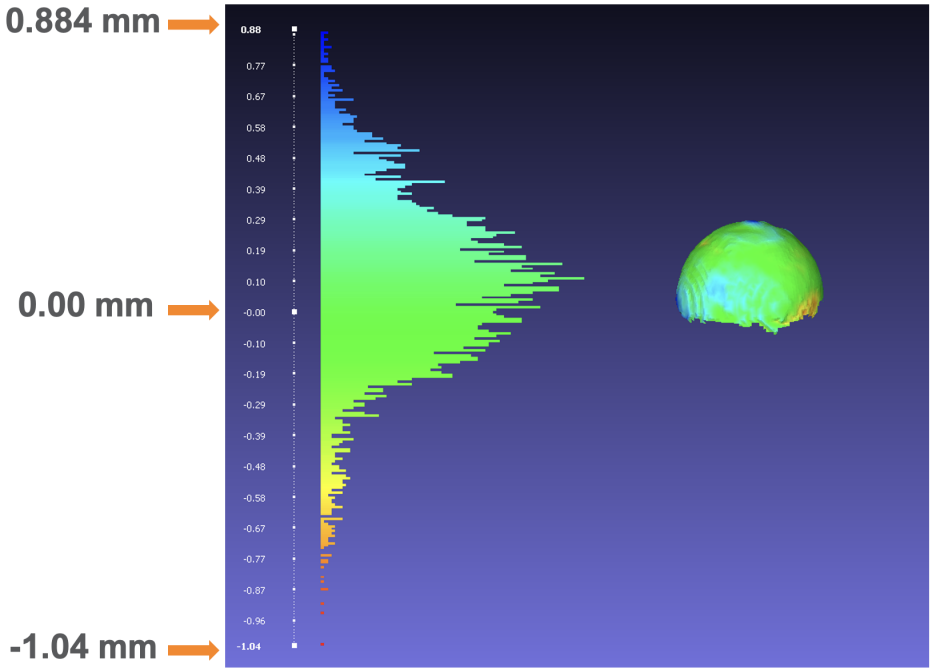


Figure 5.10: Surface deviation for the golf ball artifact at 0.5 cm/s scan velocity.

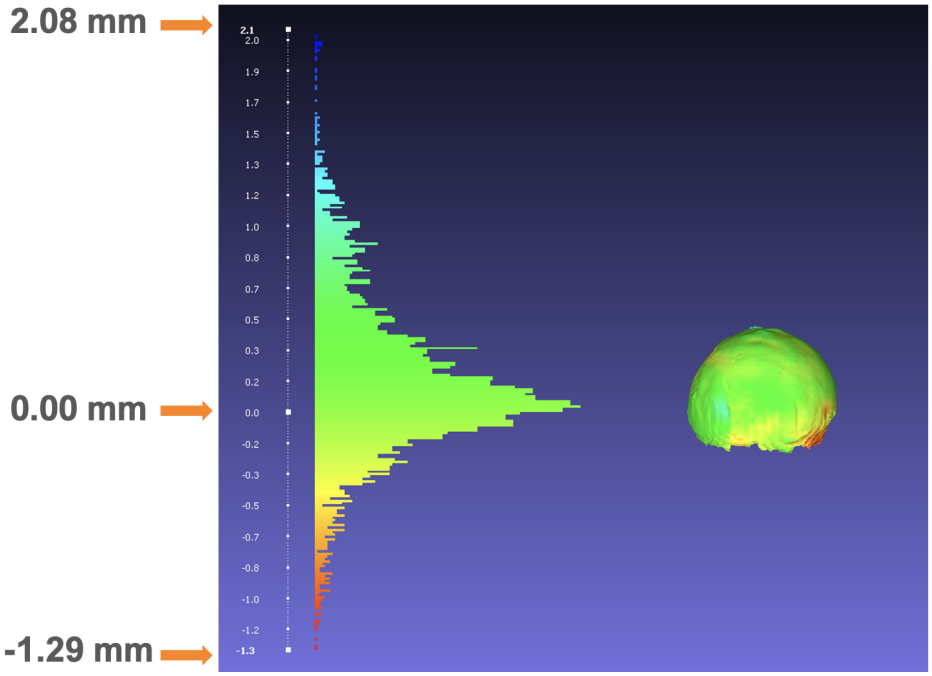


Figure 5.11: Surface deviation for the golf ball artifact at 1 cm/s scan velocity.

