ENHANCED CORROSION ANALYSIS ON CURVED STEEL SHEETS USING FREEFORM ROBOTIC ECA

By

Ciaron Nathan Hamilton

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical and Computer Engineering—Doctor of Philosophy

2024

ABSTRACT

Nondestructive Evaluation (NDE) 4.0 is an emerging approach for providing automation towards material inspection using innovative techniques from Industry 4.0. Such innovative approaches allow for vast data acquisition and analysis potential for physical component assessments that require inspection, or else risk structural failure. Inspection for conductive materials is possible from surface scanning procedures, such as Eddy current testing (ECT). ECT utilizes electromagnetic induction to find defects in conductive materials. In the case of this dissertation, corrosion may be detected with ECT before it continues to grow and damage larger components. Corrosion is “the cancer” of metallics, costing billions in irreversible damages annually. In some instances, corrosion may occur under paints, which may be near invisible through visual inspection. ECT in place may be used, however many components need fast and robust scanning procedures. Fast scanning can be enabled with Eddy current arrays (ECAs), allowing repeated coils that may be used to increase scan areas or cut down scan times, a procedure like a paint brush that obtains information about the material’s health. ECAs also allow for different configurations that may be beneficial for data analysis, such as differential scanning mode. Inspection may be automated using robotic arms systems equipped with ECA, allowing for fast, repeatable, and robust scanning. This may be useful in situations with large components that may be brought into a "robot arm sensor wash" system, such as automobiles or military vehicles. One barrier for enabling robust “freeform” scanning is obtaining the scan path which the ECA will glide along, as components may come in different shapes and sizes, sometimes with curved or complex geometries. The focus of this dissertation is to provide NDE 4.0 techniques along with ECA to detect corrosion along curved steel sheets. NDE 4.0 techniques show capabilities merging cyber-physical systems (CPS), computer vision, and the concept of digital twins between physical and digital space. To enable NDE 4.0 for robotic inspection, a framework was developed, which has five major steps: obtain a reconstruction of the physical object and surrounding environment, orient this virtual scene with respect to the robot’s base frame, generate a toolpath which the NDE probe will be manipulated, conduct the ECA scan with 6-degrees-of-freedom (6-DOF), and process the NDE results. A novel algorithm was developed, “ray-triangle intersection arrays,” which enables pathing on meshes from a raster pattern. The framework used was designed to be generalized for any surface scanning probe, in which ultrasonic testing (UT) scanning for carbon fiber inspection is also demonstrated using the same framework. For ECA, it is important to keep the probe close to the surface while ensuring the distance between the sensor and the probe, or lift-off, is minimized. For the scale of the defects obtained, which is approximately 0.05mm in depth at max, otherwise minor tilts of the probe become significant. The ECA probe contains 32 channels and was operated at 500khz using absolute mode scanning, allowing for exceedingly small defect depths to be detected. The effects of ECA scanning using a robot system are examined, showing that tilt errors from either the path-planning procedure or even the calibration or the robot will provide significant errors. To better understand the effects per coil, a “full” scan mode was examined, showing a larger image per coil, as well as the typical painting scan considered as a “fast” scan. Other errors such as heating were also examined. With knowledge of the errors from robotic scanning, post-processing procedures were developed to minimize errors. A novel algorithm “array subtraction” was developed to reduce lift-off from common factors seen in every coil, indicating prob tilt error. A digital microscope was used to compare defects ground-truth defect volume with the ECA results, in which defect versus background intersection masking was used. Three objectives are discussed to cover the generalized robust surface scanning framework, the dissection of effects of robotic scanning for ECA for corroded surfaces, and how to process and interpret this ECA data. The results show promising future applications for robust surface scanning as corrosion is decently detected. Future applications would be the previously mentioned carwash system, AI-enabled detection, and mobile platforms to expand on inspection workspaces.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS vi

CHAPTER 1 INTRODUCTION 1
- 1.1 Purpose of NDE Robotic Inspection for Complex Surfaces 3
- 1.2 NDE 4.0 Framework 8
- 1.3 Objectives 26
- 1.4 Contributions 27
- 1.5 Literature Review 28

CHAPTER 2 THEORY OF PHYSICAL AND VIRTUAL SYSTEM INTERFACING FOR NDE ROBOTICS 38
- 2.1 Physical Environments to Digital Space: Analysis 38
- 2.2 Digital Operations for NDE End-Effector Positioning 42
- 2.3 NDE Block, Point cloud, and Mesh 56

CHAPTER 3 ENVIRONMENT RECONSTRUCTION 59
- 3.1 Introduction 59
- 3.2 Mesh Reconstruction and Post-Processing 65
- 3.3 Reconstruction Methods and Devices 76

CHAPTER 4 ENVIRONMENT REGISTRATION 82
- 4.1 Point Pairs Picking 87
- 4.2 Center Point Transformation 95

CHAPTER 5 TOOLPATH GENERATION 99
- 5.1 Basic Generation Algorithms 99
- 5.2 Ray-Triangle Intersection Arrays 108
- 5.3 SprutCAM Toolpath Generation 121

CHAPTER 6 ECA SCANNING AND POST-PROCESSING 123
- 6.1 Errors from Complex Robotic Scanning 123
- 6.2 Scanning Procedures to Examine Errors 128
- 6.3 Post Processing 128

CHAPTER 7 EXPERIMENTAL RESULTS 142
- 7.1 Curved Steel Corrosion Detection 144
- 7.2 Voltage to Depth 172
- 7.3 Ultrasonic Testing on Carbon Fiber Reinforced Polymer 176

CHAPTER 8 CONCLUSION 180
- 8.1 Objective Testing 180
- 8.2 Limitations 183
- 8.3 Future Works 185

BIBLIOGRAPHY 187

APPENDIX 201

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CAD	Computer-Aided Design
CPS	Cyber-Physical System
CPT	Center Point Transformation
CFRP	Carbon Fiber Reinforced Polymer
DAQ	Data Acquisition
DH	Denavit–Hartenberg
DOF	Degrees-Of-Freedom
EC	Eddy Current
ECA	Eddy Current Array
ECT	Eddy Current Testing
FOV	Field Of View
iNDT&E	Intelligent nondestructive testing and evaluation
ICP	Iterative Closest Point
MUT	Material Under Test
NDE	Nondestructive Evaluation
PPP	Point Pairs Picking
PTC	PoinT Cloud
px	PiXels
SF	Scalar Field
SOR	Statistical Outlier Removal
SNR	Signal-to-Noise Ratio
RGB	Red Green Blue
RDK	RoboDK
RMS	Root Mean Square
UT	Ultrasonic Testing

SL Structured Light

SLAM Simultaneous Localization And Mapping

TSP Traveling Salesmen Problem

VHX Digital Microscope (VHX-7000)

XCOR CROSS CORrelation

CHAPTER 1

INTRODUCTION

Nondestructive evaluation (NDE) is the field of study about the determination of properties from a material or structure without resorting to modification or harm to the specimen under test [79]. NDE applications are important for deciding significant damages or deterioration on a material under test (MUT) which may cause a catastrophic event to the overall system if not detected and dealt with. For example, hydro turbines are widely used for power generation and come in different shapes and sizes [177] but are susceptible to various damages to be examined [52]. Such defects that may cause system failure may be difficult to impossible from conventional methods. If a person visually inspects turbines within the plant, they may see a corroding surface within only their eyesight. The issue comes from damage that may not be clearly seen, such as sub-surface defects or bonding. Failure to resolve a damaged turbine may create down times at a cost to the facility. At worst, scenarios may arise that may create hazards for operation and maintenance personnel. Another example is from airlines, which have suffered catastrophic events due to component failures [41]. These disasters could be prevented through NDE, which may be used to decide the location and severity of damage within critical materials. NDE may be used to obtain the current state of a component, which when compared with its lifespan, may give a prediction of when to replace a part through prognostics. Another example of difficult but important to inspect regions are from point plant vessels, which are not OSHA compliant [24] and result in hazardous operation for human inspectors. Because of the importance of inspection and the difficulty of using humans to inspect critical materials in many scenarios, robotics may be used in place. Robots allow for remote or automated inspection of difficult to inspect locations and may be useful as an alternative to human inspection.

Corrosion is "the cancer" of metallics which results in billions of dollars' worth of damages annually [78][18]. Awareness enabled by NDE may be used to reduce these damages [78][140][128]. NDE may be used to find corrosion in locations that are not obvious to a human inspector or would otherwise be difficult or impossible for a human inspector to have access to. Corrosion may occur on large complex surfaces for a selection of different fields, for applications not just limited to army, navy, aerospace, power plants, and pipelines [156, 70, 23, 129]. As corrosion occurs on metallic surfaces, which are conductive, methods such as Eddy current testing (ECT) may be used to detect corrosions, even ones that are small in area, $1 \times 1mm^2$ or larger, and depths of at least $10\mu m$ at higher frequencies such as $2Mhz$. The purpose of this research is to integrate NDE and industry 4.0 techniques through cyber-physical systems (CPS) methods to achieve NDE 4.0, allowing for autonomous inspection when applied with robotics. The methodology focuses on reconstructing a mesh, in which novel path-planning procedures are discussed. Through the methodology, corrosion detection may be enabled with ECT with special post processing required. The samples used are curved for both concave and convex scanning setups. An Eddy current array (ECA) is used, which contains several different channels utilized to speed up scans and increase scan coverage per time. Unique processing techniques are demonstrated to eliminate certain issues such as lift-off.

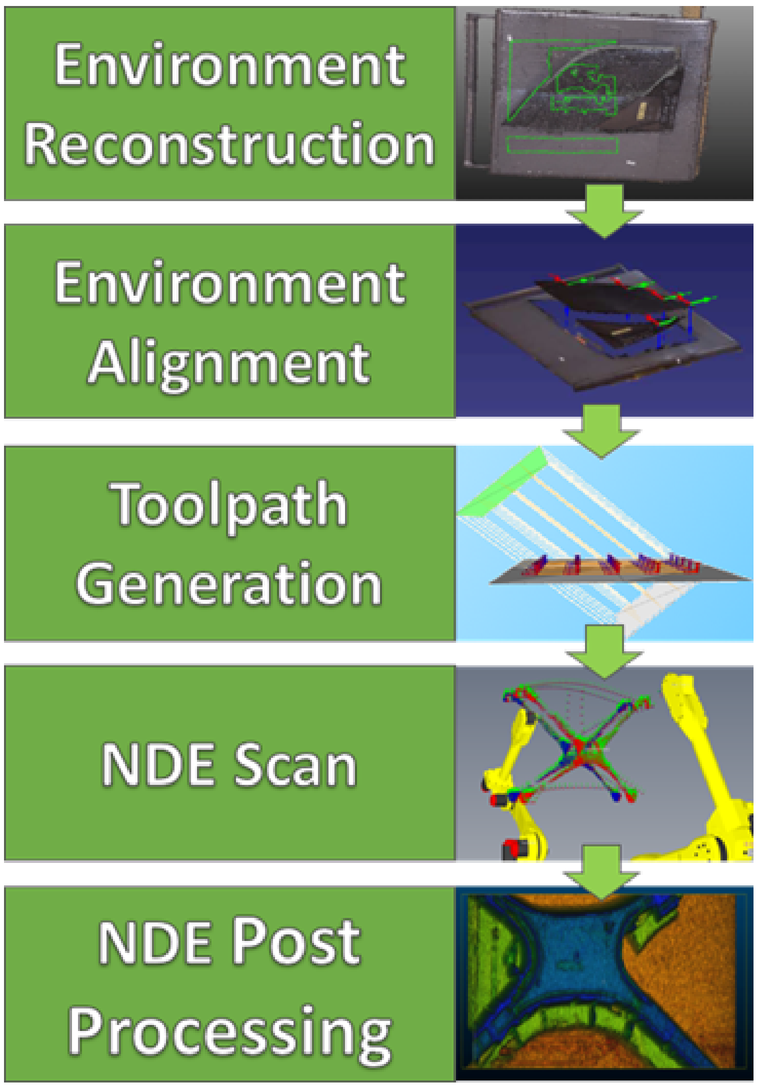


Figure 1.1 Proposed robot arm NDE framework in NDE 4.0, showing reconstruction, alignment, generation, scanning, and post-processing on carbon fiber samples.

1.1 Purpose of NDE Robotic Inspection for Complex Surfaces

An important yet overlooked procedure is the examination of “complex-shaped” objects, which requires an actuation system capable of moving along its surface. Using CPS to visualize an actuation system’s workspace allows for such scanning by reconstructing the surface and parsing a toolpath to move along. A “complex-shaped” object is defined for this thesis as an object with non-planar or non-flat surfaces. Topologies may have properties such as surface curvatures that are concave or convex, gaps or through-holes in the material, or steps or depth displacements on the area to scan. The surface shape affects two important qualities of NDE data using surface scanning probes:

1. Lift-offs (a.k.a., stand-off distances) between the examination probe and the surface, which the probe needs to be both close to the surface of the object to the scan and/or constant between sensor positions.

2. The localization of data, or its position on data obtained. Both lift-offs and localization affect the quality of scans as well as precise knowledge of the damage, such as location, classification, severity, etc...

If the sensor does not follow the surface properly, then data quality of defects may be poor or even nonexistent. Variations in lift-off tend to bias results for ECT methods [146] and even for acoustic testing [183] which has less lift-off error requirements in comparison.

Before applying the more complex techniques of CPS, similar NDE applications should be discussed. In practical applications, handheld systems used by human operators may be used. In research applications, a sensor held on a Cartesian linear actuation system such as a gantry may be used. There are several advantages as well as issues for both. Human operators have the benefit perception, classification, and decision making [189] [45]. These benefits are only natural for human conductors, while CPS needs complex architectures and algorithms required to apply and simulate similar actions. In simple terms, a human operator should know exactly how and where to reach an object to scan with the sensor. In contrast, an automated system requires a physical-to-virtual correspondence, or "digital twin" of the geometry and classification of components based on the mesh, an algorithm to apply path planning to move the sensor, then place the sensor while acquiring data. A further albeit abstract benefit is that human operators also may understand and potentially apply unique operations for specific conditions, which a computer itself could only achieve depending on complexity of artificial intelligence, potentially up to sentient levels. For example, a car door needs to be scanned, which is not perfectly flat and holds at least two materials: metal shaping most of the piece and glass covering the window. The area around the metal should be scanned, and the glass should be avoided. A human may simply understand this and move the sensor along the metal; however, an automated system may not at once understand this and may risk shattering through the glass if not seen inside a cyber-physical environment. Another example could be a pipe on a ceiling above the workspace of a robotic arm. This assumes there is a valid solution such as blocks to climb and is not bounded by wires or weight nor having unique script designed for this scenario. The robotic arm without near-sentient capabilities may never find a solution to reach the sensor to this pipe, while a human may just use a ladder to reach the pipe. In a sense, robots and NDE are techniques used as tools for their needed applications, which proves this concept as well. While humans have analysis skills, certain objective information cannot be obtained easily. Human operators cannot reliably obtain the position of the sensor constantly without aid, which actuation systems should already be capable of doing through forward kinematics. This means that data localization becomes subjective, by being reliant on the human's perspective and time of NDE data acquisition. Fortunately, there are algorithms that may collaborate with human-operated handheld systems that help localize data without physical encoder information, such as simultaneous localization and mapping (SLAM) [189], which can visually encode data information. The downside of these algorithms is that they are based on approximations from data topology rather than more concrete information from robot systems with high precision. Another issue is the limited workspace or area a human operator may conduct scanning in, either due to size, whether being too large or high to reach or too small to fit inside, or environmental hazards, such as slipping or height, intense heat, or chemical-related hazards. As mentioned before, a human conductor may find unique solutions for each, but such may increase risks, such as risk of injury or death due to falling from a high height such as a bridge. One other issue is redundant scanning, which automated actuation systems outclass human inspectors in both time of operation, resolution, and repeatability. Therefore, while humans surpass automation in intelligence of subjective matters, which allow for broader solutions, automation allows for objectivity with narrow solutions that may be difficult or impossible for human conductors. Finally, due to many of the concerns just listed, NDE inspectors need to be highly qualified, requiring expert knowledge on how to properly conduct NDE inspection.

Gantries, or other systems which use linear actuation along a Cartesian axis or rail, may provide redundant scanning, parse position information, and synchronize it with concurrent NDE data. Gantries are very stable and can move with high accuracy typically within a fraction of a millimeter. Usually, gantries are used in research settings to scan flat "coupon" samples to parse NDE data without complicated setups, such as reconstructing the sample in virtual space to set up a digital twin for path planning and NDE data. A simplified setup is to place a sample either flat or along a plane of the gantry, measure the area of the coupon, place the sensor close to the sample, keeping low lift-offs and lift-off variation, and run a raster scan along the piece to obtain a discrete scan along the sample. The issue comes when complex-surface components require scanning, such as vehicular pieces, such as hoods or doors, or aerospace pieces such as jet-engine turbines. These pieces have curved structures which will inevitably lead to unacceptable lift-off variations if scanning only on a 2D plane. Even with coupon samples, there may be uncertainties in liftoffs due to the sample not being completely flat, either due to placement or if the sample's structure is slightly curved or has surface roughness. Depending on the method used, such as ECT, which typically requires low lift-offs and variations, this may not be acceptable and lead to large data uncertainties. Some methods allow for large lift-offs due, such as thermal imaging and radiography, which even allow for portability [167][165]. However, these methods have limitations, such as surface only evaluation for thermal imaging, and dependency on orientation/thickness as well as radiation hazards for radiography [79]. Gantries with a 3D Cartesian setup may scan complex samples. This requires knowledge of the geometry of the sample, either reconstructed or pre-defined with measured placement in the workspace. However, these systems will have a constant rotation of the probe or a "flexible" probe, meaning that maintaining the probe normal to a complex-surface object is particularly challenging. A gantry may become capable of dealing with this rotation issue by adding two consecutive servo or similar motors on the end-effector, which may be suggested if a gantry must be used.

Because of the limiting factors from human operators, and the lack of complexity for gantries, a system was developed that would allow for scanning complex-surface objects within its workspace.

This system is capable of autonomously scanning complex-surface objects with a “black box” surface scanning probe in mind, meaning any NDE probe that requires being moved along the surface may be used along with the developed NDE 4.0 framework. The novelty of this system is that it uses a structured light sensor as a reconstruction device to obtain the surface profile of the object under test, align the virtual environment with the robot’s base frame, create a path for scanning, then conduct an NDE scan with 6-degrees-of-freedom (DOF) complexity. The system will show capabilities through ECT for corrosion detection. Ultrasonic testing (UT) will also be briefly shown in support of the framework provided, shown in figure 1.1.

Translating and rotating an NDE probe with the closest approximation to a surface, with a constant lift-off with low variations between different sensor transformations, should give high quality NDE data. This may also provide the best localization of the NDE data within space. Visual encoding methods are not examined since physical encoders should supply more objective knowledge of position than SLAM, which requires virtual approximation techniques from each mesh. Actuation systems can throughput an end-effector transformation, which can be synchronized with the NDE data of a sensor as the end-effector at the same time or interpolated between positions. Inaccuracy of the transformation of the probe affects both NDE data quality (variable and non-close lift-offs), and inaccuracy of known location of probe will mean inaccuracy of NDE localization. Hence high accuracy methods for placing waypoints, requiring high accuracy of a virtual surface of the physical object, and high accuracy actuation systems, are needed.

As for applications, different scanning systems are suggested that may use the technology provided in this research. One would be robot arm systems, which will be the system focused on due to ease of confirming position accuracy using an industrial Fanuc 100ib robotic arm. It is expected that this system would work as a “robotic car wash” that finds defects towards ongoing materials and potentially fixes them on the spot. For example, an item for inspection may be rolled into the robotic NDE space for inspection, such as an automobile, and aircraft, or military vehicles, either a singular component, multiple components, or even the entire vehicle for corrosion. Another adjacent example, irregardless of corrosion detection but for based defect inspection, may be from a pipeline in which the component is manufactured. In both cases, the robots automatically detect the surface profile of the vehicle and determine the scan path. Classification of components, such as the difference between a car door frame and the window, is ignored for now but is an important consideration. Other scanning systems would be an unmanned vehicular platform and an unmanned drone platform, both holding the sensor and potentially a light-weight robot arm or chained servo motor system for actuation. These would increase the robotic workspace compared to a stationary robotic arm by allowing dynamic movement of the robot's base frame. For drone platforms, they may be used to fly to hard-to-reach areas of interest, such as bolts on a bridge that would be risky for a human inspector to reach. Though these two methods are for future work, the same NDE 4.0 framework used in this thesis is suggested, as the only changes are the movement of the robot frame and requirements for mobility.

1.2 NDE 4.0 Framework

The proposed NDE 4.0 framework consists of five stages, as shown in figure 1.1: environment reconstruction, environment alignment, toolpath generation, NDE scan, and NDE post-processing. The goal is to perform each step with minimal human interaction for automation. There are three classifications for automation considered:

1. Currently automated

2. Theoretically automatable but not currently implemented for the developed system

3. Not automatable (requires human interaction)

Each phase will be explained in more detail across this dissertation, but this section is to explain the pipeline. For point cloud (PTC) and mesh processing, including reconstruction, alignment, and other procedures to clean geometric data, CloudCompare and MeshLab are used. For robotic simulation and inverse kinematics solutions, RoboDk (RDK) is used. Computer-aided design (CAD) models to scan along are specifically avoided, due to strict requirements on the location of where to scan which is difficult to objectify without rigorous measurements of the environment to ensure alignment. That is not to say CAD models are not useful for actuation, there has been work dedicated to registration methods with CAD models [81]. For lift-offs, the ECA should be consistently $1mm$ away from the surface with minimized tilt. Tilting along the probe will add variability in lift-off per coil on the array which is unwanted. Both may easily be miscalculated by a bad rotation or translation measurement between physical and digital environments, hence the importance of physical to virtual registration. CAD models are only local to their defined digital space and require measurement to achieve translation back to physical space that is prone to error. It is also considered that in practice, samples would want to move seamlessly into the robot environment without interruption on manual measurements with respect to the robot's base frame, hence the importance of automation.

The input of this framework starts from a reconstruction device, which enables a digital PTC profile correlated from the physical topology of the MUT or sample under test. This is considered a cyber environment. A cyber environment for the NDE 4.0 framework should hold at least the region of the sample under test which the NDE scan should be conducted. There may be regions of objects that are not a part of the scan but are picked up by the reconstruction device, these are considered as background components. For usage of the cyber environment with the toolpath generator, there are two major steps needed for processing the PTC: reconstruction and alignment. Reconstruction refers to forming a mesh, a linkage of vertices and faces from the PTC to obtain a surface profile. The accuracy of this reconstruction, heavily dependent on the accuracy of the input PTCs to the physical surfaces, decides the accuracy of the surface profile. Many dedicated reconstruction devices, such as Intel RealSense for stereo vision, or the Creality CR-01 structured light scanner, supply a choice for outputting to either mesh or PTCs. On top of this benefit, they also provide 2D depth field from one snapshot, and may also fuse multiple snapshots to increase the range of reconstruction. These meshes require alignment, as there is no reference from the raw data to coordinate back to the robot itself. Another method is to perform a profilometric scan, either with a laser sensor, by committing a raster scan around the region to scan while obtaining the sensors position and depth information. This supplies a PTC covering the topology of the surface to scan. Since the NDE device's position should be recorded from the robot, the PTC will also be referenced to the robot. However, reconstruction using structured light is only featured in this work.

Reconstruction is specifically the process of obtaining a physical geometric profile in digital space. A raw PTC is the first piece of information obtained, sometimes multiple merged or processed together. If a raw point is required into being converted into a mesh, which occurs with this type of profilometry, Poisson reconstruction is a useful method for this conversion [116][94], available inside of CloudCompare using scalar fields (SFs). Commercial reconstruction devices have features to obtain a mesh as an output. The output of the reconstruction phase should also allow for any processing on the PTC before reconstruction, and after to potentially smooth the mesh. Processing should also split the sample under test and the background. The mesh with both is considered as the full environment. The full environment is useful for collision detection within the robot simulation, so joints of the robot do not hit recorded physical regions. Examples are shown in figures 1.2, 1.3, and 1.9 for different methods for reconstruction with a complex-shaped sample made of carbon fiber reinforced polymer (CFRP). Figures 1.2 and 1.3 are unregistered, meaning they are not aligned with the robot.

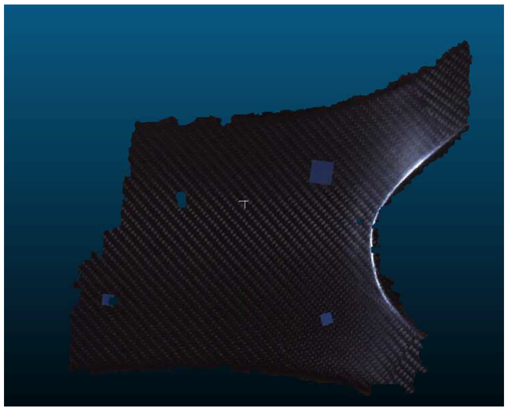


Figure 1.2 Structured light mesh as output from the Creality software.

Registration or alignment is how to orient the reconstructed cyber environment into the robot's base frame. This process is important to ensure the robot is accurately scanning over the region under test. The output is a transformation matrix, holding both position or translation and rotation information, which is applied to a mesh to align. The transformation should orient the cyber environment with respect to the robot's base frame, which enables topological knowledge to the NDE probe as an end-effector through inverse kinematics. A maligned example is giving in figure 1.4. Two methods were examined for this translation. Point Pairs Picking (PPP) is a method which takes at least three sets of points, location only, between "aligned" and "registered" points, and estimates the best transformation to match these point pairs. The aligned set should be in reference to the robot's base frame, while registered should be in the reconstructed unaligned environment. The matched point pairs should be in positions measurable in both the physical environment and the cyber environment. This is considered alignment calibration, where certain features on the surface are picked up for PPP. The issues with PPP are that it requires calibration regions, it requires measurement of the physical calibration regions with respect to the robot's base frame, and it requires choice of the same calibration marks in the virtual environment. The solutions used in this dissertation are done manually, physically by moving an end-effector to a calibration region, and virtually through CloudCompare. For the sake of the framework, reconstruction comes before alignment, though it may be possible to interchange these two processes. It is in this order since alignment while may take either a PTC or mesh but meshes are better for human interaction as points may be selected on a face between points, which may give a better approximation of feature selection then selecting a surrounding point. The second examined method is Center Point Transformation (CPT), which requires the transformation of the reconstruction sensor with respect to the robot's base frame on associated snapshots. To obtain this transformation, the reconstruction sensor may be placed as the end-effector or otherwise on the robot so it may be measured back to the base frame. Unlike PPP, which the sensor may be used around the robot, CPT needs it is on the robot. CPT enables simple and fast automation if the center point of the cyber environment, which is locally as $(0,0,0)m$ in Cartesian coordinates, as measured with respect to the end-effector or other orientation on the robot. The complication comes from black boxing inside commercial sensors which may distort this orientation from the center point from the ideal $(0,0,0)m$ orientation, potentially up to a couple of centimeters. The profilometric raster scan uses CPT per point instead of per PTC, where each point is measured through as the end-effector.

With the environment reconstructed and aligned, it is possible to generate a set path in which the NDE probe will be oriented on the sample's surface. Each individual point to orient towards is considered as a "waypoint", and the path from start to finish of all available waypoints is considered as the "waypath". Waypoints include both translation and rotation information. The goal is to generate a waypath with each waypoint gliding along the surface of the sample under test, with a lift-off to prevent collision between the sensor and sample. This collision may occur due to any errors in reconstruction or alignment if there is no lift-off. Typically, lift-offs need a balance between not being too close to the sample to damage the sensor, yet not too far away to reduce the quality of or even eliminate the returned signal from the probe. This is especially relevant for ECT and many subsurface tests, where signals even near the surface may have low signal-to-noise ratio (SNR). Along with this "low but not too close" requirement of the probe, is to ensure the lift-off is constant while the probe is gliding. If there is variance in lift-off, this will bias SNR which may distort the output data.

To obtain a waypath, a custom path planner was used. Commercial path planners exist for other surface applications such as polishing, sculpting, or welding. For example, SprutCAM has been applied for both CAD models [170] [47] and reconstructed models [145]. The benefit of these commercial path planners is that they have apply spline operations [184] to improve the smoothness of the scan path, which in turn prevents clustering of data points at waypoints where the sensor must stop and decreases operation time. From practice of at least SprutCAM using reconstructed samples, there was trouble in defining regions where traversal splines would be generated, due to the high polygon count. As such, a custom path planner was made that sacrifices splines for waypoints where the sensor stops at.