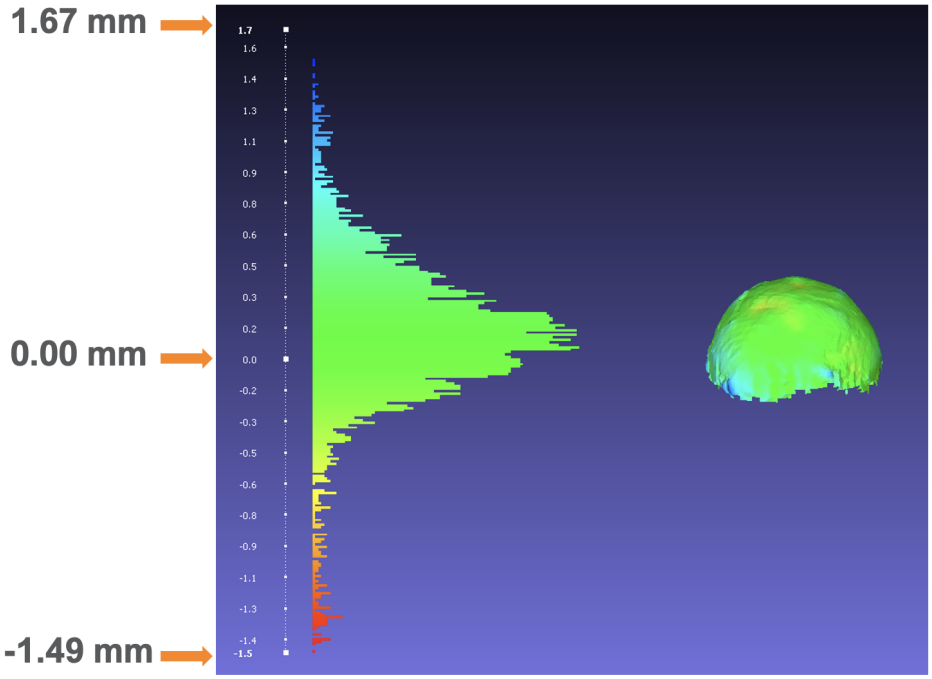


Figure 5.12: Surface deviation for the golf ball artifact at 5 cm/s scan velocity.