The custom path planner uses a novel approach: ray-triangle intersection arrays. The waypath generator inputs the reconstructed and aligned model and certain parameters such as location, resolution, and scan area. From the definition of the scan parameters, an array of rays is generated which tries to find intersections on the mesh. If an intersection is found, a waypoint is generated based on the location at the intersection. Rotation is also calculated from the normal of the face that was intersected. If there is no intersection, then a waypoint is not generated. Finally, a lift-off needs to be found. Two approaches were developed; one is a simple global lift-off where every point is translated by global Cartesian coordinates, potentially only in one direction. This approach works well on flat surfaces and supports the array shape of the scan plane. It is also a quite simple approach to set up, where only translations are added. The issue comes with curved surfaces, which waypoints generated on surfaces orthogonal to the global translation will move away from the surface, leading the probe to unwanted rotations and lift-off variations. To counter this, another lift-off approach was used, which applies lift-off per waypoint, along the normal of the intersected face. The allows for generated points to have the proper transformation to examine each face. The scan's shape may be affected, generally due to the coarseness of reconstructed models having overcompensated rotations on faces, meaning waypoints may overcompensate for its translation. This problem becomes more prevalent with higher lift-offs, where some waypoints become unreachable by the robot. One final concern about waypoints is that the robot must stabilize itself at each waypoint, meaning it must slowdown to reach the point, stop for a brief period, then re-accelerate to reach the next waypoint. This process increases scan times and clusters measured NDE data points, adding to some complications when reconstructing NDE data in post-processing about scalar fields. An example of this engine is shown in figure 1.5 and 1.6.

A waypath is not the final output to the robot, as this is oriented as points from the robot's base frame. To solve how the robot moves the NDE sensor to each waypoint, inverse kinematics must be solved. Inverse kinematics supplies the "joint set", which is the joints needed to reach every possible. Inverse kinematics is done within RDK as the robot simulation software, shown in figure 1.7. Some notes about inverse kinematics are that there are potentially infinite solutions per point, as there are several different joint rotations that may obtain the same orientation described from the waypoint. As such, the path that enables the least amount of time to move towards should be perfected. Some waypoints, despite being generated, may not be reachable. There are three conditions that prevent a waypoint, including movement between waypoints, being placed inside the joint set:

1. The waypoint is outside the robot's workspace or otherwise range to orient towards.

2. To reach the waypoint, a collision will occur. A collision may be from any piece of the robot, including joints of the arm or sensors, with the environment or itself.

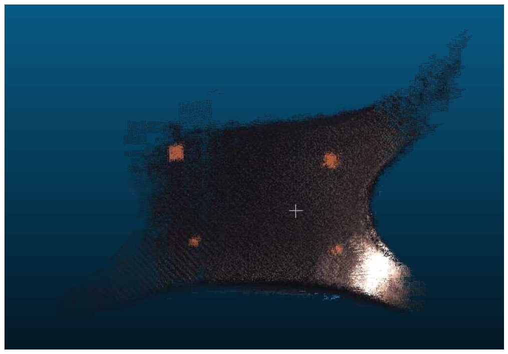
3. The waypoint generates a singularity that cannot be solved.

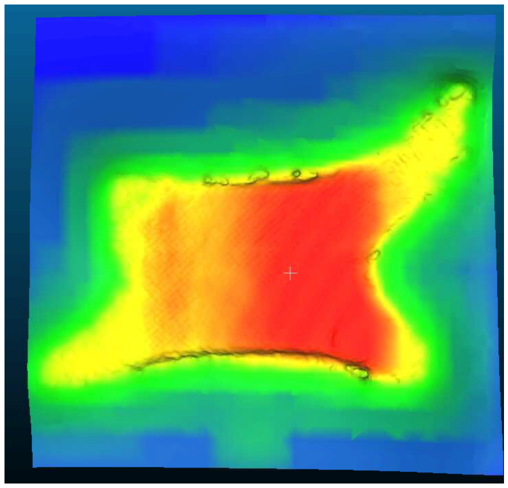
These conditions may be checked within simulation. Singularities are potentially still an issue when physically scanning. Once the joint set is generated and properly covers the region to scan by avoiding the earlier conditions, the joint set may be sent to the physical robot for conducting the NDE scan. A simulation may be run inside RDK to check for collisions, which makes the extra background information useful, so the robot does not hit what has been measured. That said, anything not collected inside the reconstructed mesh may cause a collision.

The NDE scan may be conducted when the joint set, based on the waypath along the sample, is available. The NDE should be attached as an end-effector of the robot and the probe prepared for scanning. While the scan is being conducted, there are two throughputs. The first is the probe's end-effector orientation, which is returned to the computer through Ethernet. The second is the returned signal of the probe, read in through a data acquisition device or chip connected to the computer. These two outputs are synchronized together, meaning that each recorded point floats inside Cartesian coordinates with respect to the robot's base frame, holding position, rotation, and NDE information. This is considered as a 3D+NDE point. Each 3D+NDE point is acquired as soon as both orientation and NDE are acquired, meaning they are acquired in between waypoints rather than only at waypoints. When the scan is done, a PTC of 3D+NDE points should be available, with a shape like the input reconstructed and aligned mesh sample. The method and output of the NDE scan is left ambiguous to allow for black boxing of different NDE methods. ECA inspection, which requires each channel to be oriented using the robot's tool frame per data point, is conducted to localize data with the robot. UT inspection later uses a similar process but only for one channel.

The 3D+NDE PTC should endure some post-processing. Currently, filtering of NDE data like 2D images is in a nascent stage due to redeveloping known algorithms but within 3D space.

Deconvolution algorithms are helpful for removing effects such as unwanted reflection and are generally developed with a requirement of a point spread function or similar functions for 2D images with fixed resolutions. In 3D space, many of these algorithms are not available or potentially have not been developed. This shows some of the novelty proven in this dissertation, which comes at a cost of missing available tools for post-processing. One tool that can be used, which has already been shown, PTC to mesh reconstruction. Applying Poisson reconstruction to 3D+NDE PTCs provides an NDE mesh with respect to the robot, which enables visualization of defects within 3D space. An example is shown in figure 1.8, 1.8, 1.10, and 1.11. There may be endless future applications with this sort of technology at hand. Much attention was dedicated to black boxing methods so they may be easily replaced with other methods. For example, a robot arm was used just to demonstrate the framework. If a mobile platform is used instead, such as a robot car or drone, the process remains: decide the environment with respect to the robot, generate a path to move along, record position and NDE data, and process the data. The full framework of this work is shown in figure 1.12.


(a) Point cloud, RGB from physical coloring.


(b) Unfiltered SF mesh after Poisson reconstruction.

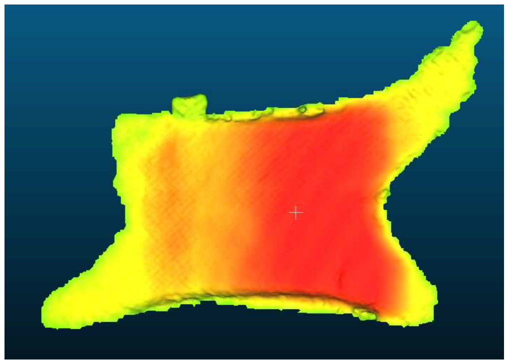

(c) SF filtering at 6.520.

Figure 1.3 Cyber environment of CFRP x-brace from a structured light snapshot: PTC to raw mesh.

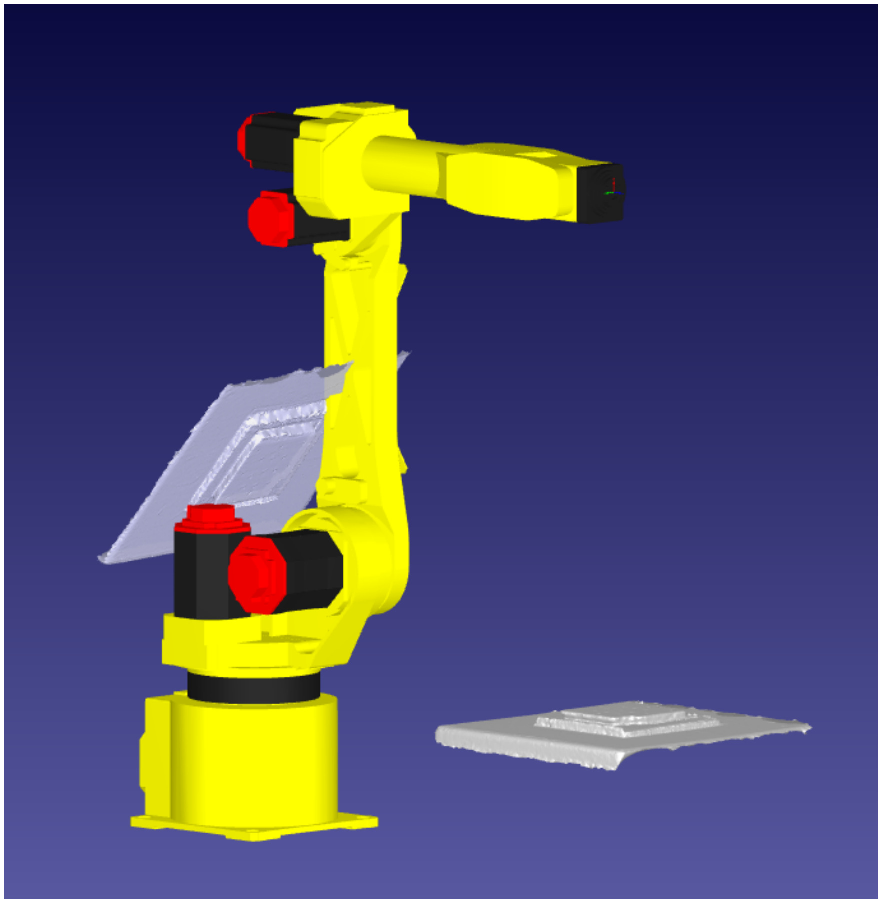


Figure 1.4 Unaligned versus aligned cyber environment mesh of a sample physically placed on a table in front of the robot. Without information of where this table should be at, the robot will orient the NDE probe at inappropriate locations.

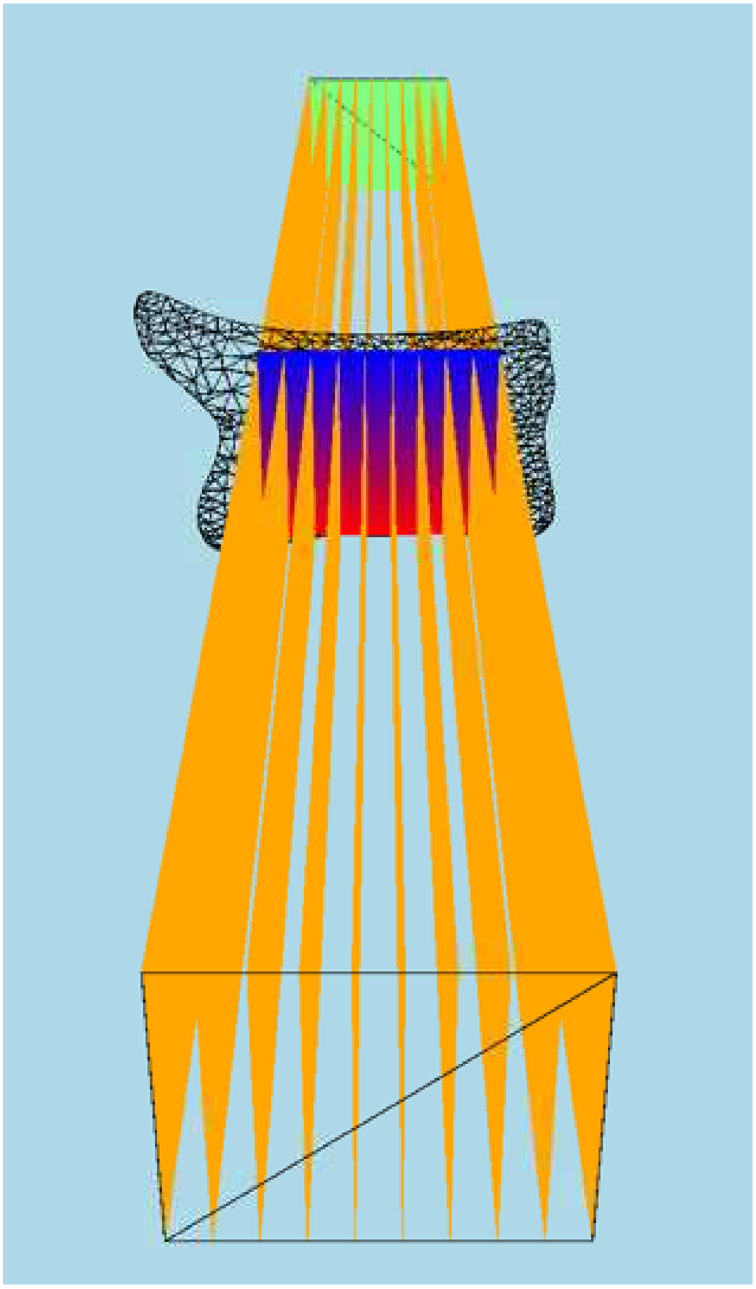


Figure 1.5 Custom path planner, where a waypath is generated from a reconstructed CFRP x-brace sample. The scan plane has an area of 13 × 15 cm and a resolution of 130 × 10 points.

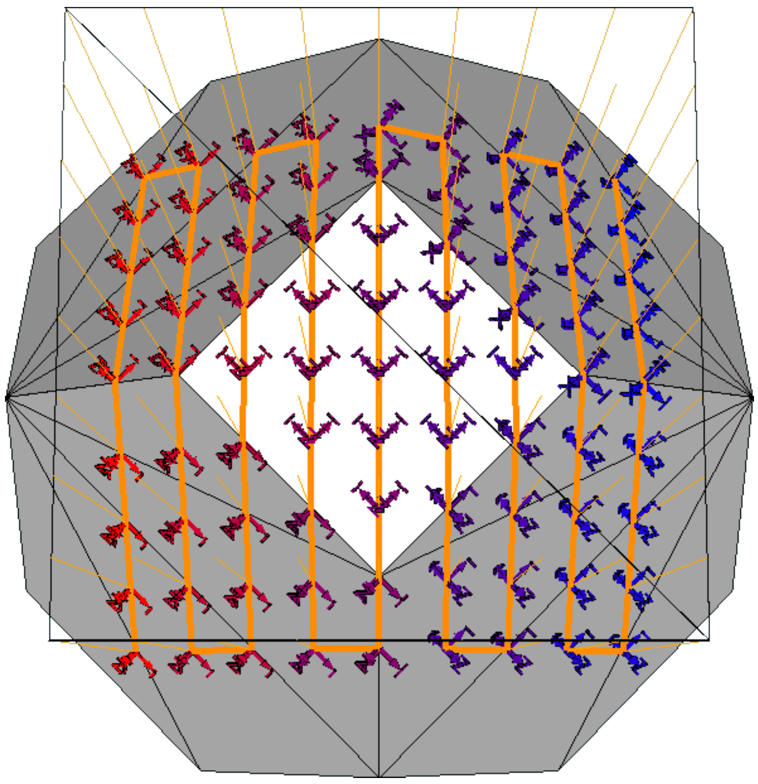


Figure 1.6 Another example on a low resolution sphere to more clearly show rotation from face normals.

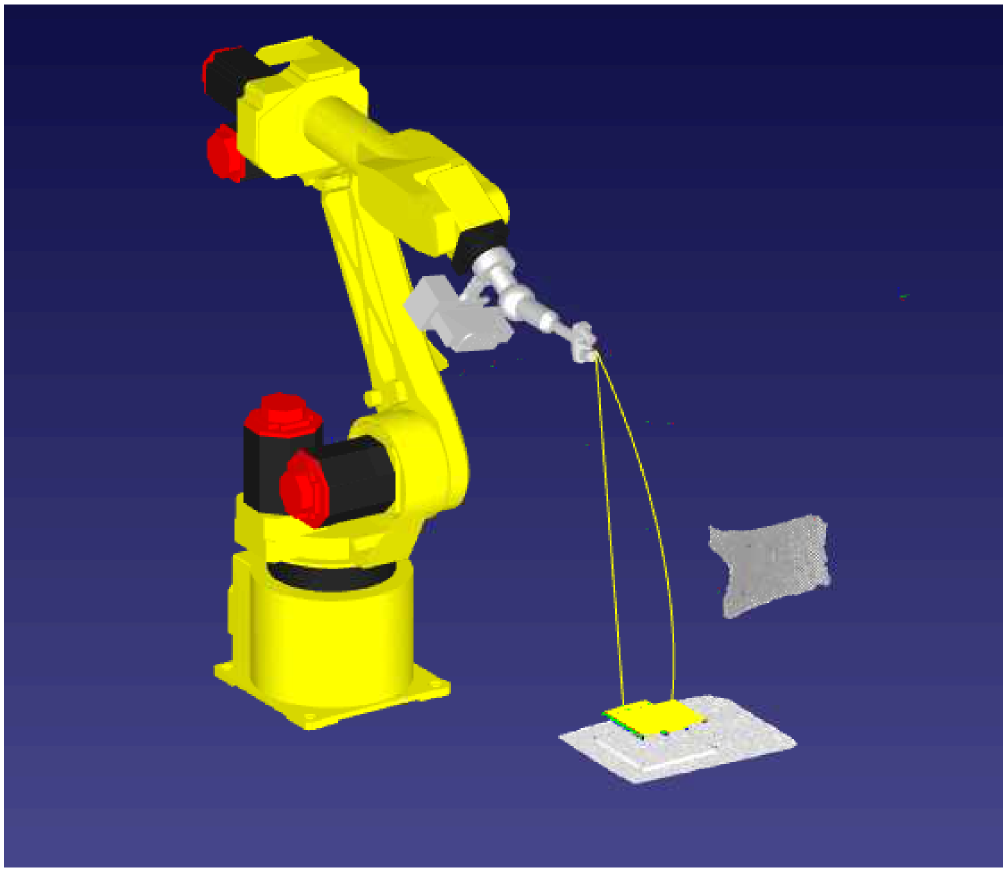


Figure 1.7 RDK simulation of the robot arm and two reconstructed samples, one aluminum sample on a table, and a CFRP x-brace. The aluminum sample shows the path where the robot will move towards, starting and ending at a safe location away from the sample.

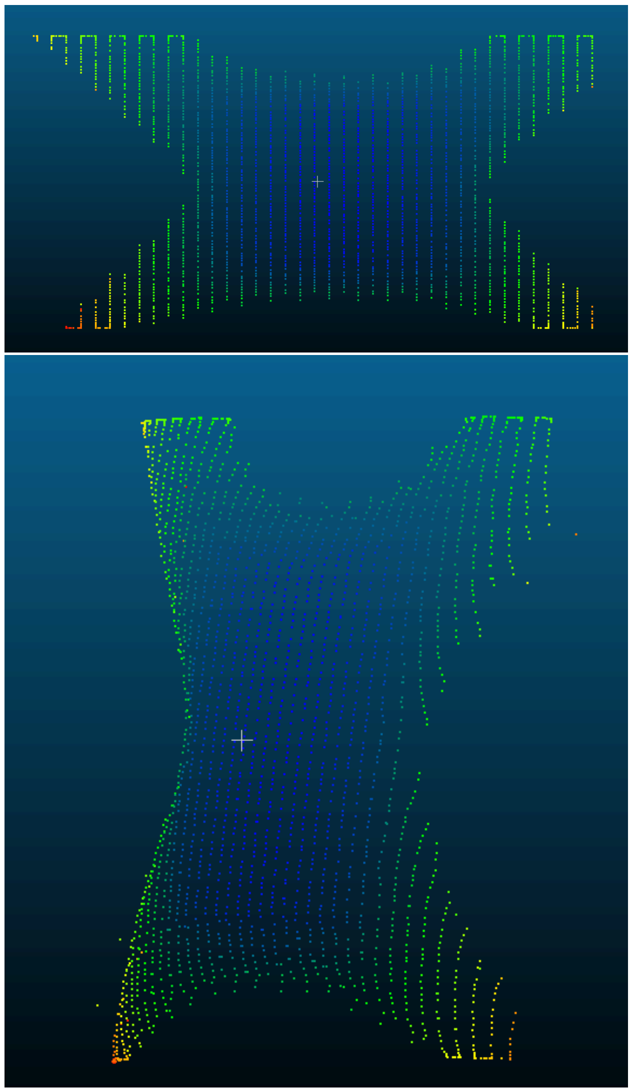
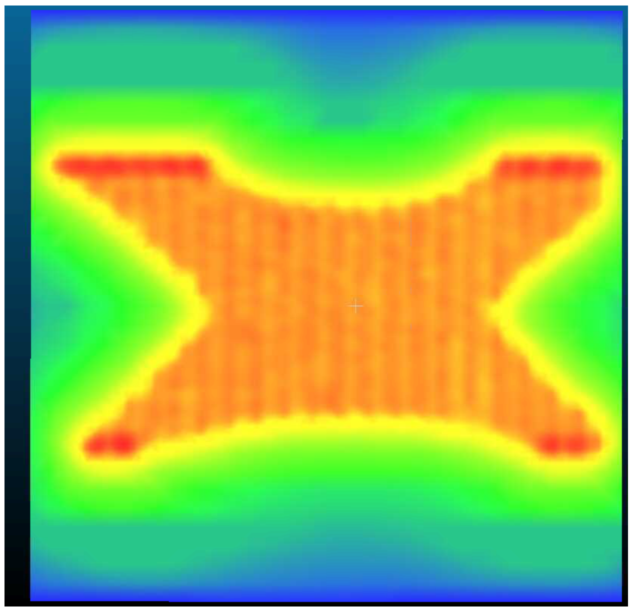
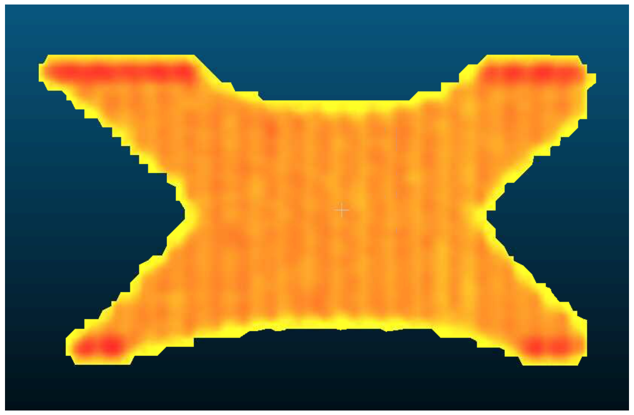


Figure 1.8 3D+NDE height-map reconstructed PTC of a CFRP x-brace, from air-coupled UT scanning. The waypath can be visualized with its raster-like pattern. Large gaps between points are just linear movements between the two. The colder colors are closer to the robot, which is also represented as the third Cartesian coordinate towards the robot.



(a) Unfiltered SF mesh after Poisson reconstruction.



(b) SF filtering at 4.913.

Figure 1.9 Cyber environment of CFRP x-brace from using UT profilometry to SF mesh. This environment is with respect to the robot's base frame.

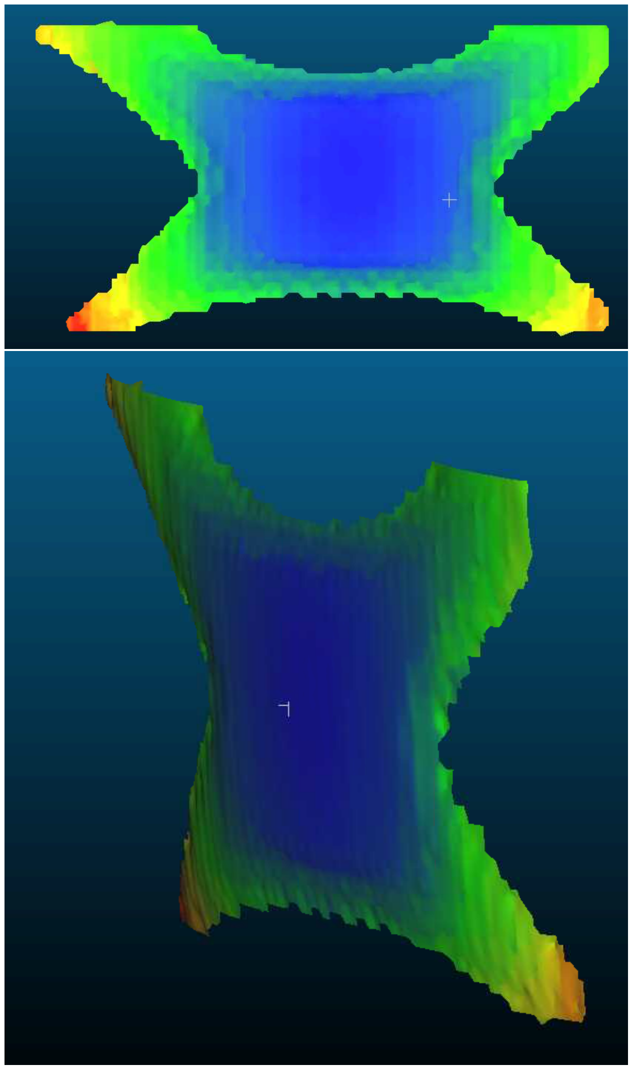


Figure 1.10 3D+NDE height-map reconstructed mesh of a CFRP x-brace.

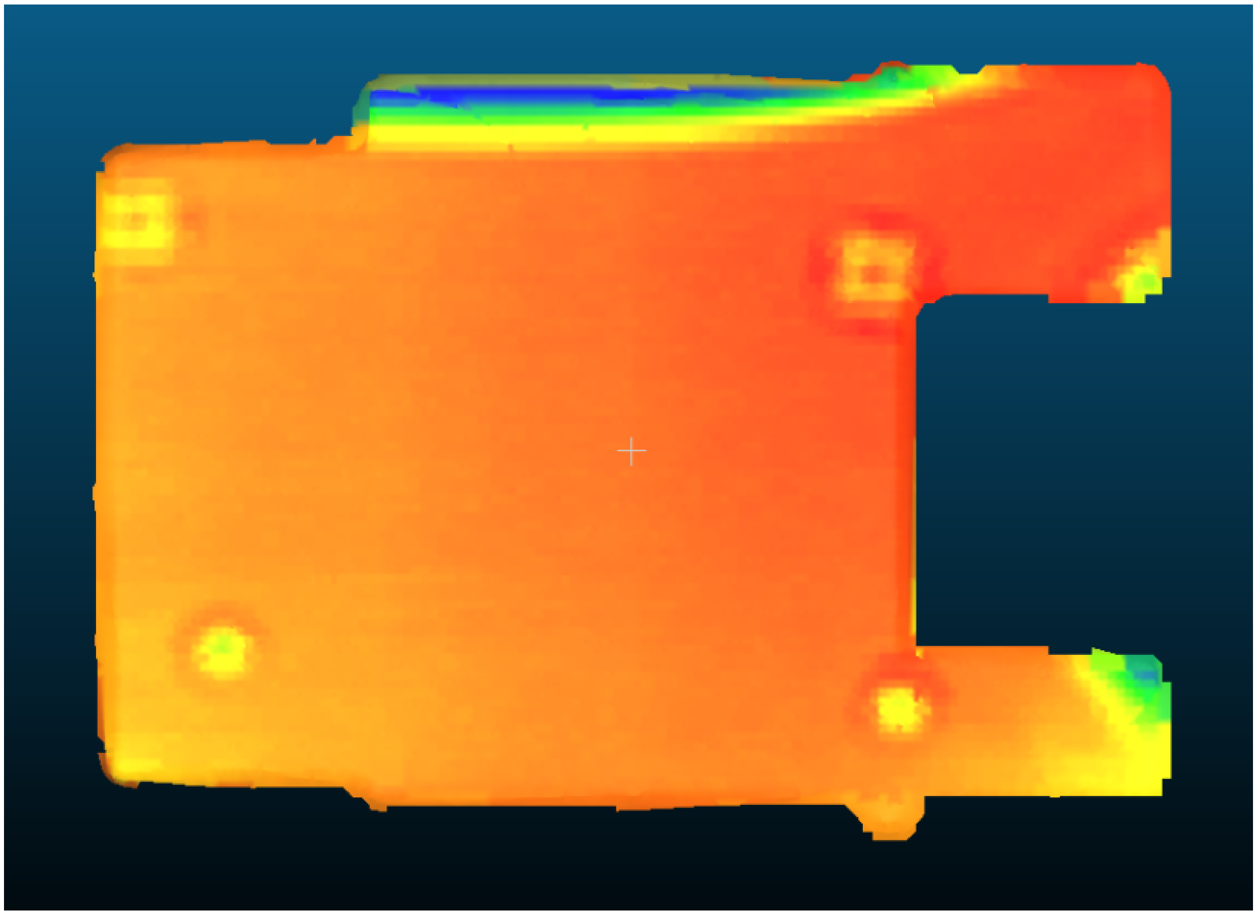


Figure 1.11 3D+NDE mesh from surface scanning on the CFRP x-brace in absolute summation mode, in which surface calibration stickers are picked up.

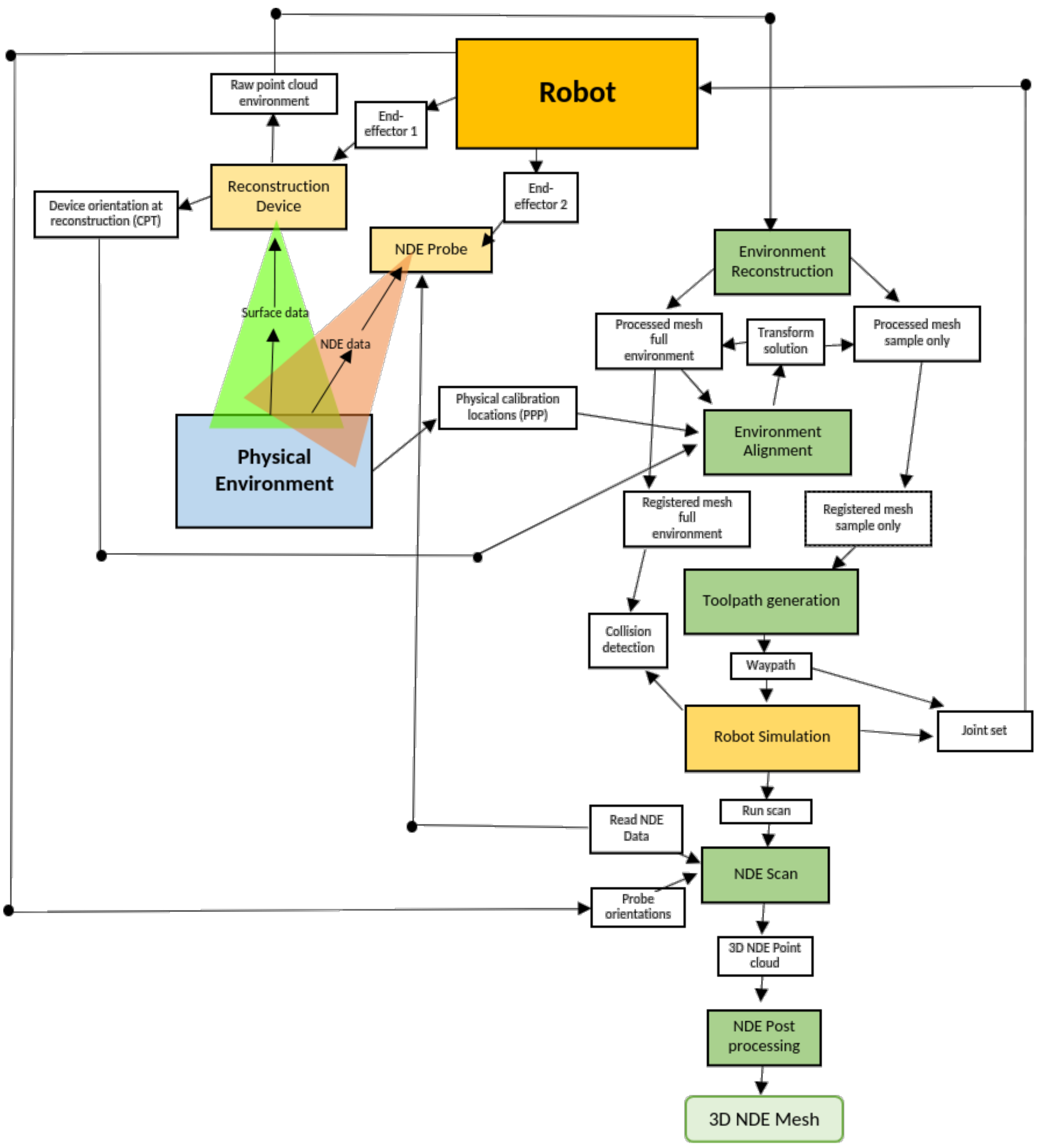


Figure 1.12 Framework layout, showing throughput of each relevant entity.

1.3 Objectives

Objective testing will be covered later, this section will provide a brief description of the objectives covered in the work. The ECA results were cross checked with ground-truth digital microscope data for several defects per examined sample to obtain the best objective information. This process uses cross-correlation to attempt to place the ground truth data with the ECA data. If it is found, the background and defect locations are checked per pixel and intersected to determine the percentage of pixels matched between the two sets. The samples examined were curved in both concave and convex patterns and were corroded for 7 days using the ATSM B117 standard.

Objective 1 *NDE complex surface scanning without a CAD model.*

Using the provided framework of reconstruction of a mesh of the area to inspect, aligning that mesh to the robot, path planning on the mesh, and conducting a "black box" NDE surface scan relevant to the sample enables freeform scanning of complex shapes. To enable robust scanning, the reconstructed mesh is used in ray-triangle intersection arrays to obtain a path where the robot may manipulate the NDE probes. NDE scanning assumes low error in the procedures prior to NDE scanning. The purpose of the objective is to enable a robust path planning procedure for surface scanning probes. Later, ECA scanning, and UT scanning experiments are conducted highlighting the framework procedure. Low error in terms of reconstruction and alignment are particularly important as this adds variability in lift-off and probe tilt. As seen in the ECA results of corroded samples, which require very precise measurements due to the low depths of at max $50\mu m$, some results are disrupted by errors that are generated from such errors. On top of this, robot mastering error was discovered to be a factor that required post-processing to be removed.

Objective 2 *ECA scanning of corrosion using a freeform scanning system.*

As just mentioned, there are some unique concerns when conducting complex scanning on ECA samples for corrosion due to sensitivity of data with small depths. Conducting an analysis on which factors would be important is vital for obtaining quality data. The largest concern is lift-off variation which saturates data. Other factors, such as ECA data desynchronization mismatching from robotic tool position, were later discovered to shift data in an unwanted way. Temperature fluctuations on the coil, either from the submitted power or changes in the room temperature caused significant changes in impedance that needed to be considered. The parameters for the ECA were to excite each coil $2Mhz$ in obtain data in absolute mode, which was found to be susceptible to heating concerns. This, appended to lift-off errors, complicated obtaining quality raw data. Hence, there was a need for post processing procedures.

Objective 3 *ECA post-processing of corrosion with respect to robotic systems.*

Post processing required methods to remove the errors mentioned previously. Lift-off may be reduced using detrending methods. However, this was not adequate for determining complex lift-off errors from robot mastering, which is non-linear and relevant to error per joint. To counter this, a novel post-processing procedure was developed: the array subtraction algorithm. This utilized commonly found errors per each coil that would be related to lift-off errors in order to find a normalized value per scan point that is subtracted to each coil. This is shown to effectively remove such data to help recovered data. Some data sets struggle from large lift-offs even with detrending and the unique array subtraction algorithm.

1.4 Contributions

To briefly break down the novelty of this work, here are several main unique contributions discussed. The examination of effects within a robotic environment to ensure accurate corrosion detection is a topic covered in detail within this work. As mentioned in the objectives, discussions on the effects of lift-off within different processes are important to obtain accurate results. Effects such as voltage saturation from heating also occur, which only add on to the errors within the environment through the high frequency and absolute mode measurements used to pick up corrosion areas. Specific post processing techniques are covered to recover saturated information as well. Methods to place the ECA information into a 3D PTC with respect to the robotic environment are also covered.

To enable the ECA freeform scanning system using computer vision, as well as other surface scanning methods, a unique framework shown in figure 1.1 was developed and deployed. This is done without a premade model or CAD of the scanning piece and is reconstructed on the spot before conducting NDE surface scanning. Registration or alignment is used to place the model into the robot's workspace, using point-pairs picking between physical calibration marks to virtual selection of the same calibration marks on the mesh. To enable pathing and motion planning on the mesh, a unique algorithm, ray-triangle intersection arrays, enables raster scanning on the mesh with respect to complex translation and rotation orientations of the robotic tool. This was shown to work with surface scanning UT, shown later in figure 7.36.

There are unique conditions that occur when conducting robotic ECA surface scanning for detecting corroded regions, requiring specific post processing to recover information on surface damages. An analysis on effects of ECT and ECA interaction with this setup are analyzed. It found that errors with the calibration or mastering of the robotic joints amounted to complex errors. A novel solution using an array subtraction algorithm to remove common lift-off effects on each coil of an ECA due to orientation error from the robot. This significantly reduced the error from calibration error. Other issues regarding probe orientation error would be from reconstruction error of the surface geometries and alignment error on the body of the mesh with respect to the robots. Special detrending algorithms were used to recover data from these scenarios. Unique post-processing for ECA with respect to robotics, in terms of ECA data to PTC to 2D post-processing. When the array data is collected real-time, it is interpreted as a PTC in real-time, in which the post processing procedures prior to 2D interpolation are considered. That array point cloud is also generated through transposing each point with respect to the translation and rotation of the tool at any time during the scan.

1.5 Literature Review

To focus on works not too broad from the scope of this dissertation, only robotic systems capable of NDE will be discussed. The importance is to detect defects to prevent the failure of a part or overall system, while preventing human intervention as much as possible, either to prevent hazards or otherwise OSHA violations, and to automate the NDE scanning process. Intelligent nondestructive testing and evaluation (iNDT&E) refers to the integration of artificial intelligence merged with the internet of things and systems for decision making [108]. These approaches will not be met in this dissertation, however they are important in the discussion of the ultramodern technology regarding NDE 4.0 compared to the framework laid out in this dissertation, to understand the progress leading to where the technology stands today.

1.5.1 History Leading to NDE 4.0

To focus on works not too broad from the scope of this dissertation, only robotic systems capable of NDE will be discussed. The importance is generally to detect defects to prevent the failure of a part or overall system, while preventing human intervention as much as possible, either to prevent hazards or otherwise OSHA violations, and to automate the NDE scanning process. Intelligent nondestructive testing and evaluation (iNDT&E) refers to the integration of artificial intelligence merged with the internet of things and systems for decision making [108]. These approaches will not be met in this dissertation, however they are important in the discussion of the ultramodern technology regarding NDE 4.0 compared to the framework laid out in this dissertation, to understand the progress leading to where the technology stands today.

Before the industrial age, manual labor was needed for production purposes, where evaluation was strictly visual based around this time. Expertise in some methods was needed. For example, "sonic" techniques were used by blacksmiths to decide the quality of certain steels used for swords by checking the ring or tone when hit. Clearly if a sword used in a combat scenario breaks due to low strength, this would not end very well for the wielder of the sword. Hence why blacksmiths would employ very primitive but relatively effective nondestructive methods to evaluate sword strength. As for Robotics, there was conceptualization of an idea of a "humanoid machine" before the world of industry let alone robotics. This came into being with Leonardo da Vinci's fully programmable four degrees-of-freedom robotic arm, dating back to 1495 but not recognized until much later after the industrial revolution [149]. In 1738, Jacques de Vaucanson designed an automated flute player "andrioide". This is considered to have booted the industrial revolution [51]. Wolfgang von Kempelen created a chess player in 1769 with mechanical control of both an arm and its fingers.

Vaucanson's work sparked interest in Jaquet-Droz, who created multiple automations to resemble human anatomy in 1774 [126].

The first significant stance of industrial applications started with industry 1.0, appearing in the 1780s. This began from mechanical looms powered by steam engines. Note that steam engines have had earlier developments well before it's industrial usage [50]. In the early 1800s, the first true steps towards NDE came forth with the first thermographic, infrared, and electromagnetic induction observations. NDE availability was premature during this time since the technology still needed further advancements. Even in terms of awareness and safety regulation, developments would need to be made after a disaster. In 1854, there was an accident where a boiler exploded causing 12 deaths, 50 injuries, and the property destruction of the boiler room and nearby blacksmith shop. The boiler was seen beforehand as brand new and from a reputable seller with high safety margins, however this was before advancement in thermodynamic principles which shocked courts and engineers after the accident [132]. This accident sparked awareness and requirements for inspection purposes, a significant step for practicality for NDE, though techniques outside of past methods were not technologically available. Despite awareness, another incident occurred from the *Sultana* steamship failure in 1865 where multiple boilers erupted, causing an estimation between 1200 and 1600 fatalities [153].

Industry 2.0 came forth in the 1870s with production lines and conveyor belts for automobile manufacturing [190] [85] [61] [99]. In 1898, a public exhibition was held demonstrating Nicola Tesla's remote-controlled and automated submersible boat [14] [126]. The idea of an "automat" or robot came further into public consciousness in 1917, from the works of Josef Čapek in the short story, and later his brother Karel Čapek *Opilec* from the play *Rossum's Universal Robots* [28] [29]. Though Karel's work was in protest of robotic applications [1], this did not stop the spark of interest on this mythical sort of technology [80]. Between 1938 and 1942, Isaac Asimov was further interested in the concept of robotics in his short stories defining the three laws for robot behavior [10]. Unmanned vehicles and aircrafts were also developed around the 1920s [12] [40] which would later ignite development of autonomous vehicles. Back to the discussion of

NDE innovations, between the knowing's of accidental catastrophes and need for prevention, there were several significant advances that have led to staples in NDE technology and requirements for inspection, eventually leading to some deployable NDE instruments near the end of this industrial age. The tragic catastrophe of the *Sultana* lead legislation where inspection was needed to insurance for boilers. Between 1911 and 1914, codes were being formed with visual testing, an immensely aged method before industry 1.0, was published as a starting method for NDE in 1915. Despite the method's long age, the coding of visual inspection gave an official impact for NDE. In the meantime, in 1879, there was nascent usages of Eddy current, and in 1895, the discovery of x-rays. Industrial radiography was also developed in 1922. During this time, several train accidents occurred due to derailing, many from broken rails and welds [148] [173]. Another disaster struck due to a serious train accident occurred due to a derailment, killing 29 people and injuring over 60 more [166]. This led to the development of electric current and magnetic field detection systems for inspecting railroad tracks. Around 1935 to 1940, Eddy current instruments began development. At the same time, World War 2 was erupting and finally erupted in 1939. World War 2 was important for NDE, so from war efforts came awareness and concerns on defects and otherwise imperfections output from industrial applications. In 1941, the American Industrial Radium and X-ray Society, later named the American *Society for Nondestructive Testing*, was created. NDE at this point was regarded as an accepted technology. From 1940 to 1950, ultrasonic and acoustic emission methods also started developing with flaw detection, portable thickness measurement instruments established, in 1942 and 1946, respectively. Acoustic methods were matured for NDE in 1950. [53]. Eddy current was also further developed in the 1950s and 1960s [77]. The avenue of materials was also changing, with more demand for switching from heavier metal structures and components to carbon fiber starting in the early 1960s [141]. In the meantime, the first graphics system was developed by the Massachusetts Institute of Technology in the mid 1950s, with the first computer-aided modeling software "program for numerical tooling operations", or PRONTO, being available in 1957 [8].

Industry 3.0 innovated further in the 1960s with the availability of logic controllers incorporated with earlier and new automation systems and further developments in robotics. General Motor's launched the Unimate to aid assembly lines for automobile productions between starting its use in around 1962, with approximately 8500 units sold [48] [56] [80]. This sparked competition, with Japanese cylindrical robot Versatran (versatile transfer) adopted by Ford in the 1960s [20], and European robots running monotonous pick in and out functions in 1967 [172] [67] [126]. The first robot used for surgical use was developed in 1985 from a robotic arm for stereo-tactic brain biopsy supplying a high accuracy of 0.05mm [80]. Continuous development arose for industrial robots, for various use from manufacturing to welding to medial applications, and by 2011, the number of operational industrial robots across with world was around 1,153,000 units, with 193,000 in America, 577,000 in Asia and Australia, and 370,000 in Europe. [134]. The availability of logical controllers enabled the digitization of NDE data as well, enabling in-process monitoring and visualization of defects on a computer [108]. There was a boom in NDE usage from arising fields, as NDE was widely accepted by this time. This leads to a split of interlacing NDE applications across several fields. In the 1970s, many advancements of NASA aircrafts required advancements in NDE technology, displaying the applications of various techniques for recommended usages [168] [130]. The NASA probably of detection program was also developed, enabling crack detection from ECT for metallic materials [151], and later in the 1990s, UT methods for fiber-reinforced composites [35]. For commercial airlines, various NDE methods were examined for practicality of inspection of cracks and corrosion of various air-frame structures which on failure would cause a catastrophic event. Eddy current proved useful in crack monitoring, while UT was practical for bond testing. Probability of detection principles were used as well. Cost benefits may also be estimated in terms of maintenance of mechanical parts [59] [147]. Other industries and such which received help from NDE were automotive, energy, civil, and military applications, just to name a few [15] [114] [74] [87] [25] [162] [86]. Computer graphics and computer-aided design was already discovering its optimization and usage in the late 1960s [13] and was well developed by the end of industry 3.0 [8] [104]. Computer vision and 3D reconstruction had early development in the 1970s, with first methods being "shape-from-shading", in which normal information for surfaces are found from lighting and shading information [82], edge and curve detection for polyhedral modeling, stereo vision, and structured light [163] [154]. NDE saw multiple usages of robots in energy, aerospace, automotive, and shipping applications, however had difficulty with scanning objects with complex shapes [22]. NDE found usage in merging robotic technologies and computer graphics for conducting complex scanning and data fusion [38] [26] [84] [108]. Artificial intelligence methods were also being developed for interpretation of NDE defect information [186].

Industry 4.0 is the current industrial phase, which was coined by the German federal government in 2011 [91]. The phases of industry can be summarized from manual labor before industry, the mechanization of labor through steam in 1.0, electricity and mass production lines in 2.0, information technology in 3.0, and intelligent technology in 4.0 [103]. Progress for industry 4.0 is in rapid development. By the end of 2021, a total of approximately 3,477,127 units worldwide were in operation [135], nearly triple the amount from the start of industry 4.0 a decade ago in 2011! Many of the related applications will discuss technologies between industry 3.0 and 4.0. Industry 4.0 enables smart automation from advances in artificial intelligence and robotic technologies to improve productivity for industrial applications. A system that merges intelligent digital spaces with autonomous instruments for real-time applications is considered as a cyber physical system (CPS) [96] [98]. Embedded systems play an influential role easing new practicality in interfacing physical and cyber worlds. Sensors and actuation systems used with an embedded system enables sensor networks and machine to machine interfacing [138]. A "digital twin", which will be referred to as the physical-to-virtual correspondence, is the interface between physical assets and digital systems, enabling data fusion for NDE with knowledge of physical entities [109] [144]. Artificial intelligence allows for smart decisions from robotics, which is a challenge as computers tend to soar better for tedious but simple operations. Hence why "human cyber physical systems" are considered, which merges the decision-making skills of a human with the computational power and physical capabilities of a CPS [189] [45]. Significant developments in robotics have been made in this age from the usage of embedded systems, cloud computing, and artificial intelligence [64] [174] [133] [92]. The internet of things, big data, and cloud computing techniques provide valuable information and computation power that may be integrated with CPS and NDE [175] [157] [115]. The usage of NDE 4.0, the integration of industry 4.0 techniques in the field of nondestructive evaluation, may prove beneficial in three potential components: increasing NDE capabilities (industry 4.0 for NDE), belief to improve industry 4.0 (NDE for industry 4.0), and potential for human integration (human consideration) [171]. Unmanned vehicles are also of interest, to allow robotics to handle operations that would be hazardous for human inspectors [42] [159]. As seen within the several past years, artificial intelligence (AI) has been surging, including for increased demand for hardware focused on AI training which would directly affect NDE 4.0. For example, deep learning allows for autonomous defect characterization and classification from NDE devices [27]. Finally, the next predicted age of industry 5.0 is currently being conceptualized, with 4.0 working as an intelligence base, and 5.0 on a symbiotic stage with emphasis on sustainability human-robot co-working [103] [43] [44] [45].

1.5.2 Related Applications

This section will discuss the application of robotics incorporated with NDE for defect detection purposes. Relevant topics to the proposed framework will be discussed as well. Techniques from industry 3.0 and 4.0 will be covered, from a range of robotics from robotic arms to mobile inspection platforms. Methods covered a broad set of applications for robotic NDE systems as well.

There is a wide selection of usage of robotics for NDE [23] [121] [187] [101], some with remote usage and some automatically ran for unmanned grounded vehicles and aerial devices. Mobile platforms with inspection systems are useful for inspecting areas that would be hazardous or even impossible for a human inspector to examine. Pipelines are a notable example of this. While pipes may be assumed to have simple shapes, they are long and winding. Some may be beneath the surface. Depending on pipeline application, they may be inspected with interior or exterior inspection systems. Harmful defects may be detected across the scan region of the robotic system that may cause a leak or eruption if not dealt with. A modern inline pipe inspection system was developed which uses laser profiling inside the pipes to detect defects within the pipe [129]. The design uses convolutional neural networks to aid in defect detection. A previously tedious and worst impossible task possible from NDE robotics. Another example of energy is power plants. There are examples of OSHA non-compliant methods for inspection of power plants, such as inside boiler walls. There are robots that may attach themselves on rails inside the robot and move around to parse and repair defects [156] [70]. For tall structures, there may be difficulty for a human inspector to safely reach a region to inspect. Some crawling-based robotic systems interfacing simultaneous localization and mapping (SLAM) methods for visualization on a mobile platform for inspection of large structural assets [113]. Portable systems within hazardous environments may prefer to have low-cost yet rugged scanning probes and instruments in case of any damage done to the robot system. ECT is a useful technique with mobile platforms where the probe may be placed under the robot itself. This enables the probe to be close to the scan region by spring-loading the probe to against the surface. These sorts of robotic applications, where the robot system is on a set path, have a different approach in framework since the scanning region is on a set region, so the need of environment reconstruction is less required. For more nimble applications such as robot arms, or non-fixed path robots from wheeled mobile platforms to drones, environment reconstruction is essential for autonomous NDE scanning.

Surface profiling robots are useful in deciding eye-hand calibration. A time-of-flight module was placed onto a robot arm for object-related tasks [4]. Stereo vision techniques have been used for finding eye-hand calibration for welding applications as well [33]. Similar welding robots used audio sensing and structured light [2] [95]. Other methods use impact echo and ground penetrating radar from mounted modules [150]. If using a CAD, then there are some options to decide misalignment between the CAD and physical object under test as well as 3D surface approximation [143]. Scan path accuracy was evaluated from commercial software using NDE techniques and accuracy maps, showing positional path accuracy [127]. Stereo cameras have been used alongside welding and NDE applications [33] [150].

With more nimble actuation systems comes more robustness at the cost of requiring knowledge of the surrounding environment. Robot arms are particularly useful in this regard; however, many earlier applications use CAD models. Much research revolves around CAD models, though some may include open-loop structures. Mostly UT and ECT NDE methods will be covered, as they are within the scope of this dissertation. Solutions exist for 6-DOF manipulation of sensors through inverse kinematics using CAD models [184] [111]. Ultrasonic testing has received help from the integration with robot arms. Many arms are in a dual configuration for through-transmission mode, which requires synchronization [110]. There are early systems using dual waterjet through transmission modes, which used CAD models of CFRP helicopter components and obtaining 3D evaluation from the separate walls retrieved from the UT system [72]. Other NDE applications for helicopter part NDE also exist [137]. This was done from setting time gates alongside the array of time versus voltage data, as well as highlighting a gate-less data acquisition scheme. Another dual water-jet robot arm system was used to scan rectangular and cylindrical composite materials from through-water transmission [73]. The paths were generated from trajectory curves along the cylinder. The dimensions of the defects ranged from 3 to 5mm, which were detected in each C-scan. A similar dual water-jet system was used alongside phased array ultrasonic testing and using more complex CAD models [119]. Manual path planning is considered, where a linear pattern of points is entered into the teach pendant, however this is a cumbersome task especially for scans requiring higher resolutions. The path planning was done in MATLAB and simulated inside of CENIT-FastSurf [123]. An aerospace composite winglet with defects ranging from 5 to 15mm was inspected with a resolution of 0.6mm and water jet nozzle lift-off of 8mm. The result showed defects from a C-scan image. Water used as the couplant is not wanted, as water near expensive electronic systems generally does not pair well, hence why air coupled is used in this dissertation. Aerospace components tend to hold large and complex shaped objects. Composite materials are also common. A mobile robot was used which the robot arms glide on a rail system [112]. A multi-robot for welding and immediate NDE system was developed, in which an open-loop architecture is used for welding, while UT performing with high accuracy with repeated scanning [169]. ECT array scanning with robotic systems has been done as well. A flexible eddy current array was placed on a robot to inspect corrosion cracks on nuclear assets, where 2D images and smith charts are shown for different frequencies [62]. A robot arm system was used for scanning aeronautical structures using ECT as well [136]. To deal with lift-off issues regarding ECT, which needs to be close and constant between all scanned regions, a force-torque sensor was used alongside it. Other NDE methods receive help from robotic actuation systems. Thermographic images may be stitched together at different perspectives to increase inspection regions [63] and with robotics [122]. Hyperspectral cameras have been used alongside robot arms for scanning CFRP panels [185]. Data fusion for aviation components, to fuse "twin data", has also been researched [105]. An advanced state-of-the-art UT inspection system enables UT probes to conduct both NDE scanning and "in-process" path planning in real-time. This system determines the trajectory of the probes to obtain surface mapping allowing for path planning from a single probe start position, which was conducted using curved steel samples for volumetric evaluation [120]. ECT robot arms developed use different methodologies. Rotary axis scanning is common for ECT inspection robots, in which the robot moves the probe to a rotating cylindrical sample to obtain fast inspection, used for nuclear asset inspection here [62]. This method is limited to objects that can be rotated, which may be an issue for heavier or much more complex objects. Aircraft wing inspection from off-line planning utilizing a CAD model can be seen here [127]. This is limited by the requirement of reverse engineering for certain required models for path planning. A flex array probe was used to scan curved models by manipulating the flex probe along the path but is limited to just one axis of scanning [188]. The difference between available ECT robotic arm systems and the developed system is the capability of freeform scanning utilizing mesh reconstruction for path planning.

CHAPTER 2

THEORY OF PHYSICAL AND VIRTUAL SYSTEM INTERFACING FOR NDE ROBOTICS

This chapter's purpose is to lay out the analytical aspects of the system, which revolve around physical and digital spaces. First, some basic definitions. Space is a boundary which entities lie within. Physical space has an infinite boundary, or the universe. To simplify the prospects of physical space for our applications, it will be any entity that partakes within or associated nearby a robot's workspace. A robot's workspace is the boundary restricted by the building of the robot. For example, an industrial robot arm's workspace is bounded by all kinematic solutions of its end-effector, the holder of a tool, in this case as the NDE sensor. This may be visualized as a "sphere" holding a volume of potential locations the end-effector may reach. This will be considered as the environmental boundary space E. The robot's base frame is the "starting" frame in which a kinematics returns its coordinates back to. Inside a robot arm's workspace includes the physical sample, holder of the sample, the ground or walls around the robot, the robot itself, any equipment, sensors or wires, etc... When using a mobile robot such as a drone, which unlike an industrial robot arm which is assumed to remain in a static location due to its heft, has a workspace towards wherever it may reach. The potential entities available for such a system, per se outside of a laboratory setting, are innumerable as a result. As a result, brevity will be deployed when discussing entities within the environment with two simple classifications of any entity in the environment: the sample, and the background. Further discussion on physical samples, backgrounds, and robotics is saved for later.