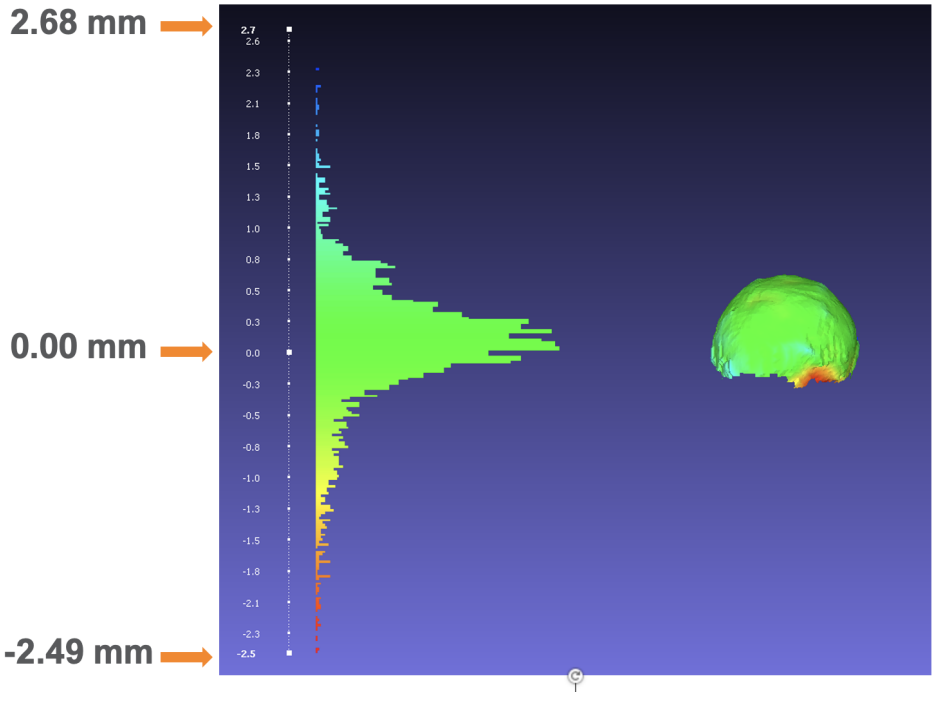


Figure 5.13: Surface deviation for the golf ball artifact at 10 cm/s scan velocity.

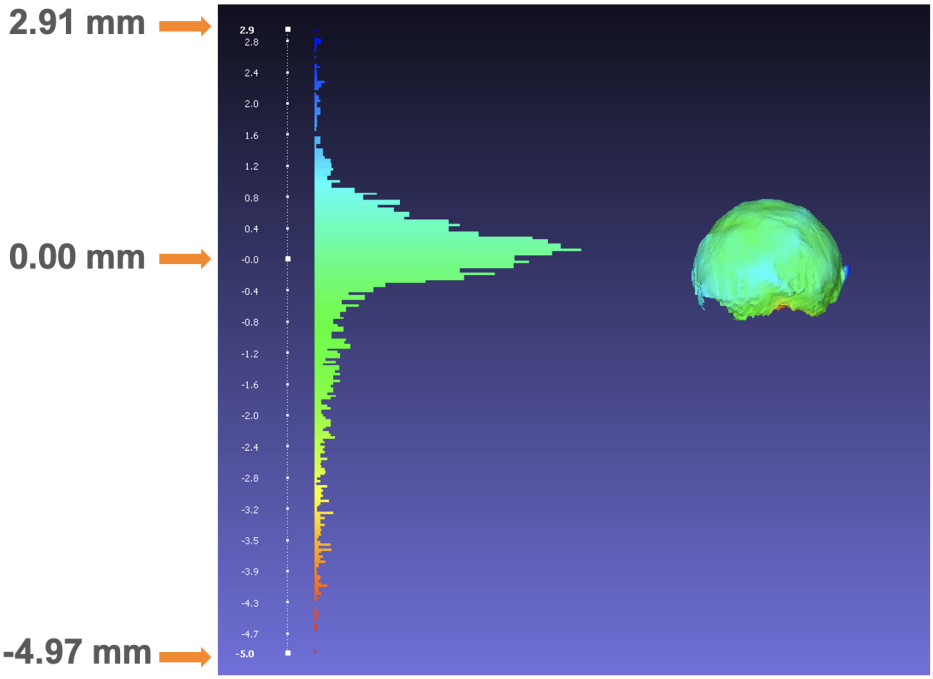


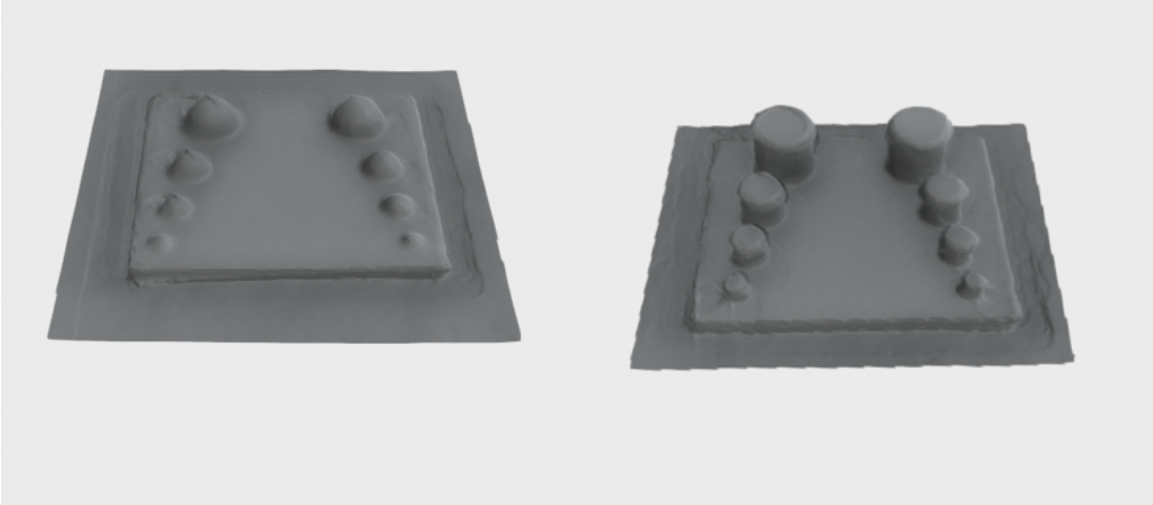
Figure 5.14: Surface deviation for the golf ball artifact at 15 cm/s scan velocity.