2.1 Physical Environments to Digital Space: Analysis

Digital space is the "infinite" territory in which entities may occupy. Digital space is assumed to be in 3D Cartesian coordinate space, having the axes "X", "Y", and "Z". The goal is to solve the virtual-to-physical correspondence which connects both worlds. "Digital twins" is a similar definition of this connection [109] [144]. The most basic entity within a digital space is a point. A point is an entity bounded within digital space with values for X, Y, Z, defined as:

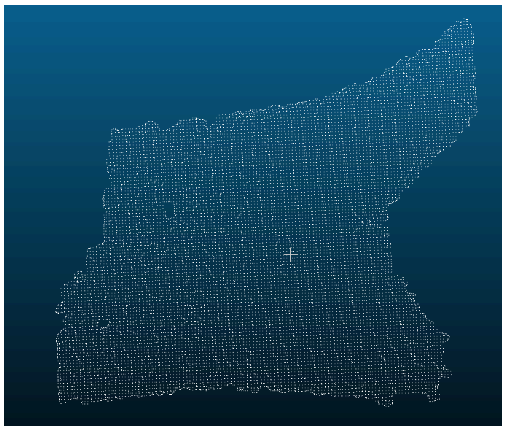
$$p = [p_x, p_y, p_z]$$

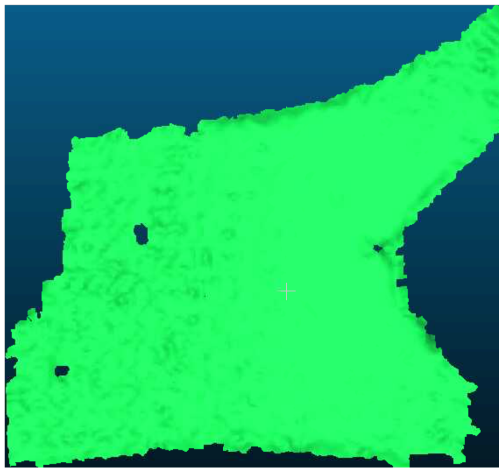
$$-\infty < p_x < \infty$$

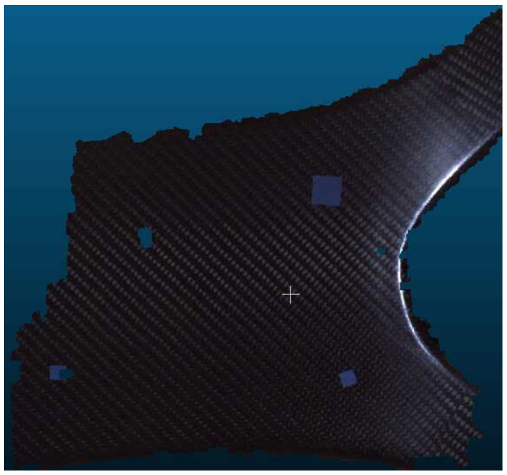
$$-\infty < p_y < \infty$$

$$-\infty < p_z < \infty$$

"Infinite" is not a true statement in digital space, as boundaries are limited to the memory occupation with respect to the digital space design. For example, a float in *C#* is a value between $\pm 3.4 \times 10^{38}$ with the smallest value being $\pm 1.5 \times 10^{-45}$, with 4 bytes of memory used per float [117]. There are dedicated constants for negative infinity and positive infinity when underflows or overflow or division by zero occurs, however a point cannot be defined by these traits without causing an error. Using float to define a digital space, the max values which bound the location of points are $-3.4 \times 10^{38} < [p_x, p_y, p_z] < 3.4 \times 10^{38}$ with 12 bytes of memory per point. If using a double value instead, the max range is now between $\pm 1.7 \times 10^{308}$ and smallest number $\pm 5 \times 10^{-324}$, doubling the memory required for 8 bytes a value and 24 bytes a point! As these values far surpass anything needed for these applications, the digital space which points lay within is considered as infinite. A point may hold different properties, such as color or scalar field values (discussed in reconstruction chapter). A point may also be considered as a vertex. Color is important in visualization of data from PTCs. Color comes in different forms. Typically, a color will hold values red, green, and blue (RGB), which each color is defined by one byte or three bytes for RGB. In this combination, there are 2^{24} or $16,777,216$ possible colors. Sometimes, color will use float values between 0 and 1. RGB will be discussed in reference to PTCs (and meshes via vertex coloring) obtained from reconstruction devices such as stereo-vision and structured light, which output color per point. Greyscale ranges only from black to white per point, using only a "grey" value. Greyscale is used for height maps, in which an axis or direction is colored "black" to "white". They are also used for displaying NDE data, where the grey color point represents the output from the NDE device. A colormap is a special function that adds non-linearity for greyscale maps. For example, instead of displaying a color just from black to white, a rainbow colormap may be used, in which the grey value is placed in an associated index from the colormap. This is useful for certain datasets which have nonlinear outputs. Examples are given in figure 2.1.


(a) Uncolored PTC from mesh vertices.


(b) Uncolored mesh.


(c) RGB colors from textures, in which calibration stickers are visible.

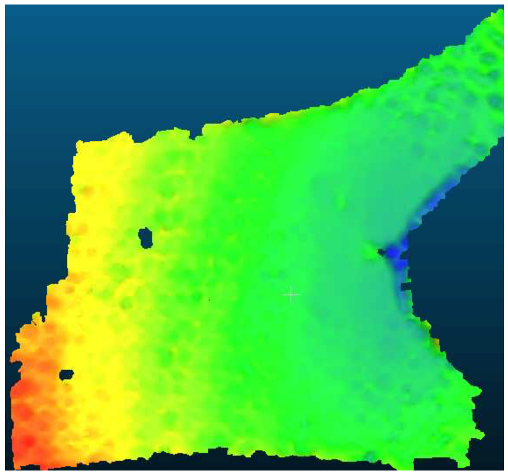

(d) Greyscale colors from a depth map, in which colder colors are closer to the viewer. This shows that the model is tilted closer on the left-hand side.

Figure 2.1 Color schemes from a reconstructed and registered structured light mesh of a CFRP x-brace containing calibration stickers.

The relevance of having assorted colors requires a list of points to differentiate from each other. A PTCs P is a linear list of points with a size n_P. For index i_P from 1 to n_P

$$P = [p_1, p_2, ...p_{n,P}]$$

There are different arrangements for PTCs. When using a reconstruction device, the formation of points will resemble a 2D grid in xy with the third axes z resembling depth at each point. During an NDE scan, points are generated alongside the scan path. It is important to emphasis that a PTC is a linear list, not a 2D or 3D list. The reason for this importance comes to the difference between a PTC and a mesh. A mesh M holds a list of vertices (plural of vertex) and a list of indices per vertex which connect vertices together to form a list of faces. In other terms, a mesh contains a list of faces. For index i_F from 1 to number of faces n_F:

$$F = [f_1, f_2, ...f_{n,F}] \subset M$$

With respect to indices, a mesh also contains, for index i_V from 1 to number of vertices n_V:

$$V = [v_1, v_2, ...v_{n,V}] \subset M$$

Meshes may also hold edges, normals, and definitions for textures. Facial profiles are important for finding the surface profiling of CAD or physically reconstructed objects, which are used to parse scan paths. Each face contains at least three vertices. To prevent redundancy, indices are used to point towards vertices that may be used multiple times for different faces. To abstract this concept, faces will be defined with respect to points rather than associated indices. Therefore, each face has vertices, with locally associated indices j_v and length of vertices inside a face L_f:

$$f_{i,F} = [v_{j,v}, v_{j,v+1}, v_{j,v+2}...v_{j,v+L,f}] \tag{2.1}$$

Faces must be at least three vertices long, nor should it the number of faces exceed the number of available vertices:

$$n_F \leq n_V - 2$$

The reason for the subtraction of two in this equality is due to the smallest requirement of three vertices per face. If the mesh is defined as a single, then the size will be at $n_F = n_v - 2 = 3 - 2 = 1$. More on triangulation of faces later. Converting from a PTC to a mesh is not a simple conversion, since a PTC is a linear list of points, while a mesh holds non-linear indexing of associated vertices to generate a list of faces. In other words, there must be a function that generates this list of indices which attach separate vertices together to form faces. Methods such as Delaunay triangulation [32] and Poisson reconstruction [116] [94] may be used to convert a PTC to a mesh. An example of a sphere interpreted as a triangulated mesh is shown in figure 2.2.

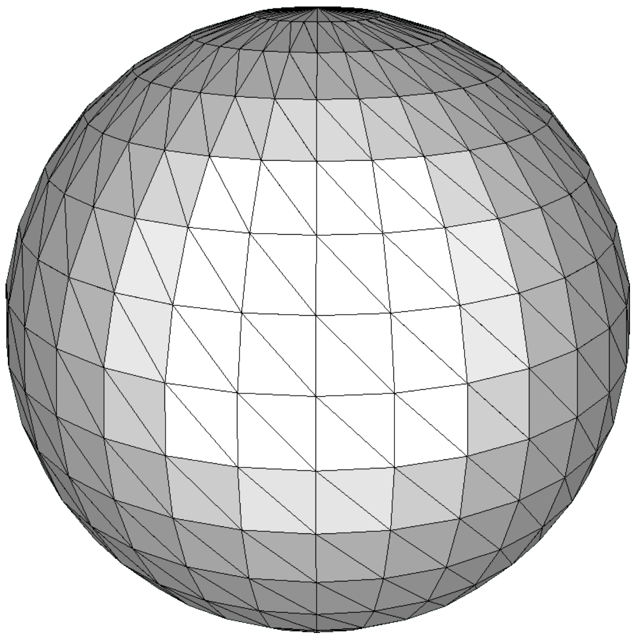


Figure 2.2 Sphere mesh made up of triangular faces.

2.2 Digital Operations for NDE End-Effector Positioning

This section will define several methods that are used for theory discussion and in future sections. The goal is to decide the calculations to simply send the sensor to the sample within the suggested framework, and to build open functionality to perform NDE scans with later chapters. These sections will hold baseline information for future chapters too.

2.2.1 Transformations in Digital Space

The discussion so far explains entities within digital space within only one reference frame. A reference frame is a set of 3D Cartesian axes which define space compared to the axes [30]. If a PTC or mesh is defined, every element will remain within its own local reference frame. The local reference frame is defined inside digital space at the zero point, at translation $[0,0,0]$. A robot arm works in multiple reference frames, one difference reference frame per joint. This can be considered as a "linkage" of frames, with the robot base frame starting the link and the final links being the end-effector tool end. When defining a cyber-physical environment, the reference frame of the PTC should be equal to the reference frame of the robot base frame, which is considered as the global coordinate in which all other frames are related too. In other words, every reference frame should have an alignment to the base or global frame. Figure 1.4 shows what happens when reference frames are not aligned. To align, one of the best practices is to incorporate matrix transformations. A transformation matrix O is a homogeneous 4×4 matrix which holds the properties of translation and rotation which can be applied to a body in digital space. Transformation matrices for the sake of this dissertation do not hold other properties such as scaling or shearing, as they are inappropriate for the desired application. A translation may be used to decide the alignment between two frames. Translation between frames A and B is defined as [31]:

$$O_A^B = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

or more simply, transform matrix O_A^B may be defined to hold the sets of translation and rotations:

$$O_A^B = [T_A^B, R_A^B]$$

In this matrix, there are two components, the rotation R and translation T. O is used as "orientation" to differential from translation T. Translation T is defined as the three-element vector:

$$T = [t_x, t_y, t_z]$$

and with respect to frames A and B:

$$T_A^B = [t_{A,x}^B, t_{A,y}^B, t_{A,z}^B] \tag{2.2}$$

For simplification's sake, $t_x = t_{A,x}^B$, $t_y = t_{A,y}^B$, and $t_z = t_{A,z}^B$. Translation T is transposed into the last column of the transformation matrix. R_A^B is defined as a 3×3 matrix holding rotation information:

$$R_A^B = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix}$$

A rotation matrix is defined from three rotation matrices from every Cartesian axis, with Euler rotations α along x, β along y, and γ along z [181]:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & sin(\alpha) \\ 0 & -sin(\alpha) & cos(\alpha) \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} cos(\beta) & 0 & -sin(\beta) \\ 0 & 1 & 0 \\ sin(\beta) & 0 & cos(\beta) \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} cos(\gamma) & sin(\gamma) & 0 \\ -sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = R_z(\gamma)R_y(\beta)R_x(\alpha) \tag{2.3}$$

Since the rotation matrix is bulky and may be defined from Euler rotations, rotations are mostly referenced as:

$$R = [\alpha, \beta, \gamma] \tag{2.4}$$

Or with respect to frames B and A:

$$R_A^B = [r_{A,x}^B, r_{A,y}^B, r_{A,z}^B] \tag{2.5}$$

The rotation parameters alone are not bound:

$$-180^\circ < \alpha < 180^\circ$$

$$-180^\circ < \beta < 180^\circ$$

$$-180^\circ < \gamma < 180^\circ$$

though, each may be restricted due to requirements of the robot environment. A waypoint p_w will contain a transformation and rotation:

$$p_w = [T, R]$$

Just applying transformations is not the only problem but finding the proper transformation. More on this in the environment transformation section.

2.2.2 NDE End-Effector Positioning and Robotic Simulation

Robotic simulation may be incorporated for a digital system and human operator to understand what will happen in an interfaced physical environment. In this framework, before a robot is sent to move, it will be processed within simulation. A robot arm is used, so discussion will focus on how a robot arm operates, however mobile platforms may integrate similar operations by replacing the means off moving the NDE probe compared to that separate application. For moving the probe as robot end-effector, kinematics between joints starting from the robot's base frame are used [139]. A joint set J with a length of n_j for any joint θ_{i_j} can be defined as:

$$J = [\theta_1 \theta_2 ... \theta_{n,j}]$$

Joints are variable and are physically controlled by motors on the robot. These joints are used with Denavit–Hartenberg (DH) parameters, which define the parameters of the kinematics chain between two joints. These four parameters are defined as [161] with respect to Cartesian coordinates:

1. θ: Joint angle: Revolute joint variable along the z-axis, between the previous and new x-axis.

2. d: Link offset: prismatic joint variable along the z-axis to a common normal.

3. α: Link twist: angle about a common normal

4. a: Link length: distance along a common normal

These parameters are used alongside a DH matrix [46], which is used to obtain the transformation between joint frames. A DH matrix is defined by four matrices from each DH parameter:

$$R_\theta = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_d = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_a = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) & 0 \\ 0 & sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It should be noted that the following parameters are the following 3×3 forms operated as 4×4 matrices, with translation zeroed for rotation operations, and vice-versa for translation operations:

$$R_z(\gamma) \rightarrow R_\theta$$

$$T_z = [0, 0, t_z], \rightarrow T_d$$

$$T_x = [t_x, 0, 0] \rightarrow T_a$$

$$R_x(\alpha) \rightarrow R_\alpha$$

So $\gamma_{R_z(\gamma)} = \theta$, $t_z = d$, $t_x = a$, and $\alpha_{R_x(\alpha)} = \alpha$. The full set of DH parameters per joint is defined as, for any index i_j:

$$DH = [(d_0\alpha_0 a_0), (d_1\alpha_1 a_1), ..., (d_{n,j}\alpha_{n,j} a_{n,j})]$$

$$DH_{i_j} = [d_{i,j}\alpha_{i,j} a_{i,j}]$$

Note that each θ_{i_j} is defined separately by the joint set J. The orientation between frames is defined as

$$O_{i,j}^{i_j-1}(\theta_{i,j}, DH_{i,j}) = R_{\theta_{i,j}} T_{d_{i,j}} T_{a_{i,j}} R_{\alpha_{i,j}}$$

With

$$i_j \geq 1$$

The desired solution is to equate the transformation of the sensor as the end-effector at n_j back to the robot's base frame at 0. To do this, all DH transformations defined from the DH parameters per joint should be calculated:

$$O_O^{n,j} = F_{fk}(J, DH) = O_{n,j}^0(J, DH) = O_1^0(\theta_1, DH_1) O_2^1(\theta_2, DH_2) O_3^2(\theta_3, DH_3) ... O_{n,j}^{n,j-1}(\theta_{n,j}, DH_{n,j}) \tag{2.6}$$

There are two important operations for using and retrieving joint sets using $O_{n,j}^0$. The first is forward kinematics, in which the joint set is known and $O_{n,j}^0$ should be solved for the transformation of the end effector back to the robot's base frame. There is only one solution for this problem, simply placing into the above equation. Any solution from forward kinematics is bounded by the the specifications of the robot from each frame. The set of possible forward kinematic solutions is considered as the workspace environment E, which tends to be spherical with robotic arms holding a volume of solution. Figure 2.3 shows E for the robot used within this thesis, which is simulated within RDK. There are restrictions, such as prevented solutions that would allow for self-collisions of robot components.

The second operation is inverse kinematics, in which the joint set must be found from a desired transformation $O_{n,j}^0$. This enables the robot to move the end-effector from an input path of transformations it should meet. Ignoring any discretization on joints, inverse kinematics has an

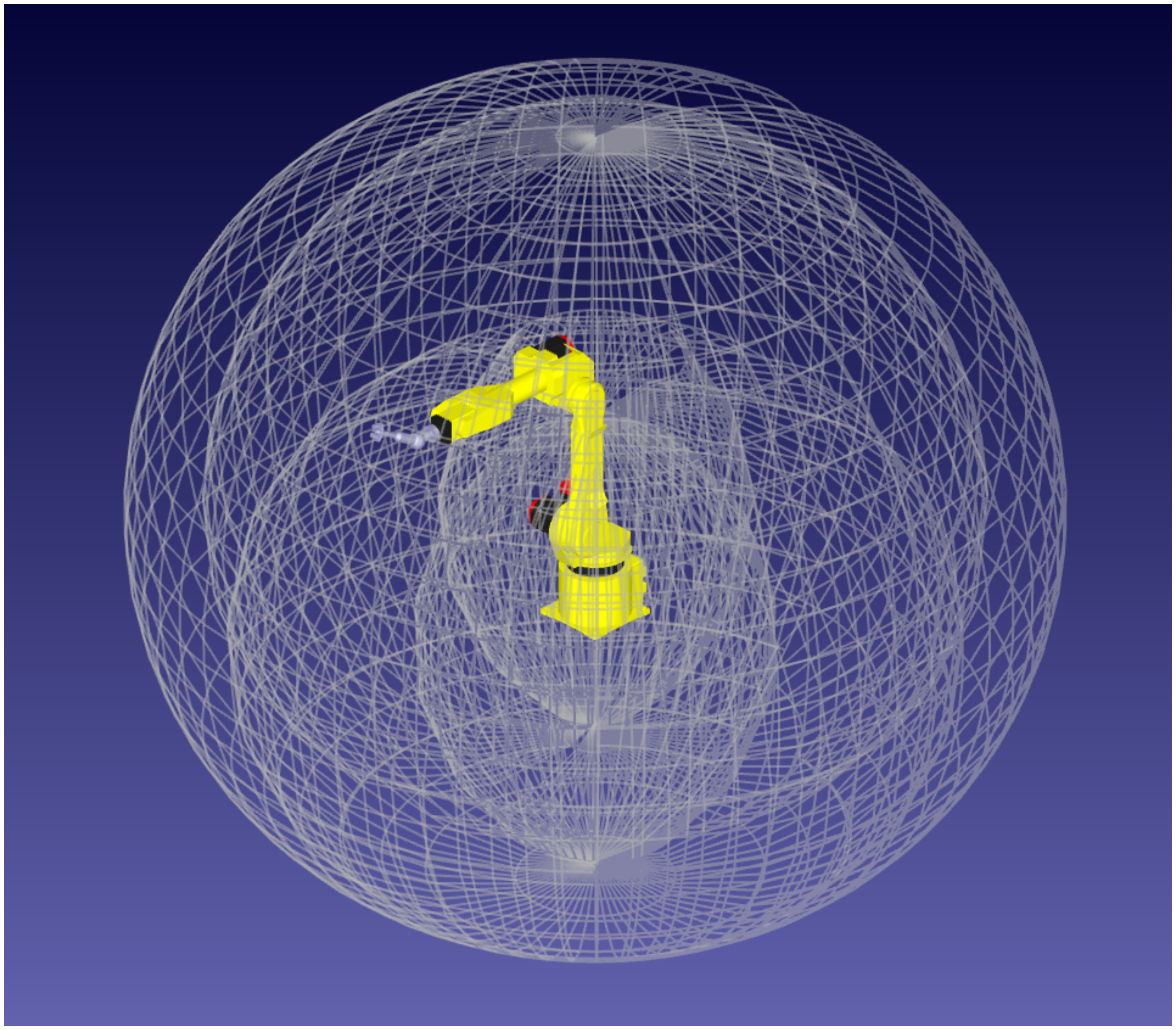


Figure 2.3 Fanuc 100ib workspace environment with respect to the NDE end-effector.

infinite number of solutions that are bounded within E. Inverse kinematics may be defined from the function:

$$F_{ik}(O^0_{n,j}, DH) = J \tag{2.7}$$

This means that after holding F_{ik} from an orientation from the base to the final end-effector frame $O^0_{n,j}$ will supply the joint set. Solutions for inverse kinematics become more complicated when considering most best joint movements compared to an earlier transformation, or with obstacle avoidance in which some optimal options must be ignored for ones that prevent collisions [142]. Inverse kinematics solutions have been studied for NDE [111]. Because of the complications of optimally solving inverse kinematics, RDK is extensively used for deciding solutions. Collision detection, which checks if the robot or attached end-effectors intersect with any part of the environment, is useful for preventing damage to the robot or its surroundings. How collision detection works will not be covered in this dissertation, as it is done within RDK.

2.2.3 Triangulation of Mesh Faces

Triangulation should preserve the surface profile but not change the location of any vertices. Triangulation is important to finding normals, discussed in the earlier section, and to allow for ray-triangle intersection, discussed next section. Plenty of optimized algorithms exist for triangulation [16]. The algorithm itself will not be discussed, as many different implementations are used with some black boxed from reconstruction sensor libraries. What is relevant is what properties occur after triangulation. First, a triangle is a face, defined as three vertices, the least needed to define a face. The definition for face f_{i_F} from equation 2.1 with respect to subfaces tr_i, with L_t for the number of vertices in a triangle, n_t being the number or triangles per within a face:

$$tr_i = [v_{a,i,F}, v_{b,i,F}, v_{c,i,F}]$$

with $L_t = 3$. This is a simple definition of a face, saying that a triangle explicitly has three vertices indexed within it. Equation 2.9, for normal calculation, requires the definition from 2.2.3. A set of triangles T_{i_F} with respect to a face f_{i_F} is defined as:

$$Tr_{i,F} = [tr_{1,i,F}, tr_{2,i,F}, ..., tr_{L,t,i,F}] \tag{2.8}$$

with:

$$Tr_{i,F} \subset f_{i,F}$$

This shows that after a face is triangulated, each output triangle is a subset of its related face. The number of triangles is limited to the number of vertices L_f inside the face.

$$n_{tr} = L_f - 2$$

For example, if the face is a rectangle or $L_f = 4$, then $n_t r = 4 - 2 = 2$, so two triangles are needed to define each face. Finally, since each vertex within the subset of triangles is shared with its parent face, the summation of all vertices may be defined in two ways. First, if each vertex is not indexed, or each vertex is unique, then the number of vertices on the mesh n_v, with full set of triangulated faces Tr so $Tr \subset M$ and the number of triangulated faces $n_{f,tr}$:

$$n_v = \sum_{0}^{n_{f,tr}} Tr = 3 * n_{f,tr}$$

This is not optimal, since vertices in similar positions should not be duplicated within memory, which takes unnecessary space. Instead, vertices should be indexed to prevent this duplication, meaning that vertices are shared. With no duplicate vertices from triangulated faces with respect to their parent faces:

$$n_v = \sum_{0}^{n_{f,tr}} Tr = \sum_{0}^{n_{f,tr}} F$$

This means from the number vertices from the original set of faces F before triangulation is the same as the number of vertices after triangulation. Setting up the second definition is more complex since it requires changing the indexing from the original mesh.

2.2.4 Normality and Rotation from Triangular Faces

Normal vectors are useful for deciding the orthogonality of a point holding rotation properties, such as for waypoints. A normal vector is a unit vector which is defined to be perpendicular to an associated surface [180]. A normal may be defined with respect to Cartesian coordinates:

$$n = [n_x, n_y, n_y]$$

With the property:

$$|n| = 1$$

A probe in which path planning defines its manipulation should try to be normal to the surface of the mesh, in which using a normal vector as a baseline for rotation should prove useful in this regard. Algorithms for waypoint generation will be discussed later. This rotation is also useful for deciding direction of lift-off for waypoints. A normal vector n_{i_F} is defined per face and holds three elements determining orthogonality per Cartesian axis. Normals may be explicitly defined on mesh generation. For the sake of this discussion, normals will be generated from the normalized cross product between three vertices on the associated face:

$$f_{i,F} = [v_{a,i,F}, v_{b,i,F}, v_{c,i,F}] \tag{2.9}$$

Using cross product result c:

$$c = v_a v_b \times v_a v_c$$

and solving the normal, which is the definition of a unit vector [178]:

$$n_{i,F} = \frac{c}{||c||} \tag{2.10}$$

This approach for solving normals is used within lighting engines to quickly find "flat shading" based on the orthogonality of the face [176]. The purpose here, however, is to use orthogonality to determine rotation of a point on the face. For solving rotations, it should be noted that in the code developed, after a normal is found, then "normal to quaternion" then "quaternion to rotation matrix" functions are deployed via OpenGL/OpenTK [88], which may not be the most optimal approach, however this does not significantly affect performance of the code. Since results use this method, this conversion process will be explained to preserve its methodology.

A quaternion q is a four-element vector which is useful in representing a rotation, in this case will preserve "axis" and "angle" information [54]. The angle and axis will use the unit vector along y, defined as $N_y = [0, 1, 0]$, and the normal vector N [57] [55]. Angle θ may be solve by taking the dot product between the unit vector and the normal vector:

$$\theta = acos(n_y \cdot n)$$

The axis a is determined from the cross product:

$$a = n_y \times n = [a_x, a_y, a_z]$$

To solve the quaternion from axis and angle calculations [58] [55]:

$$q = [q_x, q_y, q_z, q_w]$$

$$q_x = a_x * sin(0.5\theta)$$

$$q_y = a_y * sin(0.5\theta)$$

$$q_z = a_z * sin(0.5\theta)$$

$$q_w = cos(0.5\theta)$$

Quaternions may then be converted into Euler rotations [21] [17]:

$$\alpha = atan(\frac{2 * (q_w q_x + q_y q_z)}{1 - 2(q_x q_x + q_y q_y)})$$

$$\beta = asin(2 * (q_w q_y - q_z q_x))$$

$$\gamma = atan(\frac{2 * (q_w q_z + q_x q_y)}{1 - 2 * (q_y q_y + q_z q_z)})$$

$$R = [\alpha, \beta, \gamma] \tag{2.11}$$

Finally, Euler coordinates may be converted into a rotation matrix using equation 2.3, though this multiplication may be optimized [158]. There are certain mesh requirements with usage of equation 2.10: this equation should be applied to each face on the mesh, each face should be a triangle holding three vertices. This means that for using this equation, the mesh should be triangulated, discussed in the next section.

2.2.5 Intersection on Triangular Faces to find Translation and Normals

Ray-triangle intersection decides the location of a point on a triangular face. The algorithm deployed is based on Möller–Trumbore's algorithm [124]. In short, the algorithm checks every face of a mesh and will determine if a ray collides with any face. If a ray intersects, then the position at the intersection on the face and ray is returned. A modification to this algorithm was deployed, which on intersections the normal is calculated from the intersected face, solving equation 2.10 for rotations. This means any intersected point can be used to find position and rotation on the mesh, which is needed for the implementation of path planning used. If there is no intersection, then simply an intersected point will not be generated. This allows for robustness if several rays are intersected on the mesh, perhaps in a grid fashion which is how path planning is incorporated, and if the sample holds portions that the grid will not intersect at, such as if it were to be in air, then the probe will not try to move to those locations. More on path planning later.