5.3 Primitive Artifacts

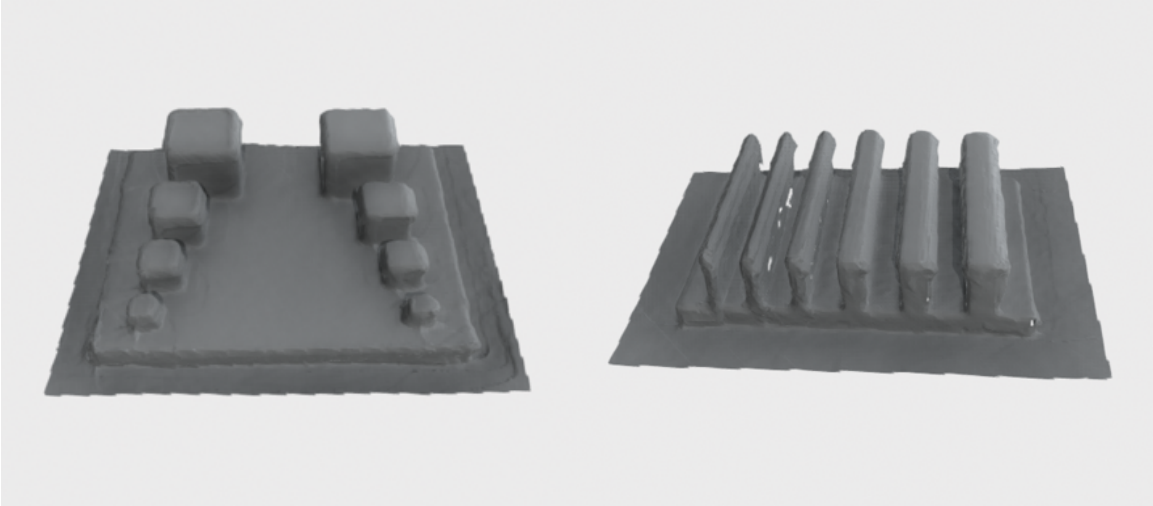
Each of the different primitive artifacts were scanned once at 1 cm/s using the scan path outlined in 4.1.2. The resulting meshes can be seen in Figure 5.15. Visual inspection of the meshes reveals deformations along the sharp edges of the artifacts. These edge deformations are particularly pronounced in the cube and fin artifacts, with the top edges exhibiting a bubble-like effect. The cylinder artifact demonstrates the bubbling effect on the top edge as well, but is not as prevalent. Sphere artifact meshes demonstrate some deforming towards the top and morphing into the base plate of the artifact. Overall distortion was minimal for sphere artifacts demonstrating a geometry which would be more accurately measured by the low-cost scanning system. After observing each of the primitive artifact's resulting surface meshes, a new design should include a full sphere instead of half sphere to reduce surface errors at the base plate interface.

5.4 Scanning During A WAAM Build

To test a more real-life scenario example, scans were collected during a WAAM process. The reconstructed mesh from the final layer of the WAAM build can be seen in Figure 5.16. Several key findings emerged from this experiment. Foremost, the welding lens installed on the camera for protection was found to have no significant impact on the quality of the scanned results. It was also demonstrated the system would remain in place during a WAAM process that was completed over several days. Finally, between layers 10 and 15, the weld table was slightly shifted by an operator. This displacement is evident when comparing the meshes, highlighting the potential for automated detection of distortions. Figure 5.17 shows the displacement between the two layers.



(a) Sphere Artifact Mesh **(b)** Cylinder Artifact Mesh



(c) Cube Artifact Mesh **(d)** Fin Artifact Mesh

Figure 5.15: Surface reconstruction of primitive artifacts.

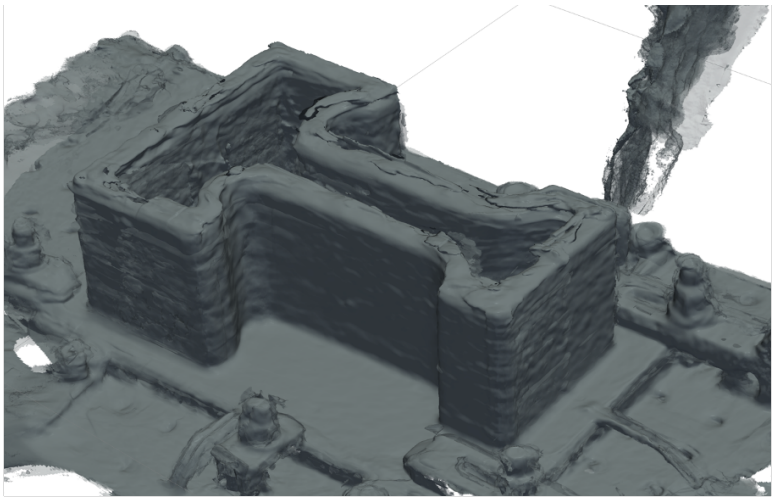
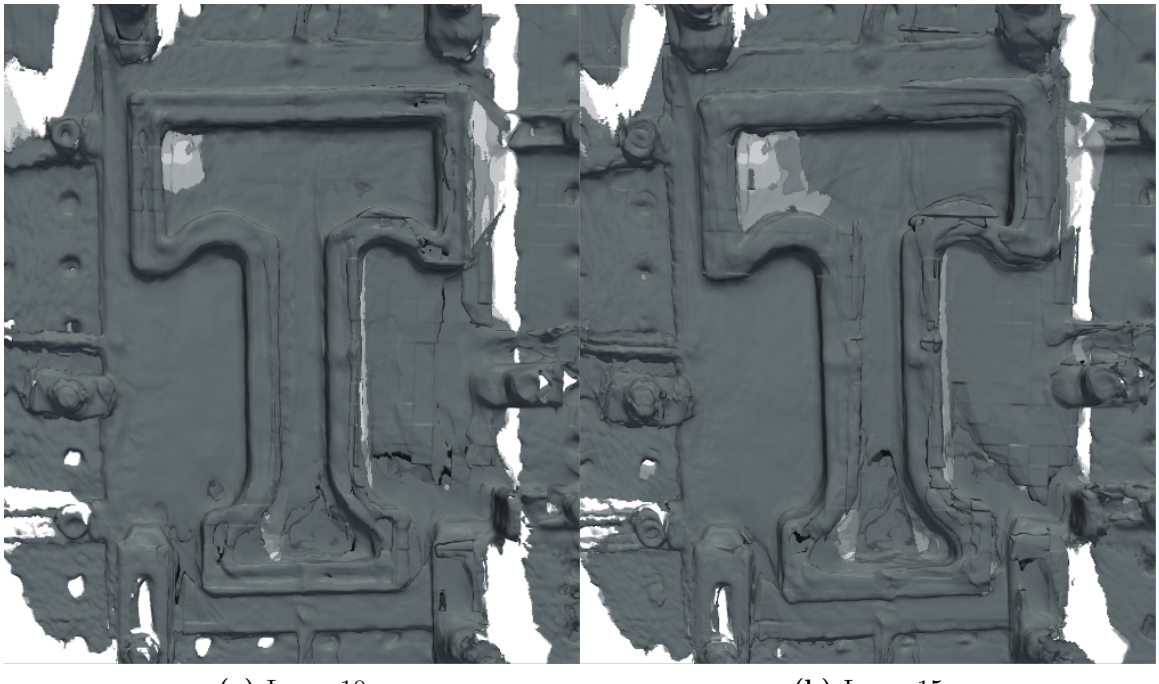


Figure 5.16: Surface reconstruction of the WAAM Power T part after printing.