Per intersection, there are several checks that are needed. Order of operation matters for optimization purposes. If a *null* is returned per triangle, this shows that an intersection did not occur at that triangle but may occur at another. This also stops the intersection check at that face and continues checking other faces. Every face of the mesh is checked. If no intersection is found at every face, *null* is returned as the result. Otherwise, the position of the point p and its normal N is returned. The algorithm assumes a non-culling approach. For any triangle with $t = [v_a, v_b, v_c]$ and ray defined from start $p_{r,start}$ and directional normal vector d_r:

1. Find the determinate the determinate d from two shared edges e_1 and e_2:

$$e_1 = v_b - v_a$$

$$e_2 = v_c - v_a$$

$$p_a = d_r \times e_2$$

$$d = e_1 \cdot p_a$$

2. Compare determinate d to detect backwards triangles. Return *null* if the determinate is less than zero:

$$d < 0 \rightarrow null$$

3. Compare determinate d if the ray is parallel to the face. Using small number ϵ. Return *null* if the determinate if the determinate is between ϵ:

$$d > -\epsilon \rightarrow null$$

$$d < \epsilon \rightarrow null$$

4. See if u intersection is outside the triangle:

$$d^{-1} = 1/d$$

$$t = p_{r,start} - v_a$$

$$u = (t \cdot p) * d^{-1}$$

If u is within either range, return $null$:

$$u < 0 \rightarrow null$$

$$u > 1 \rightarrow null$$

5. See if v intersection is outside the triangle:

$$q = t \times e_1$$

$$v = (d_r \cdot q) * d^{-1}$$

If v is within either range, return null

$$v < 0 \rightarrow null$$

$$u + v > 1 \rightarrow null$$

6. Find the point of contact p, after previous checks for optimization purposes:

$$p = (e_2 \cdot q) * d^{-1} \tag{2.12}$$

7. After solving p, if the point is not within bounds, return $null$:

$$p < \epsilon \rightarrow null$$

8. After solving p, ensure the point is within the robot's workspace E. If not, return $null$:

$$p > E \rightarrow null$$

9. All checks have passed. Calculate the normal N from equation: 2.10. Return p and N.

Further optimizations may be deployed, such as a boundary box check if a mesh or set of vertices are within range before running the intersection algorithm, but this wasn't implemented within the code used for this dissertation.

2.2.6 Lift-offs from Waypoint Sets

Lift-offs require transformations applied to waypoints. Two types of lift-off transformations were examined: a global translation to the complete set, and local transformation based on the normal of the intersected face. So, the desired waypoint $p_{w,off}$ shifting a global translation vector T_{sh} from an initial waypoint p_w would be at:

$$p_w = [T, R]$$

$$p_{w,off} = [T_{off}, R_{off}]$$

$$T_{off} = T + T_{sh}$$

$$R_{off} = R$$

Global translation is simple, in which a Cartesian shift may be applied along any desired direction. If a sample requires 5cm lift-off across the sample, and the sample is on a table, then a translation of 5cm along z is applied to every waypoint in the list. This method, while simple, has a few limitations. First, this method requires a global translation $T_s h$, which an "approach" angle should be defined that rotates $T_s h$ away from the set of points. This potentially requires an extra parameter for a human operator to input. Defining an approach angle may not be easy to objectify, and as a result, if a global translation for lift-offs is used, they will be along one axis. For example, the z-axis may be defined so only translation occurs along the z-axis while $T_{x,off}$ and $T_{y,off}$ are equal to zero. Finally, this approach, while effective for flatter samples, tended to perform poorly for curved or complex samples. For example, when used with a cylindrical pipe sample, the results performed poorly as lift-off become high and non-constant on the side areas of the pipe. The rotation may be off as well, potentially being orthogonal to its desired transformation! This contradicts the need for autonomous examination of complex materials, so another lift-off category was devised: local transformation from facial normals.

To counter this, a second method was used which positions the waypoint a set lift-off distance away from the surface of the mesh. This allows the preservation of the surface normal vector so each waypoint points at the surface while maintaining a constant desired liftoff. To achieve this, the following is performed with a scalar lift-off z_{sh}:

$$T_z = [0, 0, -z_{sh}]$$

This places the lift-off locally away from a desired point. An orientation matrix O_{sh} may be found from applying the rotation of the waypoint R that is respect to the normal vector of the associated face to T_z:

$$O_{sh} = R * T_z$$

This orients T_z along the normal. Finally, to solve $p_{w,off}$:

$$\begin{gathered} O_{sh} = [T_{sh}, R_{sh}] \\ T_{off} = T + T_{sh} \\ R_{off} = R \end{gathered} \tag{2.13}$$

The waypoint rotation should already be oriented with respect to the normal of the face, so R_{sh} is unused. Because the locality of a rotation from an individual waypoint R, each point may rotate along different paths. Discussion on the performance of this algorithm will be provided.

2.3 NDE Block, Point cloud, and Mesh

With knowledge of cyber and physical interfacing and how to actuate the NDE probe, it is now proper to describe the structure of the final output of the system. The NDE scanning section will discuss the process of scanning within this framework, which will require synchronization between the NDE data acquisition system and the robot's controller for obtaining position. The goal is to obtain a PTC of NDE information generated from this synchronization. An NDE points cloud requires a set of data points D_{nde}, with local data point $d_{nde,j}$ with j from 1 to PTC size m. Each $d_{nde,j}$ holds three pieces of information: the transformation output from the robot controller from the sensor as end-effector O_j, the current NDE data $Block_{nde,j}$, and the color of the point $C_{rgb,j}$ which is dependent on $Block_{nde,j}$ and a selected colormap N_{map}. O_j contains transform information T_j and R_j.

The output of O_j is dependent on the path planning output in which the robot will move the sensor towards. O_j are not equal to waypoints, as transformations of the sensor are obtained in between waypoints as well. $Block_{nde,j}$ is an abstract section of data that is dedicated to obtaining a single scalar value that is dependent on the NDE technique deployed. For this thesis, ECT is used, which will output complex impedance in terms of real and imaginary. Only real is considered for simplicity. Though the output from the data acquisition (DAQ) device will be a vector of voltages against time per transducer excitation, this should be shrunk into a single scalar value as saving thousands or perhaps hundreds of thousands of full ECA signals per point per coil will heavily tax memory and saving information. If an array system is used, then O_j will need to adjust for the position of each local sensor on the array. When a scalar value is obtained per point, a color on the PTC $C_{rgb,j}$ may be placed, which will help visualization of defects. $C_{rgb,j}$ is based on the value of $Block_{nde,j}$ and its position of a colormap N_{map}. N_{map} may be defined in two ways: a static min and max structure in which the expected NDE data is bound inside, or dynamically which decides finds the min and max data of the complete set of points. For the latter, either the min and max can be found at the end of the scan, or dynamically from real-time scanning. Both the color and associated NDE data should be saved, so the data may include visualization for a human inspector and damage analysis from the NDE data. The number of points m within the PTC is dependent on the length of the scan, the time of scan, and the data acquisition rate. Though synchronization will be covered later, it should be noted that synchronization will wait for both the NDE acquisition device and robot controller end-effector transform output to save data, meaning the slower of the two devices will decide acquisition rate. Once all the points inside D_{nde} are solved, they are compared with N_{map} to create the PTC P_{nde}. Through reconstruction methods such as Poisson reconstruction, P_{nde} may be converted into a mesh M_{nde}, which helps in visualization by adding faces which interpolate the color structure and may be associated with the input environment mesh used initially for path planning. Figure 2.4 shows the hierarchy of the data system.

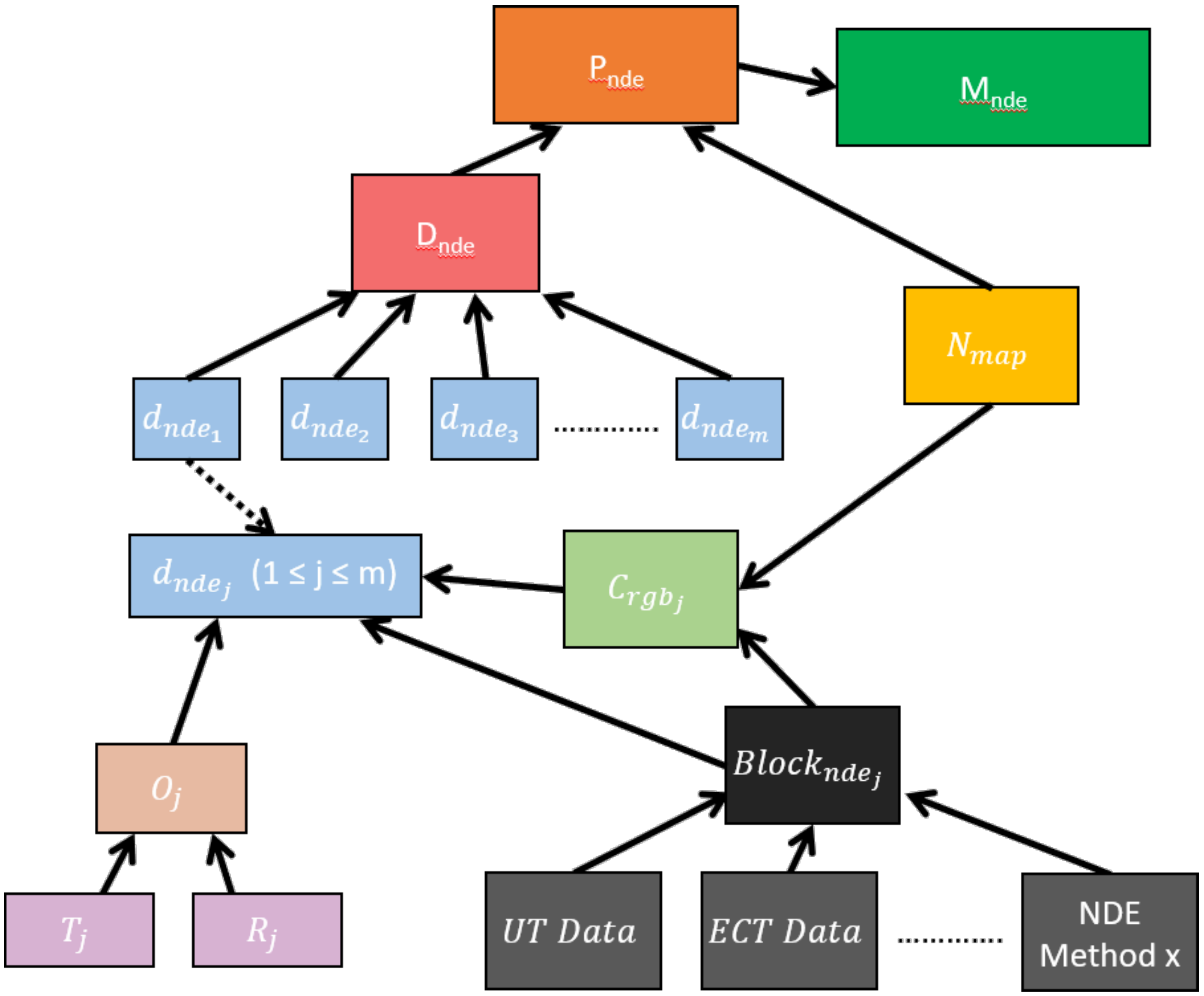


Figure 2.4 NDE Image Data Hierarchy.

CHAPTER 3

ENVIRONMENT RECONSTRUCTION

The previous chapter outlined the basics of a point p, a PTC or list of points P, and a mesh M. As discussed, the environment E is broken down into two categories: the sample itself, everything else as background. The points and faces which define the face should be used for path planning, while background components may be used for collision detection within the simulation software. Predefined CADs for path planning are avoided, as both the positioning and topology of the predefined virtual object may differ from physical space or may be difficult to calibrate. As seen in the literature review section, CADs see their purpose for autonomous NDE scanning, however the vision of this thesis is to allow for objects to scan to quickly move inside the environment, then for it to be scanned autonomously. This chapter discusses reconstruction methods to obtain physical typologies from several different methods, such as stereo camera, structured light (SL), and from the NDE probe itself. Registration of the mesh from the local reconstruction sensor back to the robot's frame is an essential step covered in the next chapter. This enables knowledge of the mesh's location within the robot environment where path planning of the NDE probe will take place.

3.1 Introduction

A reconstruction sensor should return the depth profile of objects in front of its sensing area, creating point cloud (PTC) P. P will be used to generate M, which will enable surface path planning from the object under test after it has been registered or aligned to the robot's based. Much discussion will be on the PTC P itself. Discussed in this thesis, reconstruction devices may either obtain a 3D PTC from a single snapshot or recording of PTC information at one spot of time or record individual depths with a recorded position of a depth sensor as a reconstruction device. The latter may be a 1D laser profilometer with variable results. To reconstruct over areas with 1D depth sensing modules, it should be moved along a path with its position recorded. In other words, the depth sensor should be used as an end-effector from a path that will record an area along the environment to reconstruct. This may be a raster-like path above the sample. This sensing area may be defined in several ways. It should be noted that the PTC P is not a 2D array, but a 1D list of points which lie within 3D Cartesian space. The organization of points from a snapshot obtaining device will be in a 2D grid like pattern with differing depths, while the pattern from a 1D depth system is entirely dependent on the path defined from the scan patrolling the sensor along the environment.

Field of view (FOV) is the range along local Cartesian coordinates x and y and width w and height h in which points will be generated. Using a definition where x and y are centered from w and h respectively, the range of x and y are defined as:

$$-w * 0.5 < x < w * 0.5$$

$$-h * 0.5 < y < h * 0.5$$

FOV may also be found by the viewing angle for each axis. The definition of FOV depends on the interpretation of sensor intrinsic parameters, such as if the output should always hold a static w and h, or changing depending on depth distance z. The range of z is also different for different devices, defined as:

$$0 < z_{min} < z < z_{max}$$

Depth z should enable high accuracies to ensure more correct profiling of the physical surface in digital space, which in turn should improve path planning accuracy. Minimum distance z_{min} of the region to reconstruct cannot be equal or less than zero, as that would show that region being behind the sensing device or touching the device. Typically, there is a conservative range for z_{min} so multiple points may be picked up, though this depends on the type of sensor used. For reconstruction sensors picking up 3D PTCs, enough information needs to be parsed to accurately obtain a profile, which requires a certain distance z_{min} to be defined. For z_{max}, a similar tale is told which if the sensor cannot interpret items at a distance, then they will simply not be picked up. The range for z is important to acknowledge within the application. For depth sensing devices, many devices may be used that can measure up close, perhaps up to a several microns [97]. However, as discussed later, the method for reconstruction will use a blind approach, which options that are too close to the sample without knowledge of the surface may lead to unwanted collisions between the laser or robot and the environment to reconstruct. This means it's more preferred to have a high accuracy but within range for z_{min} and z_{max} for 1D sensors. For this thesis, a Fanuc 100ib robot arm is used, which has an approximate range of $6m$. This means that it may be ideal to have a range within $z_{max} = 6m$, though any higher that supports high accuracy may be used as well, and z_{min} being not intrusively high. One may wonder if it may be possible to increase the area and resolution for the physical to digital reconstructed environment. This is possible by obtaining multiple snapshots or depth scans then parsing them together after registering them each, either to a local frame from one of the PTCs, or to the robot's base frame. Multiple snapshots are discussed in the next chapter.

Resolution is the number of points obtained per axis. Higher resolutions may lead to higher fidelity in reconstruction, however in practice, sometimes oversampling leads to "bumpy" surfaces which overcompensate NDE sensor positioning. A PTC from a 3D PTC snapshot device will have a resolution along x and y, defined as rs_x and rs_y. For example, the Intel RealSense D435i camera used in this thesis has a resolution of 1280×720 pixels (px) bounded within w and h, though it also has an RGB camera of 1920×1080 px for surface texture generation. This definition is for a single PTC, in which multiple PTCs may be used to expand the reconstructed region or be used for temporal processing. For 1D depth sensing modules, resolution is dependent on the number of points obtained during the depth sensing scan, which is dependent on the time taken to conduct the scan, dependent on speed of probe, speed of the depth DAQ, and number of raster swipes (for raster scans).

3D PTC snapshot devices will now be abbreviated as "cam" or camera for short. To differentiate from other points and meshes, each will be abbreviated as points p_{cam}, PTC P_{cam}, and mesh M_{cam}. A single orientation of the camera, O_{cam}, will orient the entire PTC output. Figure 3.1 shows an ideal setup where a camera facing a flat panel will obtain a grid, with each point holding some sort of depth information. Likewise, depth sensing will have points p_d, PTC P_d, and mesh M_d. A lift-off should be selected so a measured depth z_d is within sensing range between z_{min} and z_{max}, dependent on the sensor used. The orientation of the depth sensor O_d exists on the waypath W_d, dictated by waypoints $p_{w,d}$. This example shows a raster scan pattern, which will be defined in the path planning chapter. The reading of the depth sensor z and the orientation O_d between waypoints will determine a point $p_d = [T_{p_d}, R_{p_d}]$. For the translation T_{p_d}, which is like equation 2.13:

$$O_d = [T_{O,d}, R_{O,d}]$$

$$O_{sh} = R_{O,d} * [0, 0, z_d]$$

$$O_{sh} = [T_{sh}, R_{sh}]$$

$$T_{p,d} = T_{O_d} + T_{sh}$$

This rotates the depth to the perspective of the device, then shifts the read translation with respect to the depth point's translation. The rotation of the point must be flipped, as the perspective of the point should orient towards the reconstruction device. Having a flipped rotation causes trouble for mesh reconstruction. To do this, simply rotate the point by 180°:

$$R_{p_d} = R_{O,d} * 180^\circ$$

When placing a point into a 3D engine, it technically doesn't have a rotation itself, but a normal per point N_d. This conversion requires a universal normal vector $N_y = [010]$ along the y-axis, and applying the rotation per point:

$$n_d = N_y * R_{p,d}$$

From the PTCs generated, meshes may be generated. Mesh reconstruction from PTCs will be discussed in the next section. Meshes M_{cam} and M_d are analogous, however there are subtle differences when it comes to reconstruction due to the structure of the input PTCs giving separate scalar fields (SFs).

There are some restrictions on usage of reconstructed environments. First, consider the type of NDE scan that will be used from the surface profile. For example, if ECT is used, then the probe will require to be remarkably close, within potentially 1mm, or even touching the surface. UT has looser requirements but still requires a lenient distance between the NDE probe and sample,

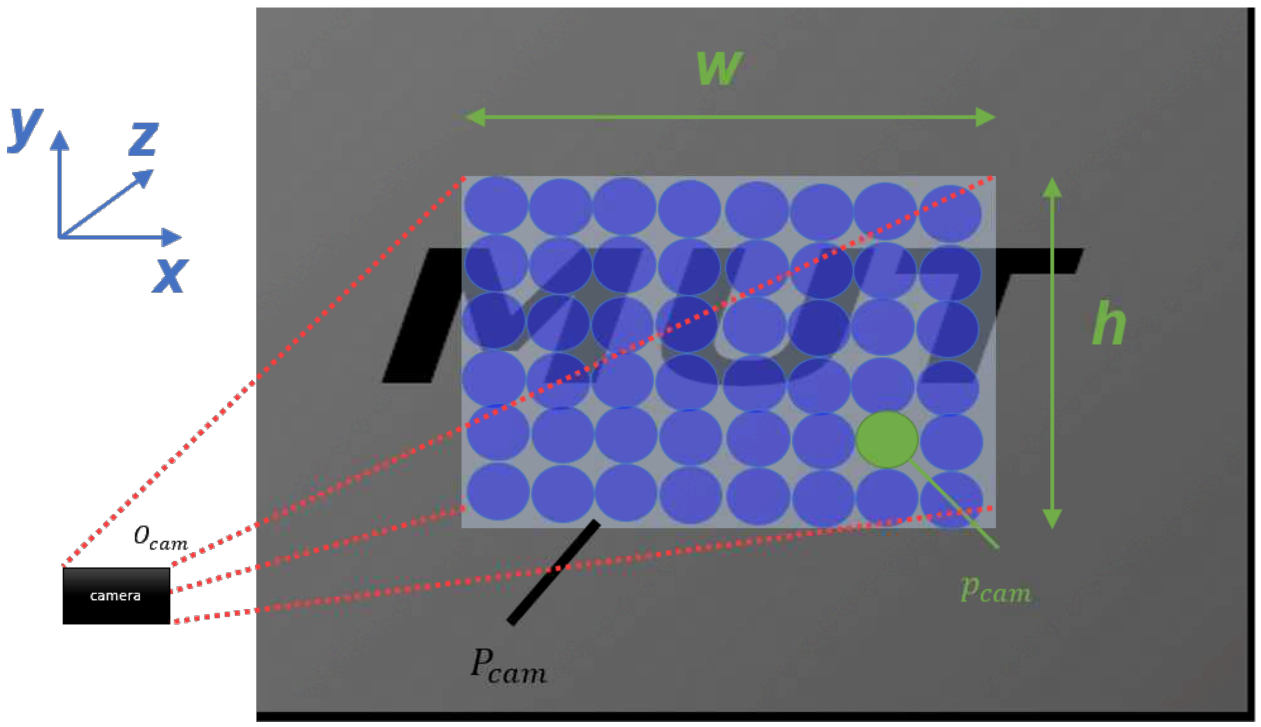


Figure 3.1 Camera orientation against a flat panel.

especially for subsurface scanning. In both scenarios, it is not wanted to touch the sample to prevent any damages. Error in registration or alignment is also significant in this regard. Erroneous rotations may also generate unwanted effects in the scan, for example in recordings of reflected signals in UT, which may add a layer of gain shifting in the output data that shouldn't be wanted. Another restriction is the requirement for the sample to not be positioned during the scan, meaning between the first snapshot of the reconstruction sensor to the last piece of information obtained from the NDE probe. If the sample moves during the scan, it will be difficult to track the amount of movement while also adjusting the scan path in real time during the scan. Miss positioning is a registration issue, which will be discussed in the next chapter. An example is if there is a collision of the probe or robot which may shift the sample within physical space. In a lab environment, if somebody touches or moves the sample in between scans, this will add unwanted errors in assumed positioning of the sample, creating unwanted effects in the NDE scan. It is recommended that the sample has measures to ensure it doesn't move during the scanning procedure, but applications where it may move quickly inside or outside the workspace may restrict this ability. Also, like

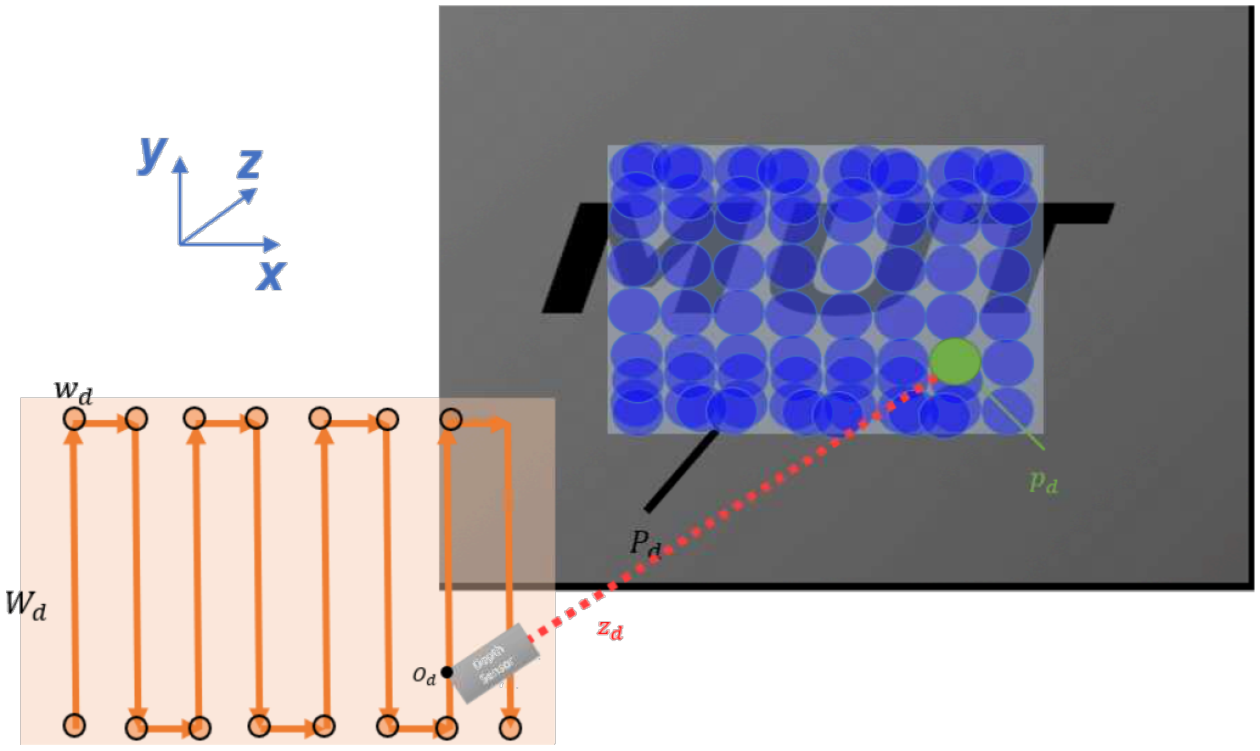


Figure 3.2 Depth sensor orientation from a waypath against a flat panel.

before, the sample must remain in a static location.

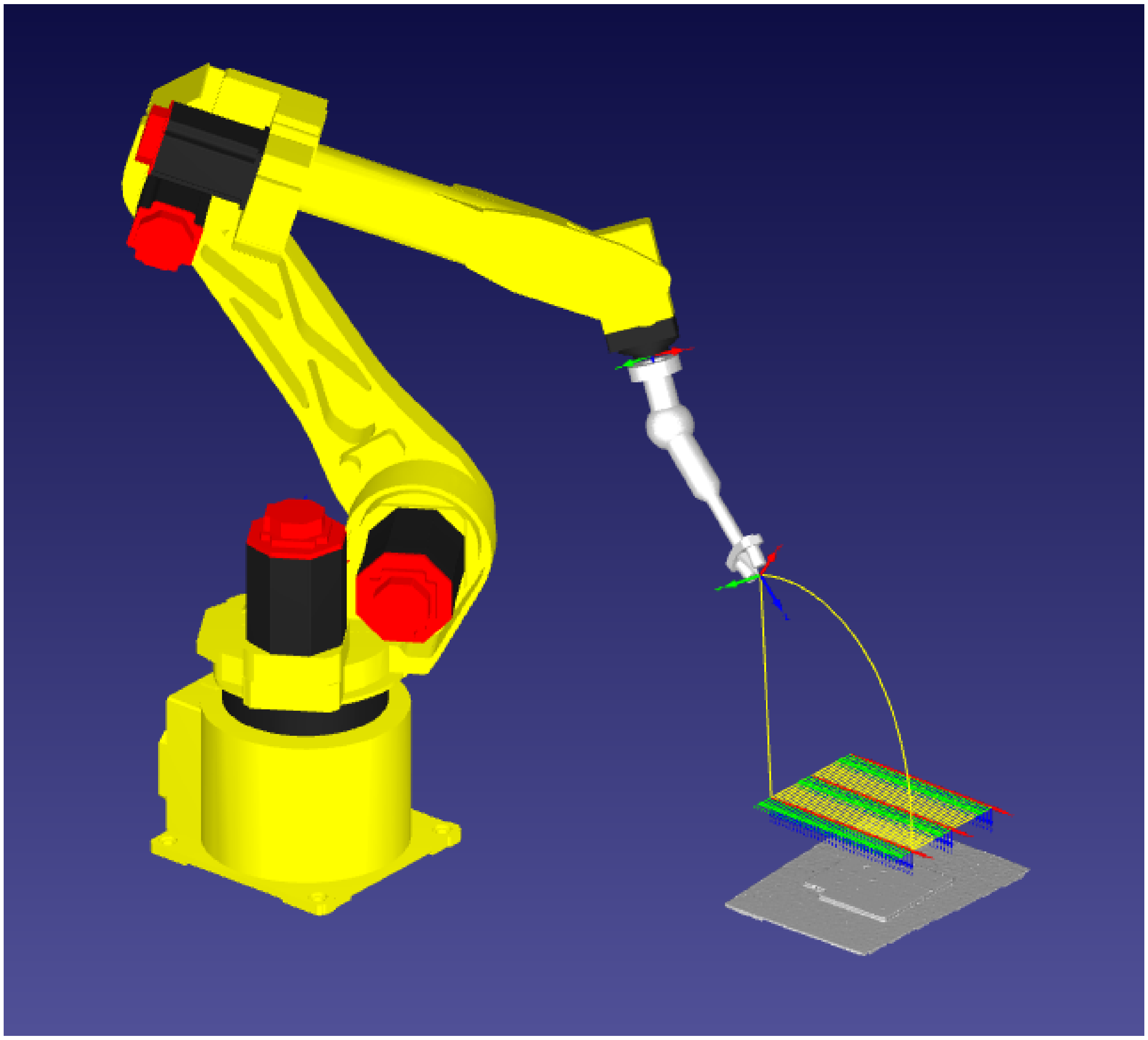


Figure 3.3 Depth sensor path planning in simulated RDK environment.

3.2 Mesh Reconstruction and Post-Processing

To obtain the physical to digital environment meshes M_{cam} from 3D snapshot sensors and M_d from 1D depth sensors that are suitable for path planning, the meshes should be reconstructed from a parent PTC P. For mesh reconstruction, the goal is to obtain a set of triangulated faces F from indices related to vertices. As explained in the last chapter, this enables parsing transformations on the surface of the physical surface interpreted in digital space through ray-triangle intersection, parsing the point of intersection and rotation from the normal of the face. On top of mesh reconstruction, there are recommended operations to remove unwanted points as statistical outliers, segment the sample from the background, and smooth the sample. Some of these steps may be processed on the PTC before mesh reconstruction, and some afterwards. Processing is done with CloudCompare and MeshLab, and both are automatable, with the C++ language interfacing for CloudCompare, and the Python language with MeshLab. CloudCompare handles registration, statistical outlier removal (SOR), and Poisson reconstruction, while MeshLab handles decimation and smoothing. Both are possible to meet within C-sharp, however CloudCompare has difficulties due to requiring wrappers. Figure 3.4 shows a recommended pipeline, however some operations may be added, removed, or reorganized. For example, registration may occur either on a PTC or on a mesh, which it may be easier to through point pairs picking to use a mesh instead since points may lay on each face rather than exclusively on points on the PTC. Registration or orienting the mesh to the robot's base frame is discussed in the next chapter. Another example is that many commercial software already outputs a mesh for the user which saves some implementation but those meshes might still need smoothing operations.

Using multiple PTCs post registration is useful for expanding the region of the scan, done by either moving the device at different approaches or using multiple cameras at different static locations. A merged PTC P_{merg} of an indexed PTC P_i from 1 to size n_P may be found through the summation:

$$P_{merg} = \sum_{i=1}^{n,P} P_i$$

This is not a summation per point, which would distort position of points, rather appending each point within any PTC P_i into an initially blank P_{merg}. As PTCs are linear 1D lists, P_{merg} will be an extended 1D list of points from each P_i.

After merging, typically there will be outlier data that may distort Poisson reconstruction if it is not removed. Outlier data may be information that is not intentionally picked up by the sensor, as random points or perhaps even wires or small components that appear in some PTCs but not in others. Poisson reconstruction outputs a SF per vertex within its generated mesh, which a histogram may be used to filter smaller components before the majority curve of the wanted. However, it is a better practice to cut unwanted points as they may generate alongside Poisson reconstruction

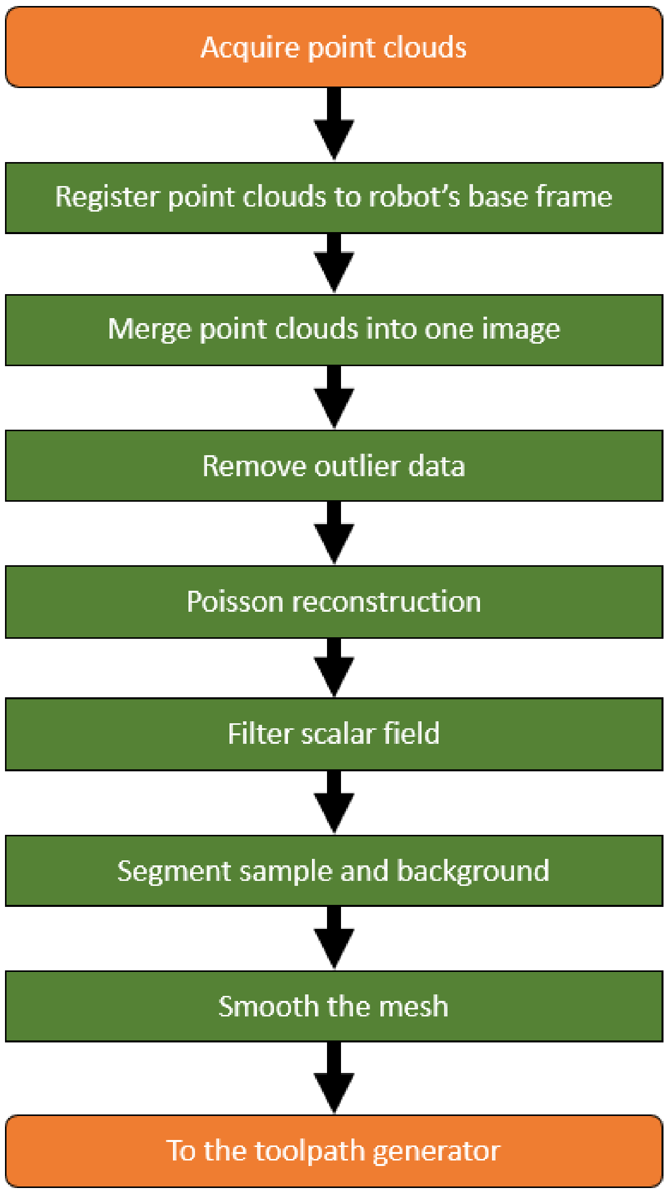

Acquire point clouds
Register point clouds to robot's base frame
Merge point clouds into one image
Remove outlier data
Poisson reconstruction
Filter scalar field
Segment sample and background
Smooth the mesh
To the toolpath generator

Figure 3.4 PTC to filtered mesh processes.

even after filtering the SF. Two methods were examined. First is to use an algorithm such as an SOR filter, which may remove individual rogue points or separate clusters of points not attached to the PTC. This process automatically removes points generally efficiently. The complication of SOR filtering is that points or clusters too close to the mesh may not be detected and may generate unwanted geometries on the surface to scan. An example is shown in figure 3.5, in which most points are removed, but some stay close to the sample region. This leads to the second method: cropping. Cropping is the process of manually selecting a region to scan and segmenting that region with the rest of the PTC. Simply after cropping, the unwanted segment can be removed from the PTC. The process of cropping is discussed when segmenting the sample and background. The drawback of cropping is that it is a manual process without the help of artificial intelligence, which may be overkill for this operation.

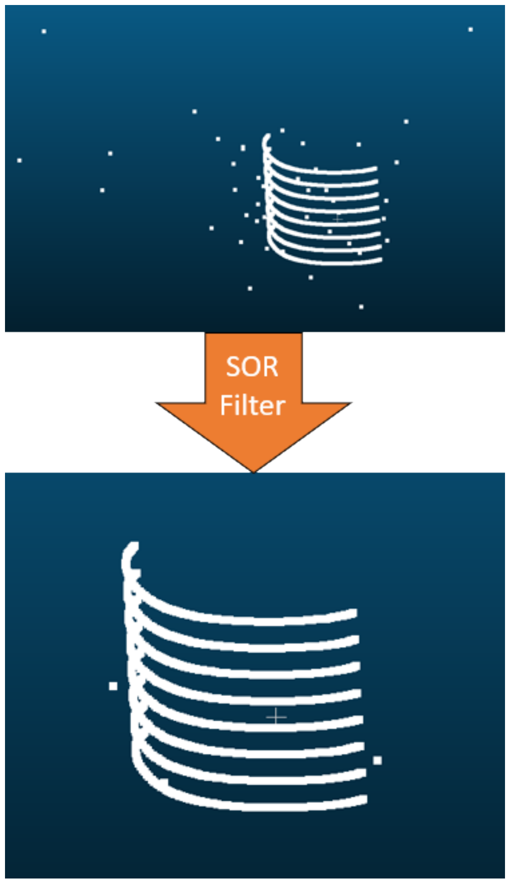
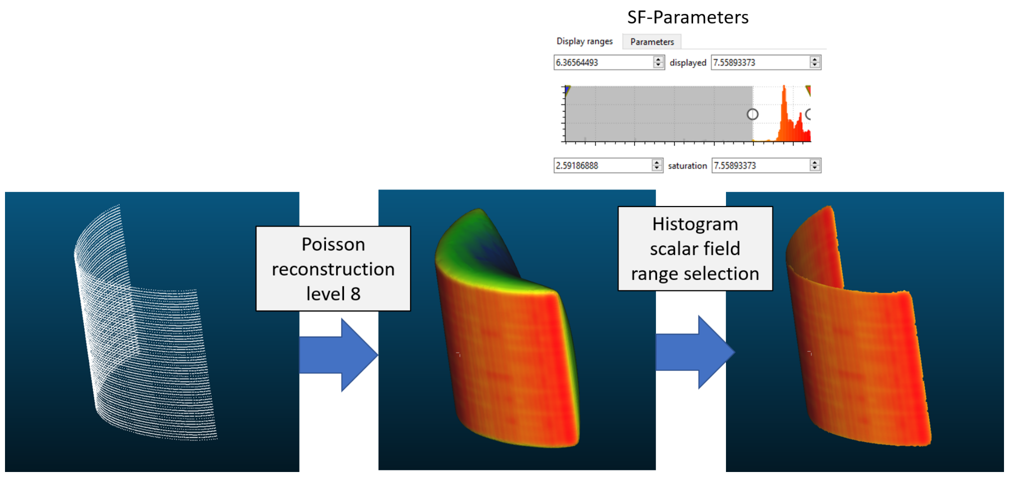


Figure 3.5 SOR filter example within CloudCompare.

Once the PTC is cleaned, the mesh may be generated via Poisson reconstruction. The raw output of Poisson reconstruction is a mesh which has SF values per vertex. Higher values of SF tend to show clustering of vertices, showing the area to reconstruct. There are several inputs that may be placed into this algorithm. One is the choice of boundary conditions, including Dirichlet and Neumann [93] [155]. Dirichlet tends to output more spherical boundaries, while Neumann tends to supply mode flat boundaries. Neumann is selected instead of Dirichlet due to consistency in providing results with less bumps of the edge of the output mesh, which Dirichlet boundaries may provide. Octree depth levels may be chosen, which lower values provide faster and smoother but less resolute meshes, while higher supply higher dense meshes at the cost of smoothness and computation time. Level 8 octree depths are universality used for any Poisson reconstruction in this thesis.


SF-Parameters
Display ranges
Parameters
Poisson reconstruction level 8
Histogram scalar field range selection

6.36564493 displayed 7.55893373

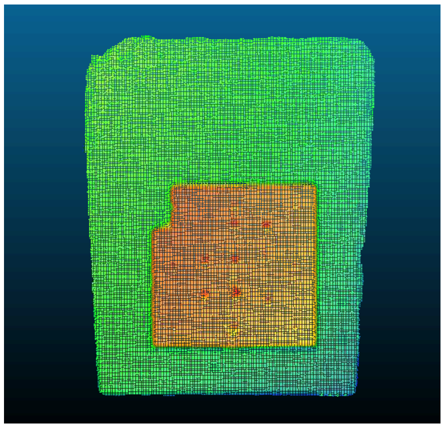
2.59186888 saturation 7.55893373

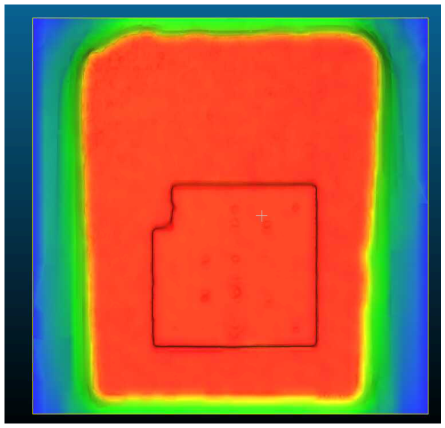
Figure 3.6 Poisson reconstruction example in CloudCompare.

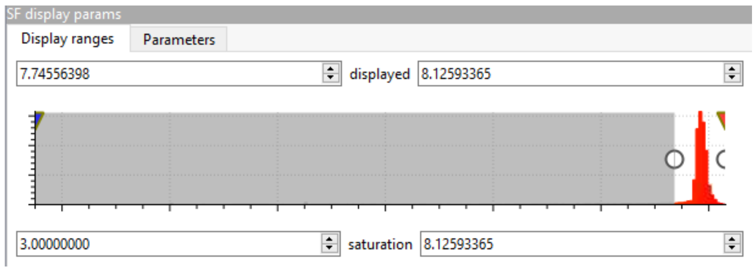
SF filtering is done by examining a histogram of SF parameters from the generated mesh, and removing vertices and faces that are on the lower spectrum of the histogram, showing vertices that are not clustered. An example is shown in figure 3.6. Usually, a peak will appear, indicating where the filter should not pass through. This process is automatable even by simply setting a threshold before the maxima, however this is currently done manually. In figure 3.7, the PTC is obtained through SL, which supplies a mostly uniform structure of the PTC, though "mostly" is used since multiple PTCs were used for temporal purposes. In contrast, 3.8 generated a depth PTC from UT depth scanning, which holds clusters based on the speed of the probe, which can be seen on the edges and middle on the unfiltered SF mesh. The start and stop locations also have the highest clustering. Despite this, it is irrelevant to the output mesh post filtering as only the lowest before the main spike is filtered.

Methods such as Delaunay are possible, but sometimes lead to mixed results. Delaunay can supply fast and automated reconstruction from planar meshes without filtering out SF parameters like Poisson reconstruction. Delaunay reconstruction may work alongside methods such as 1D depth raster scanning. The issue is that sometimes if the input PTC is not planar, which may occur if there is missing information, then the output may distort. Several comparisons between Dirichlet versus Neumann boundary conditions, or between Delaunay and Poisson reconstruction, are shown in figures 3.9, 3.10, and 3.11.

The final processes are the segment the sample and background, and to smooth the sample's geometry. The background should be kept, but only for path planning and collision detections, to prevent any damage to the sensor, robot, or surroundings. The background should be removed for the path planning however, as it may be intrusive for path planning with waypoints being generated in the background. Autonomous methods for segmenting PTCs exist [34] but would be difficult to currently implement and confirm within the current setup [131], and potentially unnecessary. Therefore, manual segmentation is used. There are two useful techniques as options inside of CloudCompare and MeshLab. One polygonal segmentation in which the user inputs a custom polygon from a certain camera perspective to segment out the sample from the background. This is useful if the sample and background are merged post mesh reconstruction. The second method is connected components selection, which may be used if the sample and background are not touching, or have connected faces, on the reconstructed mesh. Simply, the user selects the sample and background individually. For extra filtering, every other part exclusive to the sample and background, such as rouge clusters of faces, may be removed as the surfaces of the sample and background are known. Once the sample is known, it is recommended to smooth the surface to prevent rotation bias from rough surfaces generated through mesh reconstruction. Laplacian smoothing is used [160], generally with 10 iterations of smoothing. An example is shown in figure 3.12. Decimation is also used on samples with many faces, saving computation time in any processing including path generation used due to restrictions on the custom path generation engine where only $30,000$ faces may be used per mesh.


(a) Point cloud, RGB from depth with red closer to the viewer.


(b) Unfiltered SF mesh.

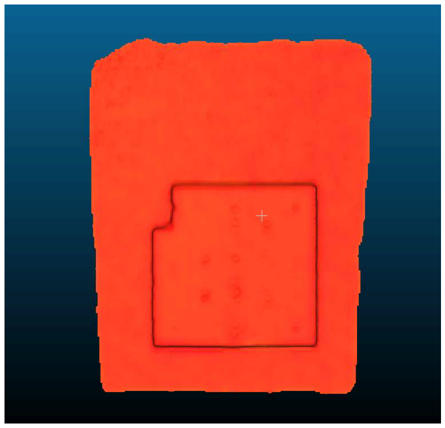

SF display params

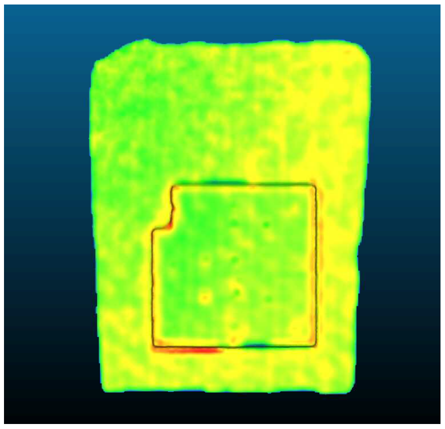
Display ranges | Parameters

7.74556398 displayed 8.12593365

3.00000000 saturation 8.12593365

(c) SF histogram and parameters within CloudCompare.


(d) Filtered SF mesh with previous SF coloring.


(e) Filtered SF mesh with updated SF coloring.

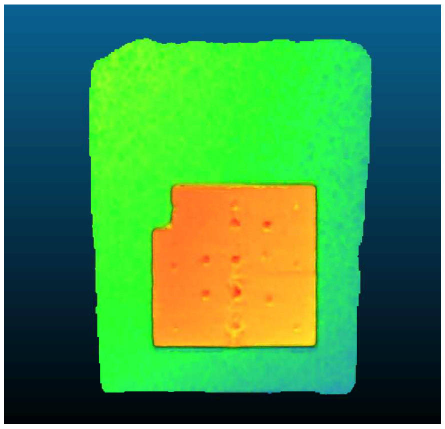
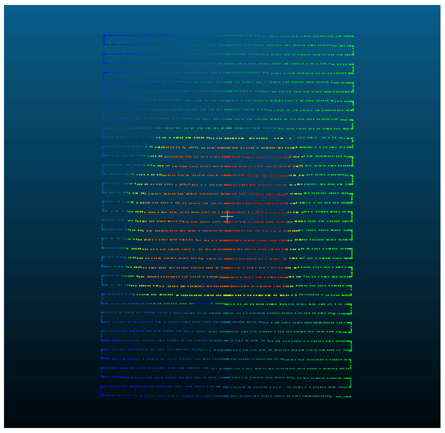
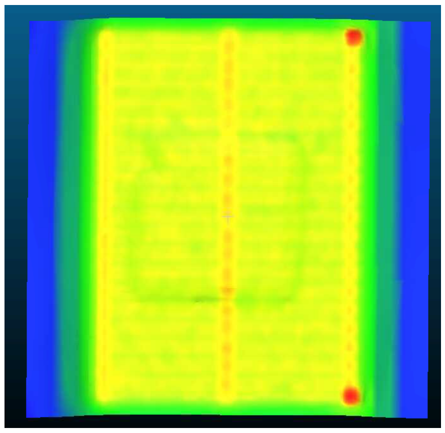
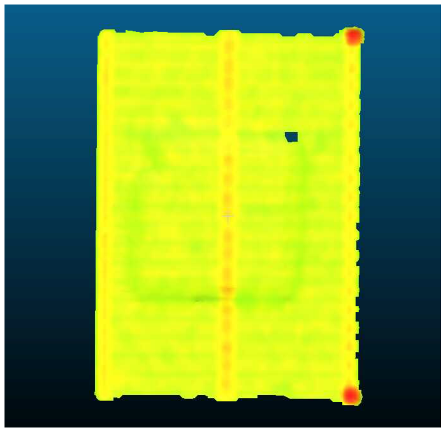
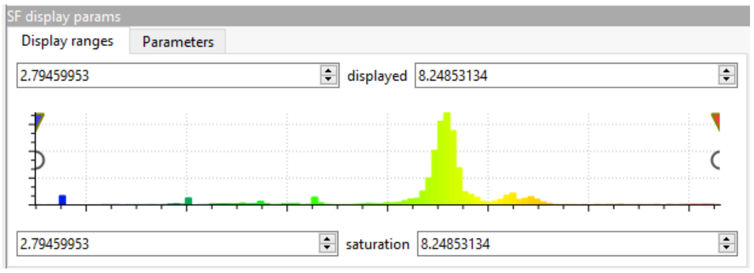

(f) Filtered mesh, RGB from depth with red closer to the viewer.

Figure 3.7 PTC to raw mesh process from SL on an aluminum sample placed on a table.


(a) Point cloud, RGB from depth with red closer to the viewer.


(b) Unfiltered SF mesh, with Neumann boundaries.


(d) Filtered SF mesh with previous SF coloring.

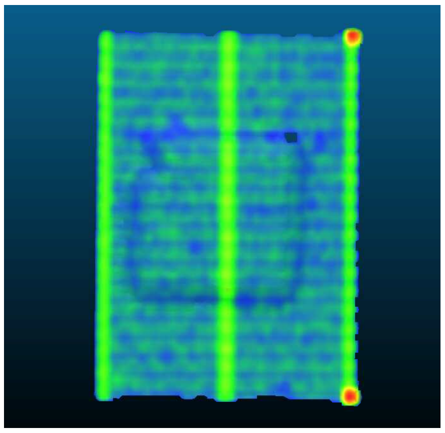

SF display params

Display ranges | Parameters

2.79459953 displayed 8.24853134

2.79459953 saturation 8.24853134

(c) SF histogram and parameters within CloudCompare.


(e) Filtered SF mesh with updated SF coloring.

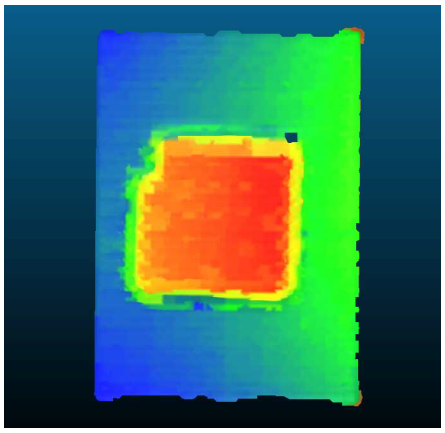

(f) Filtered mesh, RGB from depth with red closer to the viewer.

Figure 3.8 PTC to raw mesh process from UT sensor in depth detection mode, on an aluminum sample placed on a table.

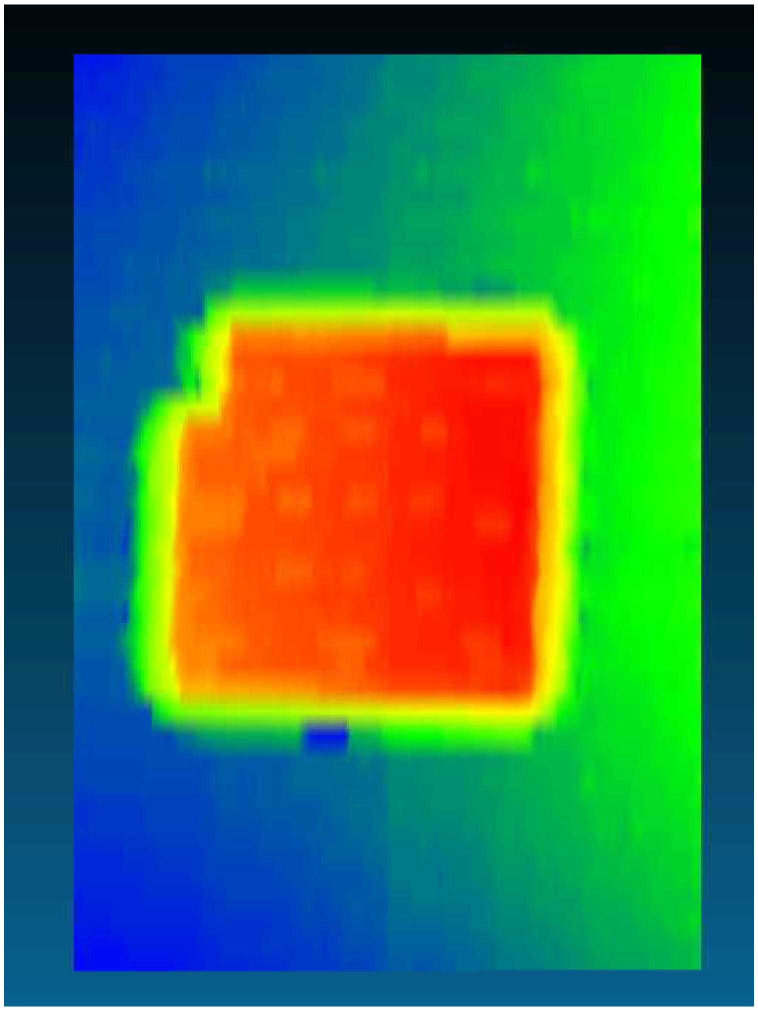
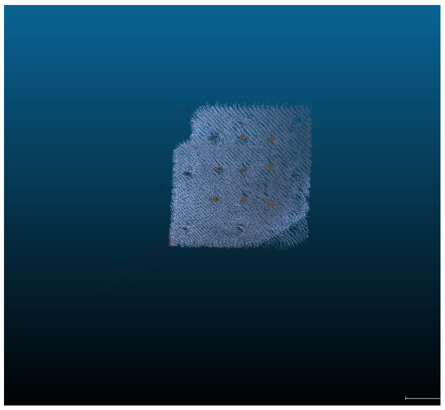
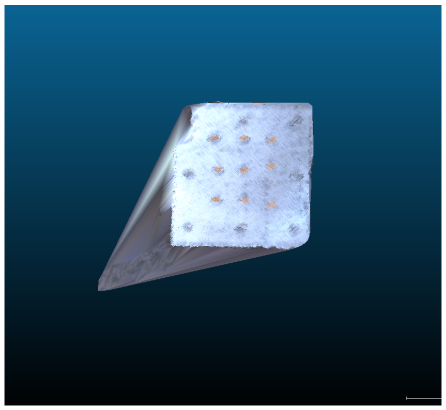
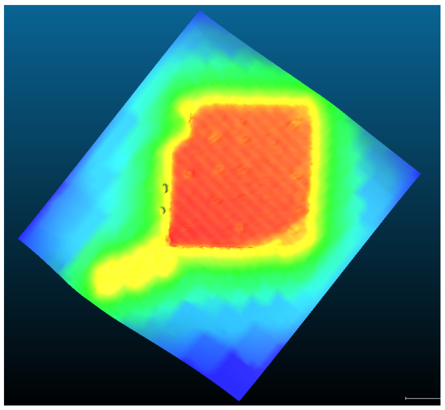


Figure 3.9 Delaunay 2.5D reconstruction, from PTC obtained from figure 3.8.


(a) Initial PTC from SL.


(b) Mesh reconstruction using Delaunay 2.5D function from CloudCompare.


(c) Mesh reconstruction using Poisson reconstruction with Neumann boundaries from CloudCompare.

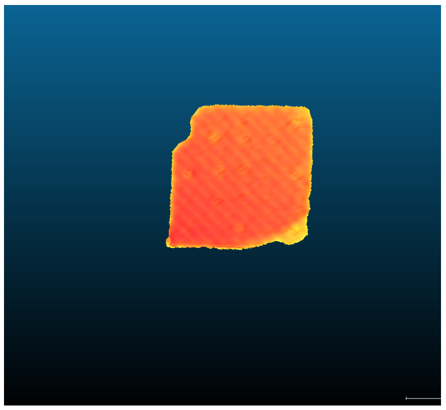
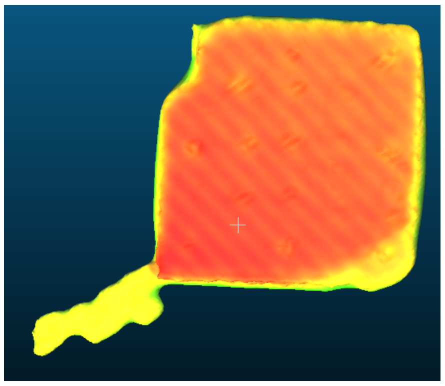
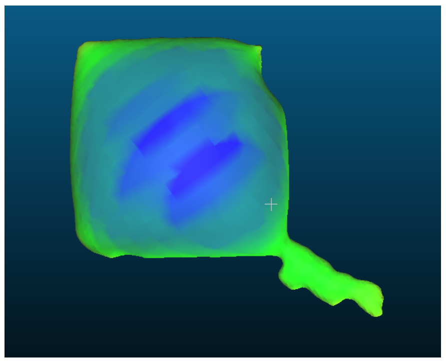

(d) SF filtered mesh.

Figure 3.10 Comparison of aluminum sample reconstruction from SL between Delaunay and Poisson reconstruction methods.


(a) Unfiltered reconstructed mesh.


(b) Back side of unfiltered reconstructed mesh.

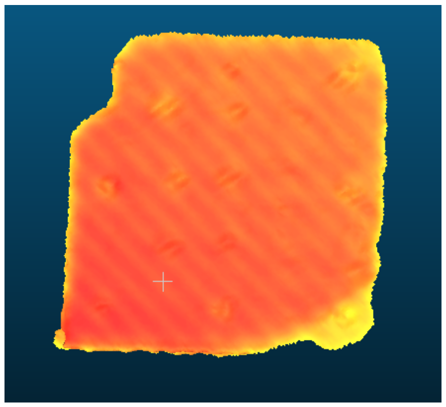

(c) SF filtered mesh.

Figure 3.11 Poisson reconstruction with Dirichlet boundaries, from the PTC in figure 3.10.