(a) Layer 10 **(b)** Layer 15

Figure 5.17: Surface reconstruction of different print layers for the WAAM Power T.

Chapter 6

Conclusions

6.1 Conclusion

The performance of a low-cost scanning method has been evaluated for measuring part shapes during wire arc additive manufacturing. This work built on and contributed to previous work focused on developing a complete low-cost 3D scanning system. The system successful combined an Intel RealSense D405 camera, a KUKA KR6 robotic arm, Open3D's TSDF integration algorithm to generate meshes of WAAM parts. Robot position data and camera depth frames were aligned through time and spatial synchronization. Time synchronization utilized a flash-trigger to align camera frames with robot position. Spatial synchronization utilizes a checkerboard based extrinsic calibration to calibrate the camera as a TCP for the robot.

A series of performance experiments were conducted to assess the system's capability and limitations. A golf ball was used as a test artifact to determine effects of robot velocity on integration values and scanner error. Several custom test artifacts were designed, including spheres, cylinders, cubes, and fins, to evaluate effects of geometry shape and sizes on reconstruction quality

The results demonstrated the system was capable of creating meshes accurate to within 1 mm. Results also indicate scan velocity plays a significant role in quality of surface meshes. Although increasing scan velocity introduced more noise and error, it offers a trade-off by reducing computational load and overall processing time. When evaluating the effects of

part geometry and surface texture, low-texture parts exhibited poor scan quality, including incomplete depth data and the presence of holes. Sharp edges were also prone to distortion and a bubbling effect in the reconstructed meshes. While these represent limitations of the system, such issues are less important for WAAM application, as WAAM manufactured parts are inherently possess high surface texture due to layered deposition and rarely contain sharp edges. Consequently, the low-cost system remains a viable solution for 3D scanning and distortion monitoring of WAAM components.

To further assess real-world applications, the system was also tested on an active WAAM deposition process. Testing demonstrated the system can operate under realistic environmental conditions, capturing usable data without interfering with the process. The system demonstrate an ability to detect changes in the part during WAAM process. While more work is still necessary to refine the system, overall the findings support potential low-cost solution for in-situ geometry monitoring.

6.2 Future Work

Future work will continue to explore and quantify limitations of the system. Additional parameters, such as camera settings (including exposure and resolution) and environmental setting (such as lighting and robot path planning), will be investigated to better understand effects on scan quality and accuracy. While the artifacts used in these experiments provided valuable insight into system performance, further testing with a variety of artifacts is needed to determine other sources of error, such as feature to feature spacing error. Implementation of a real-time distortion detection is a key next step as results have shown changes in geometry can be detected with the system. Because the system is highly sensitive to calibration, efforts need to be focused on automating or improving calibration methods such as spatial calibration. Overall, efforts will focus on creating a more robust, user-friendly scanning system, suitable for reliable geometric monitoring of WAAM processes.

Bibliography

Bibliography

[1] Zachary C. Koller. Low-cost optical surface measurements for wire-arc additive manufacturing. Master's thesis, University of Tennessee, 2025. URL https://trace.tennessee.edu/utk_gradthes/14500. Master's Thesis. viii, 6, 8, 15, 19, 20

[2] Anders Grunnet-Jepsen, John Sweetser, Tri Khuong, Sergey Dorodnicov, Dave Tong, Ofir Mulla, Hila Eliyahu, and Evgeni Raikhel. Intel realsense self-calibration for d400 series depth cameras. Technical report, Intel Corporation, 2022. URL https://dev.intelrealsense.com. White Paper, Revision 2.7. viii, 16, 17