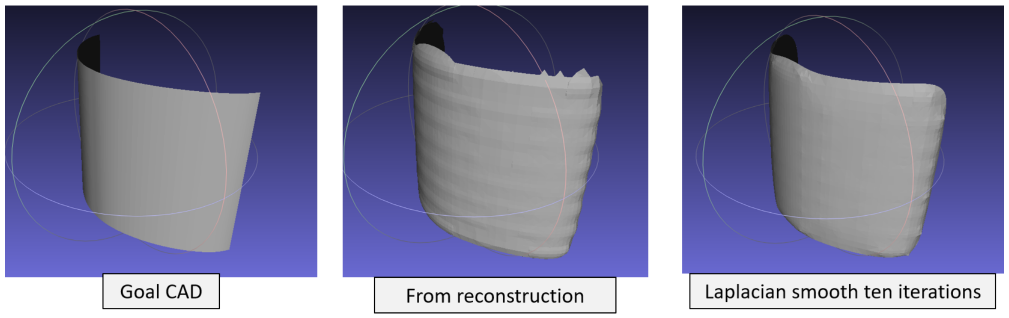


Figure 3.12 Laplacian smooth example in MeshLab.

3.3 Reconstruction Methods and Devices

Several different reconstruction methods have been tested, including stereo vision, SL, and single point raster scanning. Though stereo vision is eventually ignored due to mediocre performance in surface accuracy, it shouldn't be ignored for extended methods such as mobile applications due to its low cost and portability. Commercial options were explored for stereo vision and SL, with an Intel RealSense d435i used as a stereo camera and a Creality CR-Scan 01 used as a SL camera.

3.3.1 Stereo Vision

Stereo vision was the first method examined for surface reconstruction, shown in earlier works [76]. Stereo vision works like human vision, in which two cameras are used to perceive depth. How this works is that the two cameras obtain an image each at the same time, in which matching algorithms are deployed to obtain depth per point [102]. How matching algorithms are decided to come from camera intrinsic parameters. Stereo cameras supply fast, high resolute, and portable solutions for 3D PTC snapshots.

In the case of using the Intel RealSense camera, stereo vision is used alongside Intel RealSense's SDK, along with several filtering algorithms [6], [39]. Much discussion will use Intel RealSense as an example. The filtering process is shown in figure 3.13, which would be input to the processes shown in figure 3.4. Such processes allow for decreasing various noises with background data that may conflict with mesh reconstruction. In short, a filtered output will be generated:

1. Decimation filter: selects a kernel that immediately limits the number of faces per mesh. This is used first to decrease computation time. A median filter is used within the kernel for each depth.

2. Temportal filter: selects a simple depth range which any point outside with be removed from the PTC. This is done early to decrease computation time.

3. Depth2Disparity transform: converts the depth map to disparity, which is the reciprocal of depth, used for the next two sections of filtering.

4. Spatial filter: smooths data using various 1D horizontal and vertical iterations [68]. Is edge-preserving.

5. Temporal filter: uses past snapshots in order improve the final output to be closer correlated the the physical surface. This requires for the camera and recorded environment to remain in statics locations.

6. Disparity2Depth transform: converts back into a depth map.

7. Hole-filling filter: Attempts to find four neighboring vertices to remove any holes on the PTC to reconstruct.

The Intel RealSense SDK also supplies direct outputs of the mesh itself after these procedures, which may allow for reconstruction to be avoided.

There are complications using stereo vision, at least tested with the RealSense camera. Simply, the PTCs used for reconstruction are not correct enough for the high-precision applications needed from robotic arms. At 2 meters, the depth accuracy of the camera is 2 percent. At this distance, the accuracy is up to $40mm$, which is exceptionally large with respect to NDE scanning with a robot arm with much higher accuracy. Figure 3.14 and 3.15 shows results of complex shaped CFRP car pieces from different CFRP car pieces placed on a table, snapshots taken above using the Intel RealSense stereo camera. Upon visualization compared with SL in figures 3.16 and 3.17, the SL supplies more detailed geometric features. This affects path planning and output NDE results, as stereo cameras tend to distort edges significantly which makes scanning exceedingly difficult. Because of this, SL is examined instead for the rest of this thesis. For mobile applications, stereo vision may be more proper where positioning of the sensor may have a relatively similar accuracy restriction. Similarly, reconstruction methods such as photogrammetry, in which several images from differing perspectives are used to generate a 3D environment [118], may be useful for such applications [49].

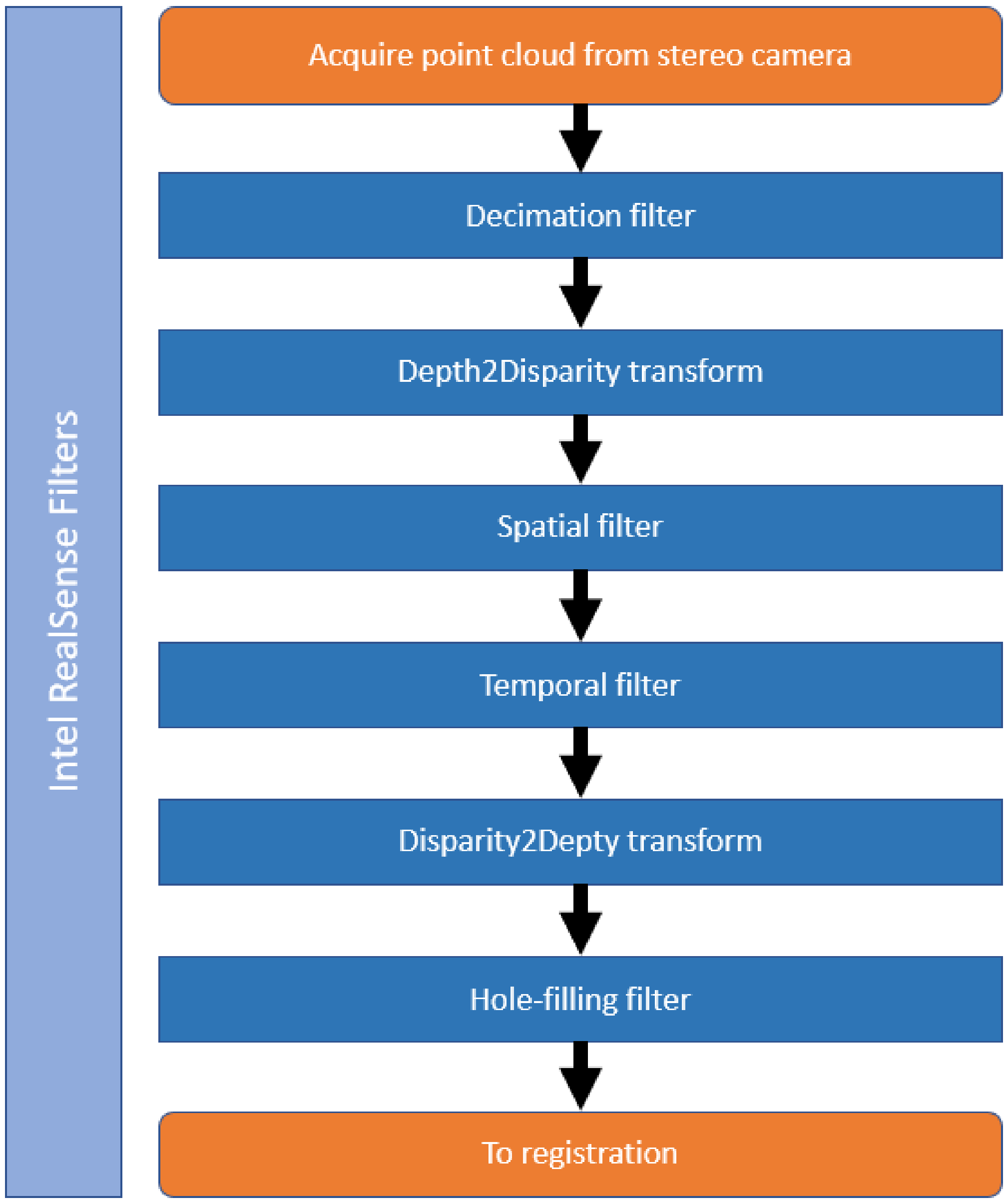
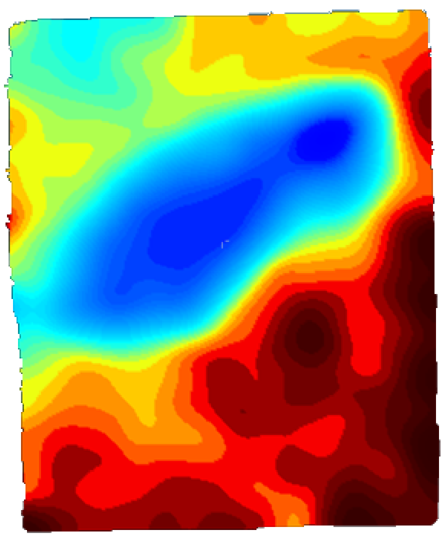
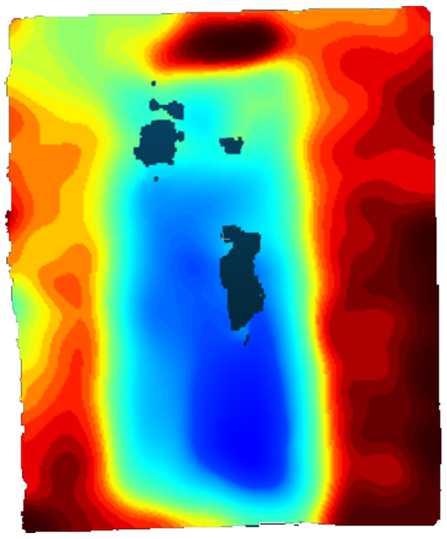


Figure 3.13 Intel RealSense filtering process.

3.3.2 Structured Light

SL for PTC reconstruction supplies high accuracy, cheap, and portable options that have seen uses for surface NDE and robotics [106] [5] [83]. SL uses light fields associated with spacial distributions, such as amplitude, phase, and polarization, to obtain features that would parse 3D environments [60] [7] [69]. A Creality CR-Scan 01 is used, which while typically used for more artistic, preservation, or 3D printing purposes, is used for robotic NDE scanning and path planning. It holds an accuracy of 0.1mm, black boxing of certain functionalities of the device, which for


(a) Car 1


(b) Car 2

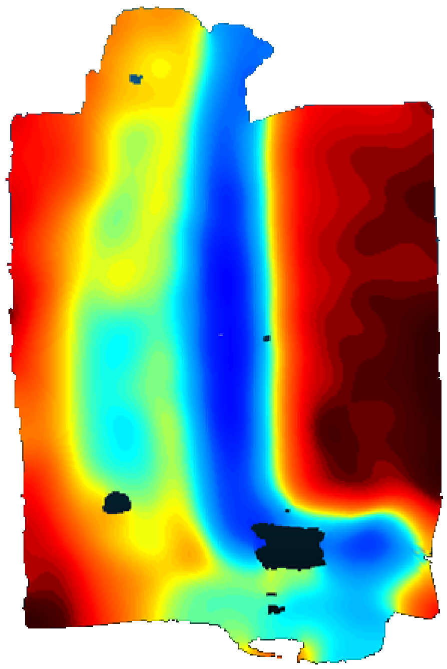
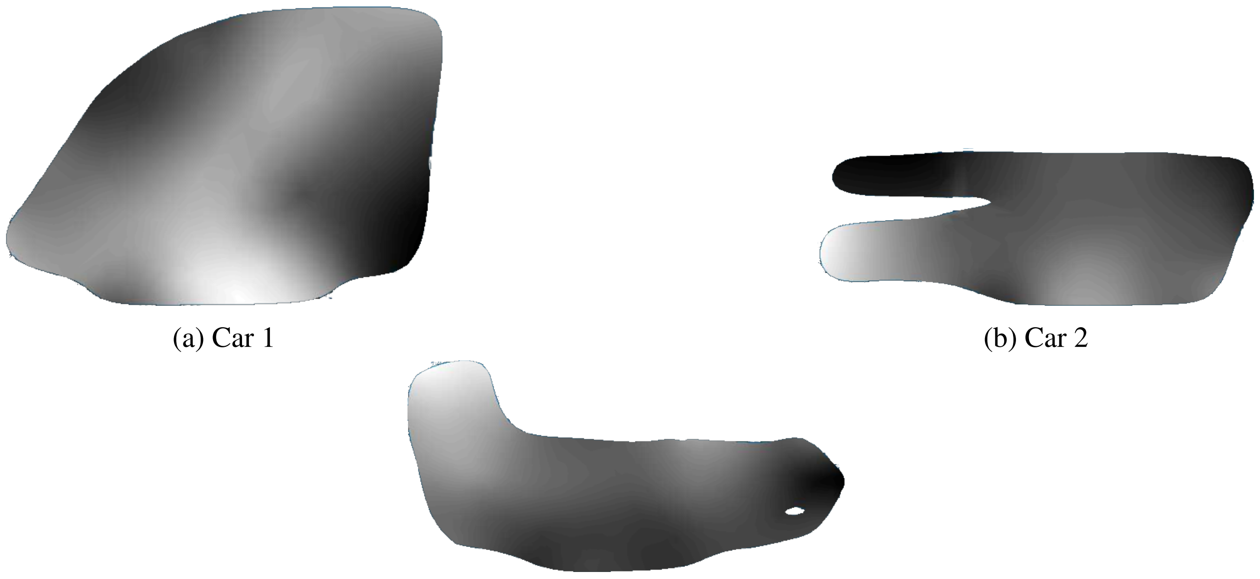

(c) Car 3

Figure 3.14 Stereo vision environment mesh results on CFRP car pieces, depth colormap.

commercial purposes is understandable, however when it comes to registration later, this becomes problematic unlike with the RealSense SDK which avoids most of this black boxing. Figures 3.16 and 3.17 show mesh outputs from the SL device on the CFRP car samples seen which is very suitable for NDE methods requiring high accuracy such as ECT, and works very well with UT. It also has advantages of a portability mode which allows for auto alignment intelligent matching algorithm

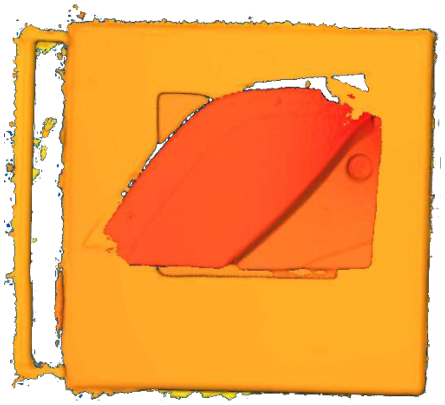

(a) Car 1

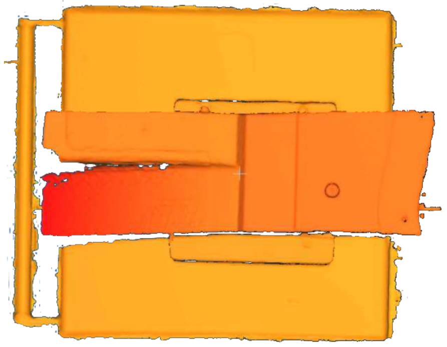
(b) Car 2

(c) Car 3

Figure 3.15 Processed stereo camera sample mesh results on CFRP car pieces, depth colormap.

for orienting multiple PTCs. This algorithm may also be used to obtain temporal information, overall improving output PTCs.


(a) Car 1


(b) Car 2

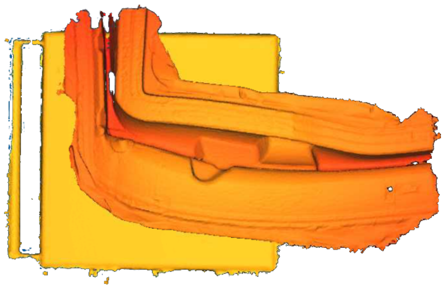
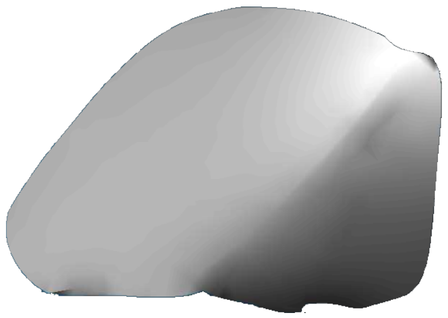
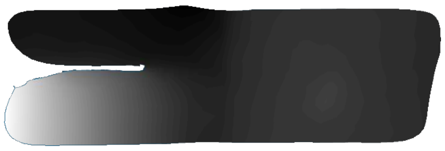

(c) Car 3

Figure 3.16 SL environment mesh results on CFRP car pieces, depth colormap.


(a) Car 1


(b) Car 2

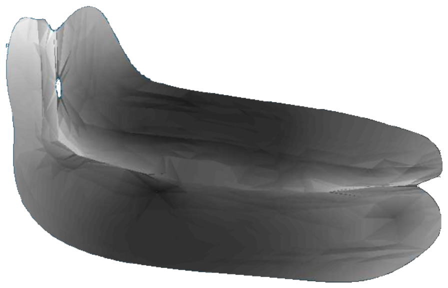

(c) Car 3

Figure 3.17 Processed SL sample mesh results on CFRP car pieces, depth colormap.

CHAPTER 4

ENVIRONMENT REGISTRATION

While environment reconstruction obtains the knowledge of the surface geometry of the sample for scanning and surrounding background, registration is how to align the geometry for it to be useful. If the geometry is placed back into a robot arm's base frame, then this enables scanning on the geometry. Therefore, the goal of registration is to obtain a single transform that references the relationship between the first orientation of the mesh to the robot. The complication is accurately obtaining this transformation automatically, particularly for snapshot images obtained from methods such as stereo cameras or SL. Just as accuracy is important for reconstruction, which decides the proper geometry for an NDE probe to move along, accuracy in registration is important in ensuring the NDE probe is moving along that geometry. As shown back in figure 1.4, having no alignment will not allow for NDE scanning as there is no alignment defined. The alignment must be estimated, which two methods will be discussed. Alignment may be placed onto a PTC P or mesh M, as the output of registration is a transform matrix holding translation and rotation and may be applied to either. The first is point pairs picking (PPP), in which at least three points are selected on the mesh that have known locations with respect to the robot. This method requires manual intervention, at least in its current usage, to measure the physical locations that as reference mark to the robot, and the locations on the digital environment for alignment. The second method is center point transformation (CPT), which uses the known orientation of the camera to the base and simply uses that as the transformation. This enables autonomous registration of the snapshot to the robot's base frame. Either the reconstruction device needs to be in a static location that is measured to the robot's base frame, a method that isn't covered but may be viable, or is placed on a measured location on the robot arm such as the end-effector. For example, if a mount is placed on the wrist of the arm, then measuring the displacement of the location on reconstruction camera where it obtains the snapshot location may be used as the final frame of the device. Forward kinematics can then be used to solve the location of the snapshot to the robot's base frame. The issue from this is that sometimes the snapshot itself does not orient with the measured physical location in which the snapshot is taken, which surprisingly was the main drawback from this method. CPT is technically the registration method used with 1D depth sensing raster scanning, which shows its effectiveness with high accuracy depth modules and its transformation known from the robot. Reconstruction methods such as iterative closest point [19], which automatically compares two PTCs or meshes to find the transformation frame between the two, is avoided as during testing though may find it's usage with pre-calibrated models used as reference. This was tried but did not lead to notable results due to major reliability issues in terms of accuracy. For NDE scanning, there are at least five conditions that should be met about registration. These both limit transformations of the NDE probe and reconstruction device and allow for focus on available setups within robotic environments.

1. The region to scan must be in the cyber-physical scene.
2. Error between the aligned frame and the robot's base should be minimized in accordance to the lift-off requirements of the NDE scan.
3. The physical region to scan and its associated cyber environment must be within the robot's workspace.
4. The physical region to scan and its associated cyber environment must not be obstructed by any background components, specifically preventing the NDE end-effector from reaching a waypoint or destination.
5. If using PPP, Calibration regions to be selected must be reconstructed along with the region to scan, and visible to a human operator within the cyber-physical scene.

Condition 1 is a simple statement in which if the cyber-physical scene is not available, then path planning will not have an input meaning no known waypoints can be generated. What is important to note from this condition is that it restricts the number of transforms inside of the workspace environment E. For example, if the camera or reconstruction device is placed on the robot, then it is bound by the restrictions of z_d. Condition 2 is purposefully abstract, implying that errors in alignment must be reduced to prevent NDE data loss or potential collisions. The purpose of this abstraction is due to the black boxing of surface scanning NDE methods, which differentiate lift-off requirements. On top of this, single channel "pencil" style probes are assumed, in which probes with arrays add complexity in terms of path planning and lift-off requirements for each element of the array. Conditions 3 and 4 define availability within the cyber-physical environment. Like condition 1, condition 3 is defined to ensure the point the generate is within E, or else they may not be reached and are considered as $null$ or not generated. Similarly for condition 4, if the NDE probe cannot reach the point, then it will be marked as $null$. Condition 5, only noted if PPP is used, then calibration regions need to be visible and measurable on the reconstructed mesh and physical location. If not, then PPP will be difficult to incorporate as no known or correct locations between the two frames would be known.

Increasing the area of the known scanning environment is beneficial for complex shapes which, require different perspectives to reconstruct regions unknown to a first snapshot, for large samples, or to purposefully reconstruct components of the background that may be useful, for example for calibration regions or collision detection. There is a phenomenon of "point cloud shadows", which are missing pieces of information behind a reconstructed mesh from a single snapshot. This may be avoided by merging multiple PTCs from different perspectives. Camera orientation O_{cam} is the basis of environment registration, as this factor mostly affects the required outcome of registration itself. As shown in figure 3.1, orientation of the snapshot device O_{cam} will decide projection of point's that define the PTC and mesh. By moving the camera, several different projections may be used. This is beneficial for expanding the region to scan, potentially for larger samples than the width and height of a first snapshot. The camera O_{cam} may be considered as floating near the robotic environment, enough so that the required range for z_d of the reconstruction device may be appropriately placed nearby the robot's workspace. Multiple orientations of snapshots may occur in two ways. One is to dynamically move one camera or reconstruction device; the other is to use multiple reconstruction devices shown in figure 4.1. Dynamically moving robots may be done by a human conductor holding a portable reconstruction device, then manipulating it around the robot.

This enables movement of the camera outside of the robot's workspace environment E, and robust movements down to the human operator's discernment. The drawback is that this method is not automated due to human intervention. Another method is to attach the reconstruction device on the robot, which is shown in figure 4.2. In this scenario, the camera orientation is equivalent to the forward kinematics solution defined by robot joints J and Denavit–Hartenberg parameters DH, shown in equation 2.6. Note there will be another frame to define the tool, which may be placed into DH_{n_j} and a static joint for θ_{n_j} at the final frame at index n_j. This operation is automatable, with figure 4.3 showing an ideal scenario where a hemispheroid scan is defined to obtain each piece of information of the sample within the environment. Dual robotics are shown so both sides of the sample may be defined, deploying the idea that multiple cameras may be used dynamically as well. After alignment of multiple snapshots obtained, which RealSense and Creality can merge multiple frames into a single local frame, the cyber environment area for path planning may be increased.

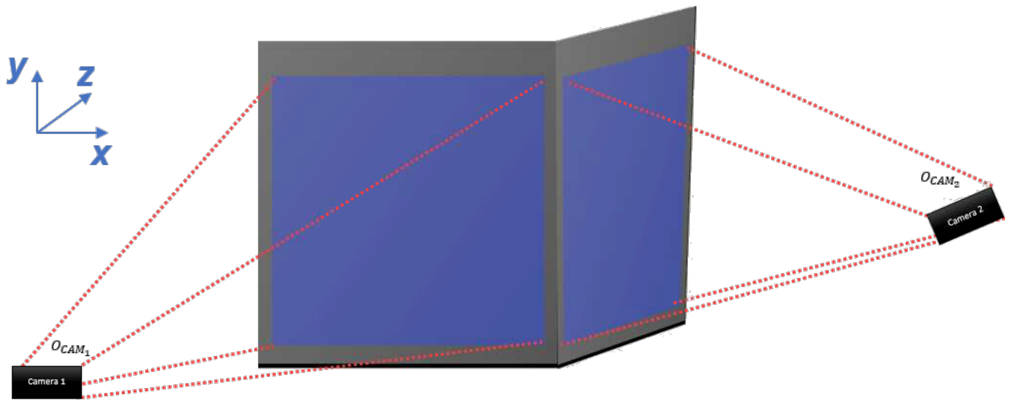


Figure 4.1 Multiple reconstruction device setup.

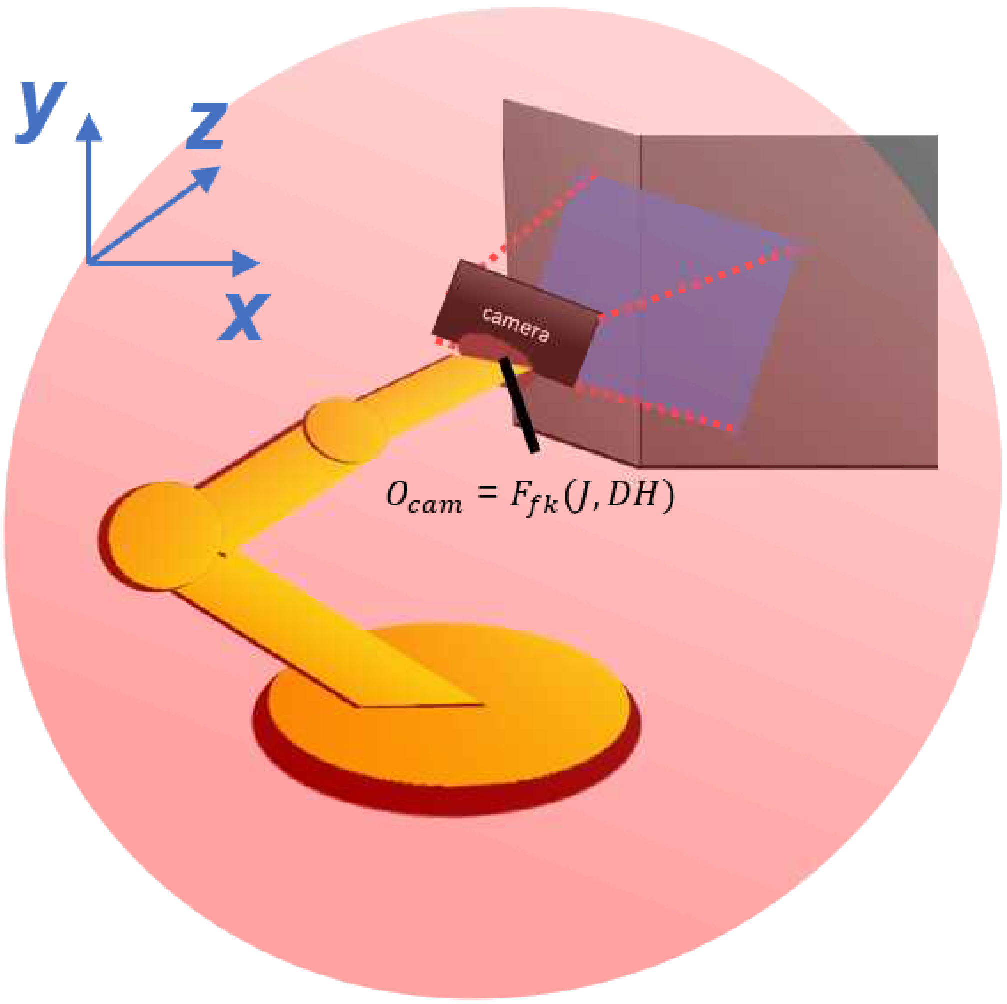


Figure 4.2 Camera as robot tool setup.

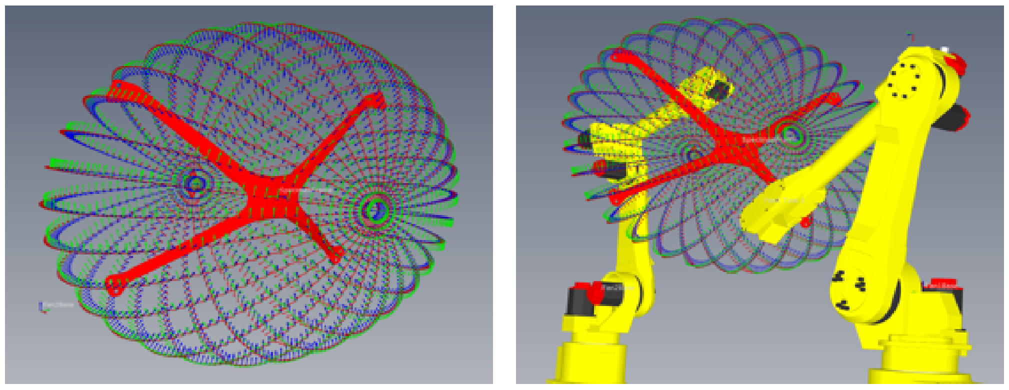


Figure 4.3 Hemispheroid scan using dual robots.