[3] Jake Dvorak. *Feasibility and Uncertainty Evaluation of Sequential Hybrid Manufacturing Using Optical Coordinate Metrology.* PhD thesis, University of Tennessee, 2024. URL https://trace.tennessee.edu/utk_graddiss/10451. Ph.D. Dissertation. ix, 6, 47

[4] Kun Li, Tianbao Yang, Na Gong, Jinzhou Wu, Xin Wu, David Z. Zhang, and Lawrence E. Murr. Additive manufacturing of ultra-high strength steels: A review. *Journal of Alloys and Compounds*, 965:171390, 2023. ISSN 0925-8388. doi: https://doi.org/10.1016/j.jallcom.2023.171390. URL https://www.sciencedirect.com/science/article/pii/S0925838823026932. 1

[5] K. Treutler and V. Wesling. The current state of research of wire arc additive manufacturing (waam): A review. *Applied Sciences*, 11(18):8619, 2021. doi: 10.3390/app11188619. 1

[6] Teodor Tóth and Jozef Živčák. A comparison of the outputs of 3d scanners. *Procedia Engineering*, 69:393–401, 2014. ISSN 1877-7058. doi: https://doi.org/10.1016/

j.proeng.2014.03.004. URL https://www.sciencedirect.com/science/article/pii/S1877705814002501. 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013. 3

[7] M. D. Barath Kumar and M. Manikandan. Assessment of process, parameters, residual stress mitigation, post treatments and finite element analysis simulations of wire arc additive manufacturing technique. *Metals and Materials International*, 28:54–111, 2022. doi: 10.1007/s12540-021-01015-5. 5

[8] C. Wacker et al. Geometry and distortion prediction of multiple layers for waam structures using artificial neural network. *Applied Sciences*, 11(10):4694, 2021. doi: 10.3390/app11104694. 5

[9] J. Lettori, C. Esposto, M. Peruzzini, et al. Geometrical characterization of circular multi-layered cmt waam specimens by 3d structured light scanning. *International Journal of Advanced Manufacturing Technology*, 136:5305–5334, 2025. doi: 10.1007/s00170-025-15107-8. URL https://doi.org/10.1007/s00170-025-15107-8. 5

[10] Shangyong Tang, Guilan Wang, and Haiou Zhang. In situ 3d monitoring and control of geometric signatures in wire and arc additive manufacturing. *Surface Topography: Metrology and Properties*, 7(2):025013, 2019. doi: 10.1088/2051-672X/ab1c98. URL https://doi.org/10.1088/2051-672X/ab1c98. 6

[11] Filipa G. Cunha, Telmo G. Santos, and José Xavier. In situ monitoring of wire and arc additive manufacturing by digital image correlation: a case study. *Procedia Structural Integrity*, 37:33–40, 2022. ISSN 2452-3216. doi: https://doi.org/10.1016/j.prostr.2022.01.056. URL https://www.sciencedirect.com/science/article/pii/S2452321622000646. ICSI 2021 The 4th International Conference on Structural Integrity. 6

[12] Mrinall Umasudhan. Image depth estimation using stereo vision, 10 2022. 10

[13] Intel Corporation. Intel realsense d400 series datasheet. https://www.realsenseai.com/wp-content/uploads/2023/03/

Intel-RealSense-D400-Series-Datasheet-March-2023.pdf, 2023. Accessed: 2025-10-18, Intel Corporation, March 2023. 10, 16

[14] Intel Corporation. Intel® RealSense™ Depth Camera D405, 2022. URL https://www.intelrealsense.com/depth-camera-d405/. Accessed: 2025-10-18. 11

[15] Intel Corporation. Intel realsense sdk 2.0 python wrapper, 2023. URL https://github.com/IntelRealSense/librealsense. 13

[16] Gary Bradski et al. Opencv library, 2000–2023. URL https://opencv.org/. 13

[17] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing, 2018. URL https://arxiv.org/abs/1801.09847. 13

[18] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 303–312, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917464. doi: 10.1145/237170.237269. URL https://doi.org/10.1145/237170.237269. 14

[19] Open3D Contributors. Voxel block grid — open3d documentation. https://www.open3d.org/docs/latest/tutorial/t_reconstruction_system/voxel_block_grid.html, 2025. Accessed: 2025-11-17. 15

[20] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, August 1987. ISSN 0097-8930. doi: 10.1145/37402.37422. URL https://doi.org/10.1145/37402.37422. 15

[21] United States Golf Association (USGA). Equipment rules — rule 4, part 4: Equipment standards, 2019. URL https://www.usga.org/equipment-standards/equipment-rules-2019/equipment-rules/equipment-rules.html#!ruletype=er&section=rule&partnum=4&rulenum=4. Accessed: 2025-10-15. 32

[22] Fei Li, Yunfan Du, and Rujie Liu. Truncated signed distance function volume integration based on voxel-level optimization for 3d reconstruction. In *IS&T International Symposium on Electronic Imaging 2016: 3D Image Processing, Measurement, and Applications*, pages 3DIPM–398.1–3DIPM–398.6, 2016. doi: 10.2352/ISSN.2470-1173.2016.21.3DIPM-398. 43

[23] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh processing tool. In *European Interdisciplinary Cybersecurity Conference*, 2008. URL https://api.semanticscholar.org/CorpusID:1866174. 43

Appendix

A Example Camera TCP Calculation with Numbers

Assume the checkerboard pose relative to the camera, $\mathbf{T}_{\text{cb\_to\_cam}}$, is obtained from OpenCV with a 90° rotation about the z-axis, $\mathbf{R}_{\text{cb\_to\_cam}}$ and a translation vector *tvec*.

$$R_{\text{cb\_to\_cam}} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad tvec = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \end{bmatrix}$$

$$T_{\text{cb\_to\_cam}} = \begin{bmatrix} 0 & -1 & 0 & 0.1 \\ 1 & 0 & 0 & 0.2 \\ 0 & 0 & 1 & 0.3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Invert the matrix:

$$R_{\text{cam\_to\_cb}} = R_{\text{cb\_to\_cam}}^{T} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{t}_{\text{cam\_to\_cb}} = -\mathbf{R}_{\text{cam\_to\_cb}} * tvec = - \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \end{bmatrix} = \begin{bmatrix} -0.2 \\ 0.1 \\ -0.3 \end{bmatrix}$$

$$T_{\text{cam\_to\_cb}} = \begin{bmatrix} 0 & 1 & 0 & -0.2 \\ -1 & 0 & 0 & 0.1 \\ 0 & 0 & 1 & -0.3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Next, assume the robot wrist pose in the checkerboard frame is

$$T_{\text{wrist\_to\_cb}} = \begin{bmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverting this transform:

$$R_{\text{cb\_to\_wrist}} = R^T_{\text{wrist\_to\_cb}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$t_{\text{cb\_to\_wrist}} = -R_{\text{cb\_to\_wrist}} * t_{\text{wrist}} = - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0 \\ 0 \end{bmatrix}$$

$$T_{\text{cb\_to\_wrist}} = \begin{bmatrix} 1 & 0 & 0 & -0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, multiplying the transforms cancels the checkerboard frame:

$$T_{\text{cam\_to\_wrist}} = T_{\text{cb\_to\_wrist}} * T_{\text{cam\_to\_cb}}$$

$$T_{\text{cam\_to\_wrist}} = \begin{bmatrix} 0 & 1 & 0 & -0.7 \\ -1 & 0 & 0 & 0.1 \\ 0 & 0 & 1 & -0.3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This final matrix represents the camera pose relative to the robot wrist, i.e., the tool center point (TCP) of the camera.

Vita

Anukkah Burleson earned her Bachelor of Science in Mechanical Engineering from the University of Tennessee in 2024, graduating Magna Cum Laude. Currently, she is pursuing a Master of Science in Mechanical Engineering at the same institution. During her undergraduate studies, she completed internships with Navus Automation and Hirebotics, gaining hands-on experience in automation and robotics. Upon completion of her graduate studies, Anukkah intends to continue driving innovation in robotics and advanced manufacturing through a professional career in the field.