4.1 Point Pairs Picking

PPP, done within CloudCompare, takes two frames, one as reference frame O_{ref} used in PTC and mesh $P_{ref} \subset M_{ref}$ and one "to align" frame O_{al} used in $P_{al} \subset M_{al}$, and obtains the transformation between the two which is used to register P_{al} or M_{al} to P_{ref} or P_{ref} [37]. At least three point pairs must be defined between the two frames. The drawback of PPP is that it requires manual choice, at least in current usage, while receiving help from correct transformations. Manual choice is also prone to human error for selecting points on meshes, which becomes difficult to confirm. The reference frame should always be the frame that relates back to the robot's base frame. To effectively do this, then there must be regions on the reconstructed environment that hold "calibration" marks. These marks are measured locations that correlated with the initially unregistered mesh to align towards. There are two options for PPP. One is to use a reference PTC or mesh environment that is already registered to the robot's base frame and contains the calibration region, a method that is not examined due to the requirement of the environment already being registered limiting usage but may see some purpose with more development. The second is to physically measure the calibrations to the robot's base frame and use these points as approximate locations on the "to align" PTC or mesh. To avoid manually measuring these spots in person, which would be difficult to confirm, the robot arm itself may be used in which an end-effector is used to measure the location itself. This method, while taking some time to manually move the robot and has human subjectivity in choice, has shown effective usage alongside PPP. Calibration spots should remain in constant locations, perhaps in the background, to avoid remeasuring each location.

How PPP obtains the transformation is through approximation of the best transformation which lowers the root mean squared (RMS) error value between each point pair. Higher RMS error correlates to the potential error between the alignment of the cyber space versus physical space. RMS values should be treated carefully, however, as it only decides the error between the measured physical location and measured location on the aligned mesh. This means that if there is error on these measured points versus the actual points, then RMS will not be useful in finding this error.

For this, it's important to differentiate these types of PTCs, points, and orientations:

1. P_{al} or M_{al}: PTC or mesh to align
2. P_{ref} or M_{ref}: PTC or mesh in the reference frame
3. $P_{ph,A}$: Set of physical points as references, actual position A
4. $P_{ph,\alpha}$: Set of physical points as references, measured position α
5. $P_{cy,A}$: Set of cyber points to align, actual position A
6. $P_{cy,\alpha}$: Set of cyber points to align, measured position α
7. $p_{ph,A}$: Physical space point ph, actual position A
8. $p_{ph,\alpha}$: Physical space point ph, measured position α
9. $p_{cy,A}$: Cyber space point cy, actual position A
10. $p_{cy,\alpha}$: Cyber space point cy, measured position α
11. $O^{al}_{ref,A}$: Actual transformation or orientation (contains both translation and rotation) vector between $p_{ph,A}$ and $p_{cy,A}$
12. $O^{al}_{ref,\alpha}$ Measured orientation vector between $p_{ph,\alpha}$ and $p_{cy,\alpha}$, which is the final alignment result for O^{al}_{ref} between aligned O_{al} and reference O_{ref} frames

There are also several functions that should be described:

1. $p = Mtop(p_{4\times4})$, matrix form $p_{4\times4}$ to a translation vector as point p shown in equation 2.2
2. $O = MtoO(O_{4\times4})$, same as $Mtop$ but with context for an orientation O holding rotation parameters from matrix form $O_{4\times4}$
3. $p_{4\times4} = ptoM(p)$, obtain matrix form from vector form
4. $O_{4\times4} = OtoM(O)$, similar to $ptoM$ but for orientations

5. $p = Otop(O)$ or $p_{4\times4} = Otop(O_{4\times4})$, take translation parameters from orientation

6. $O = ptoO(p)$ or $O_{4\times4} = ptoO(p_{4\times4})$, create orientation from translation parameters

7. $p = Mtop(O_{4\times4}) = Otop(MtoO(O_{4\times4}))$, abstraction for $Mtop$ taking $O_{4\times4}$ and converting into p

8. $O^{n_j}_{0,4\times4} = F_{fk}(J, DH)$, forward kinematics equation from 2.6, inputting the joint set J and Denavit–Hartenberg parameters DH to obtain orientation matrix between the end effector frame at n_j back to the robot's base frame at 0

9. $O^{al}_{ref,4\times4} = F_{PPP}(P_{al}, P_{ref}, P_{cy}, P_{ph})$: point pair's picking algorithm, outputting an orientation matrix from the PTCs to align P_{al} and as reference P_{ref}. The point pair sets P_{ph} as reference and P_{cy} are also input. $F_{PPP,A}$ refers to the best possible result for $O^{al}_{ref,4\times4,A}$, while $F_{PPP,\alpha}$ refers the to the result which contains error in processing for $O^{al}_{ref,4\times4,\alpha}$

In general, matrix forms are better at explaining orientation due to having a single defined output, while vector forms condense that output and may generate different orientation matrix forms from $OtoM$. Function F_{PPP} will be abstracted as the function simply used within CloudCompare [37]. There are several statements for comparing physical and cyber alignments. To begin, the actual point positions between the two spaces are the same, ignoring errors due to misalignment or miss measurement:

$$p_{ph,A} = p_{cy,A} \tag{4.1}$$

This is an ideal scenario that when measuring should lead to the best approximation of $p_{ph,A}$ and $p_{cy,A}$. This is in no relation to the cyber environment PTC P_{al}, but just for points used within PPP for measurement's sake. The points between actual and measured spaces are also interchangeable for physical and cyber point sets:

$$p_{ph,A} \leftrightarrow p_{ph,\alpha} \tag{4.2}$$

$$p_{cy,A} \leftrightarrow p_{cy,\alpha} \tag{4.3}$$

Using a physical calibration device in frame O_{calD}, connected to the robot providing relation to reference frame O_{ref}, which directs a ray perfectly towards a calibration location $p_{ph,A}$. If the device is connected as end effector at frame n_j, then and orientation matrix $O^{n_j}_{calD,4\times4}$ may be defined between the end-effector frame and calibration device frame. Forward kinematics can be found to equate the end effector frame back to the robot, which used in conjunction with the known calibration to end-effector frame supplies the frame from the calibration device to the robot's base frame:

$$O^{calD}_{0,4\times4,A} = F_{fk}(J_A, DH) * O^{n_j}_{calD,4\times4} \tag{4.4}$$

This is still in the ideal case, since the joints J_A may have been measured manually for mastering, which may add to orientation error from forward kinematics. Using the frame from actual lift-off $z_{calD,A}$, describing the end of the ray oriented from generation spot of the calibration device:

$$p_{ph,A} = MtoP(O^{calD}_{0,4\times4,A} * O_{z_{calD,A},4\times4}) \tag{4.5}$$

The "dot" in which is oriented from the robot to the calibration point is ideally solved for $p_{ph,A}$. This means that a pointer, such as a laser, which has a visible output may be used even at a distance from the calibration device to the calibration point. Ideally, the robot should face the calibration device as end-effector tool in a manner that is normal alongside an axis. For example, if pointing downwards so normal vector $N_{-z} = [0, 0, -1]$ and likewise rotation $R^{calD,A}_0 = [180, 0, 0]°$:

$$O_{ph_{4\times4,A}} = \begin{bmatrix} 1 & 0 & 0 & t^{calD}_{0,x,A} \\ 0 & -1 & 0 & t^{calD}_{0,y,A} \\ 0 & 0 & -1 & t^{calD}_{0,z,A} + z_{calD,A} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With equivalence:

$$P_{ph,A} = Mtop(O_{ph_{4\times4,A}}) = [t^{calD}_{0,x,A}, t^{calD}_{0,y,A}, t^{calD}_{0,z,A} + z_{calD}]$$

Another examine is facing forward of the robot, with $N_x = [1, 0, 0]$ and likewise rotation $R_0^{calD,A} = [0, 90, 0]°$:

$$O_{ph_{4\times4,A}} = \begin{bmatrix} 0 & 0 & 1 & t^{calD}_{0,x,A} + z_{calD,A} \\ 0 & 1 & 0 & t^{calD}_{0,y,A} \\ -1 & 0 & 0 & t^{calD}_{0,z,A} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With equivalence:

$$p_{ph,A} = [t^{calD}_{0,x,A} + z_{calD,A}, t^{calD}_{0,y,A}, t^{calD}_{0,z,A}]$$

Therefore, if using a normalized rotation along parallel to an axis, then only the translation is needed for positioning the sensor. This does not account for errors, which are now proper to discuss. There are joint errors J_e related to the ideal joint set J_A to the measured joints J_α. If the physical sample is moved between the recorded snapshot and registration, there is a displacement $p_{D,e}$ between cyber and physical spaces. In current usage, human positioning will add error $p_{H,ph,e}$. Error in DH measurements are ignored. General position error $p_{pos,e}$ considers any unlisted cases of error. Compensating for these errors, the orientation matrix from the calibration device to the robot's base frame may be defined:

$$O^{calD}_{0,4\times4,A} = F_{fk}(J_\alpha + J_e, DH) * O^{n_j}_{calD,4\times4} \tag{4.6}$$

Point position errors may be added together:

$$p_{ph,e} = p_{pos,e} + p_{D,e} + p_{H,ph,e}$$

$$p_{ph,\alpha} = MtoP(O^{calD}_{0,4\times4,A} * O_{z_{calD,\alpha},4\times4}) + p_{ph,e} \tag{4.7}$$

By minimizing errors, ideal and measured results converge:

$$p_{ph,\alpha} \approx p_{ph,A} \tag{4.8}$$

Each point should be recorded into a point pairs vector that holds each equivalent point to size n_p:

$$P_{ph,A} = \sum_{n_i=0}^{n_p} p_{ph,A,n_i} \tag{4.9}$$

$$P_{ph,\alpha} = \sum_{n_i=0}^{n_p} p_{ph,\alpha,n_i} \tag{4.10}$$

These summations refer to appending points, not adding each point together. This was to solve the physical location of a calibration spot, done within the physical workspace, but for cyber equivalent locations are solved within a mesh simulator such as CloudComapre. In current implementation, a human selects the equivalent points $P_{cy,A}$ ideally and $P_{cy,\alpha}$ measured, based on the equivalent calibration location found on the reconstructed mesh. This is where PPP is used to solve the required orientation between the reconstructed environment frame O_{cy} and the robot's base frame at 0. An example of selection for $P_{cy,A}$ within CloudCompare is shown in figure 4.4. In the ideal case, using equation 4.5:

$$O^{cy}_{ph,4\times4,A} = F_{PPP_A}(P_{al}, P_{ref}, P_{cy_{al},A}, P_{ph,A}) \tag{4.11}$$

$P_{cy_{al},A}$ refers to the cyber points in the aligned frame, which are then transposed to the reference frame for $P_{cy,A}$:

$$p_{cy,A} = MtoP(O^{cy}_{ph,4\times4,A}) \tag{4.12}$$

The physical points should already be placed as reference to the robots base frame at 0, while the frame of the cyber points to align should refer to the output snapshot's frame, meaning that $O^{cy}_{ph,4\times4,A} = O^{cy}_{0,4\times4,A}$ is the solution to place the PTC or mesh to the robot's base frame at 0 for scanning. In this process, there are a couple of errors that may occur. First, the PPP function F_{PPP} holds loss, as it is trying to solve the best translation from points. This error is represented in RMS per cm but is defined as a matrix $O_{PPP_e,4\times4}$. Because of this, the function is differentiated as F_{PPP_α} which includes error. Errors for the physical calibration points within $P_{ph,\alpha}$ have already been defined. There is human error $P_{H,cy,e}$ when selecting points within cyber space, in current implementation. The output may now look like:

$$F_{PPP_\alpha}(P_{al}, P_{ref}, P_{cy_{al},\alpha}, P_{ph,\alpha}) = O^{cy}_{ph,4\times4,A} * O_{PPP_e,4\times4} \tag{4.13}$$

$$O^{cy}_{ph,4\times4,\alpha} = F_{PPP_\alpha}(P_{al}, P_{ref}, P_{cy_{al},\alpha}, P_{ph,\alpha}) * OtoM(P_{cy_{al},\alpha} + P_{H,cy,e})$$

$$P_{cy,\alpha} = MtoP(O^{cy}_{ph,4\times4,\alpha}) \tag{4.14}$$

By minimizing errors:

$$P_{cy,\alpha} \approx P, cy_A \tag{4.15}$$

$$O^{cy}_{ph,4\times4,\alpha} \approx O^{cy}_{ph,4\times4,A} \tag{4.16}$$

The solution for aligning the reconstructed environment to the robot's base frame is $O^{cy}_{ph,4\times4,\alpha} = O^{cy}_{0,4\times4,\alpha}$. An example is shown in figure 4.4 where several point pairs $n_p = 4$ with indices i_p are selected on calibration locations defined on "x" marks on tape placed on the table. These points were manually measured for $p_{ph,\alpha}$ using the end-effector tool, emulated in figure 4.5.

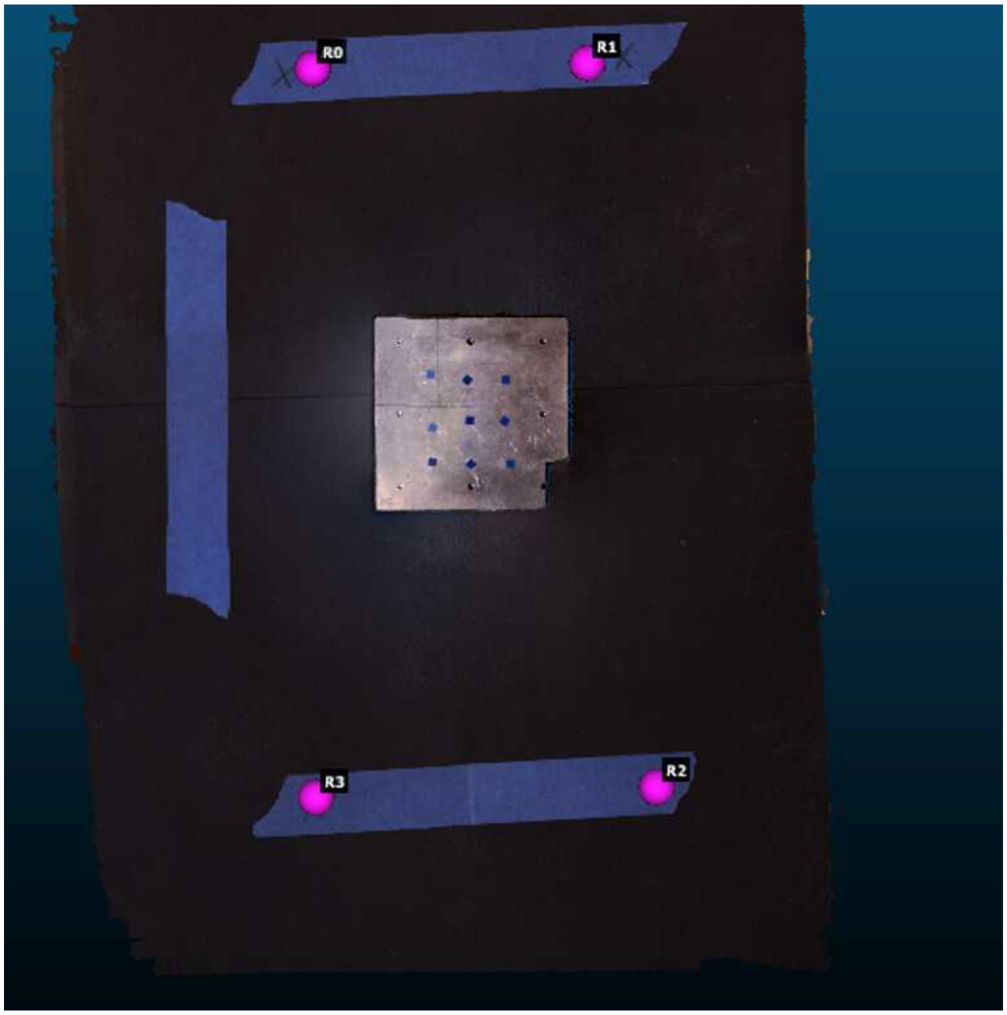


Figure 4.4 PPP with four calibration points on the background around an aluminum sample.

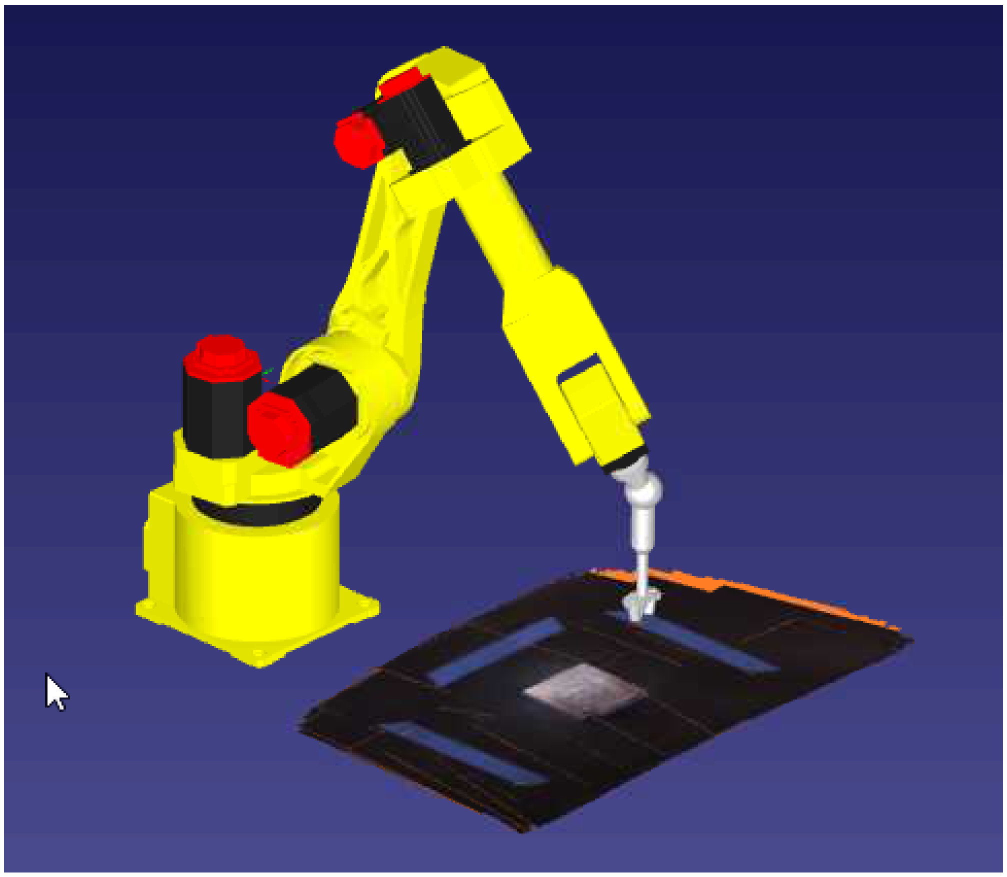


Figure 4.5 Robot simulation pointing the end-effector to the calibration spot.

i_p	$p_{cy_{al}}$	p_{ph}	RMS (cm)
0	0.806	72.826	0.0509
1	14.777	94.421	0.0471
2	−3.678	99.912	0.0234
3	−21.028	73.079	0.0273

Table 4.1 Comparison between cyber and physical points in the reference frame in figure 4.4.

$$O^{al}_{ref,4\times4} = \begin{bmatrix} 0.644 & 0.615 & 0.455 & 71.926 \\ 0.385 & -0.774 & 0.502 & -25.277 \\ 0.661 & -0.147 & -0.736 & -1.904 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$O^{cy}_{0} = O^{al}_{ref} = [T^{al}_{ref}, R^{al}_{ref}]$$

$$T^{al}_{ref} = [71.926, -25.277, -1.904](cm)$$

$$R^{al}_{ref} = [-168.7, -41.4, 30.9](^\circ)$$

4.2 Center Point Transformation

CPT is an automated method when the cyber snapshot's orientation is known about the robot's base frame. This is done from the reference frame of the reconstruction device physically to the orientation it outputs on the PTC or mesh environments. Unlike PPP which this transform is found via the PPP function for O_{cy}, CPT needs a definition of two frames: the frame of the reconstruction device O_{rd} and the frame of the cyber environment O_{cy}. The frame of the reconstruction O_{rd} should be in a measurable location with respect to the robot's base frame at O_0. To do this, the reconstruction device was placed as an end-effector tool on the robot arm, meaning O_{rd} is used as a frame between O_{n_j} and O_{cy}. In this scenario to solve O^{rd}_0 through the robot, forwards kinematics may be used using equation 2.6 with the final frame at O_rd. The ideal solution for the reconstructed environment linked to the robot's base frame O^{cy}_0 may be linked with O^{rd}_0 by finding $O^{cy}_{rd,A}$:

$$O^{cy}_{0,A} = F_{fk}(J_A, DH) * O^{rd}_{n_j,A} * O^{cy}_{rd,A} \tag{4.17}$$

$O^{rd}_{n_j,A}$ refers to the frame between the end-effector wrist joint to the "lens" of the reconstruction device where the snapshot is taken. Including joint error J_e from mastering as discussed in the PPP section:

$$O^{cy}_{0,\alpha} = F_{fk}(J_A + J_e, DH) * O^{rd}_{n_j,\alpha} * O^{cy}_{rd,\alpha} \tag{4.18}$$

The error included within $O^{rd}_{n_j,\alpha}$ may be due to fixture holding the reconstruction device or extracting the precise location of the "lens". To differentiate $O^{cy}_{rd,A}$ and $O^{cy}_{rd,\alpha}$, $O^{cy}_{rd,A}$ is the best frame that describes the transformation between the center point of the environment and the "lens". The center point of an environment is the zeroed point location, which may define as the identity or "eye" matrix:

$$eye_{4\times4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$eye = [T_{eye}, R_{eye}]$$

$$T_{eye} = [0, 0, 0]$$

$$R_{eye} = [0, 0, 0]$$

Rotations from transforms will rotate the environment at the zero location at T_{eye}. Therefore, the best transformation $O^{cy}_{rd,A}$ correlates the center point to the physical "lens". The process of CPT is shown in figure 4.6. This transformation may be unknown or even variable, depending on the application used, which the measured transformation is $O^{cy}_{rd,\alpha}$.

To solve for $O^{cy}_{rd,\alpha}$ so $O^{cy}_{rd,\alpha} \approx O^{cy}_{rd,A}$, this frame should be calibrated to solve the best possible transformation for the environment to physical space. To solve this frame, perhaps a method such as PPP may be used, which can closely approximate transformations, and through backwards-propagation, or reusing the same technique until the closest approximation is available, may a solution for CPT be available. Assume a calibrated final transformation of the mesh $O^{cy}_{n_j,4\times4,cal}$, which holds the transform from the environment back to the robot's base frame. From the PPP section, $O^{cy}_{n_j,4\times4,cal}$ is decided as $O^{cy}_{ph,4\times4}$. Using the inverse function inv of matrices, it is possible to solve O^{cy}_{rd} using forward kinematics and $O^{cy}_{n_j,4\times4,cal}$ [182]. The following solutions are then possible, for the ideal case:

$$O^{cy}_{rd,A} = inv(F_{fk}(J_A, DH)) * O^{cy}_{n_j,A,4\times4,cal} \quad (4.19)$$

and for the measured case:

$$O^{cy}_{rd,\alpha} = inv(F_{fk}(J_A + J_e, DH)) * O^{cy}_{n_j,\alpha,4\times4,cal} \quad (4.20)$$

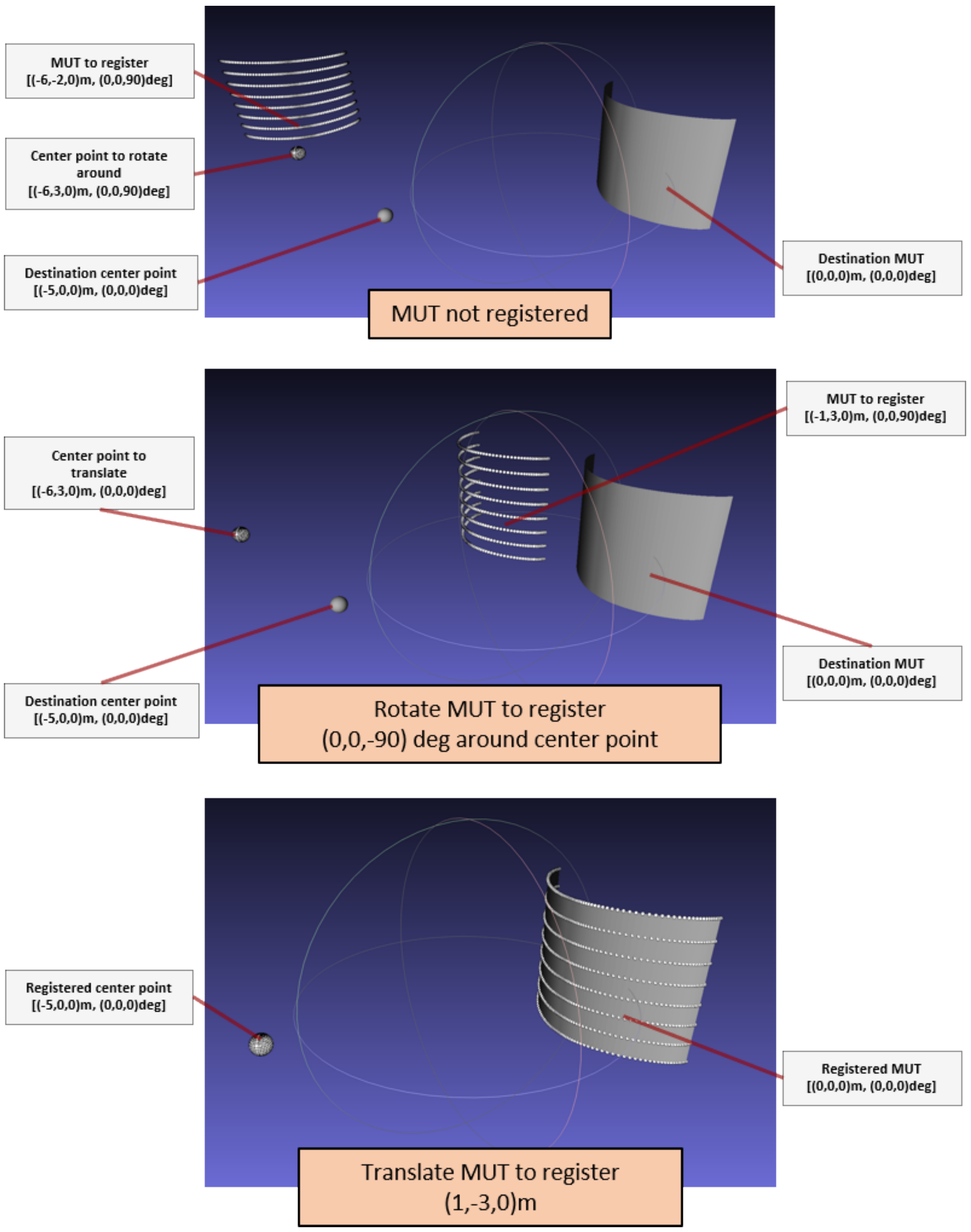


Figure 4.6 Center point transformation example, showing the process of rotating then translating an aligned cyber-physical scene to the reference frame.

Placing either into equations 4.17 or 4.18 supplies the solution for CPT. The main issue met with CPT, despite it's high advantage of autonomous registration, is that the error involved with $O^{cy}_{n_j,\alpha,4\times4,cal}$ is quite significant in practice. When testing with the Creality device, it turns out that the center point shifts around dynamically between frames, making it difficult to impossible to use CPT dynamically. This limited its usage within this thesis, when the slower but more reliable

PPP algorithm tends to provide more correct registrations. It shouldn’t be discarded as a method, however, as with better approximations, it may be a stronger method overall.

CHAPTER 5

TOOLPATH GENERATION

Toolpath generation is the process of generating waypoints within digital space that will be placed into inverse kinematics equation 2.7 for the robot simulator and physical robots to move towards. Much discussion has already been addressed on how to obtain waypoints through ray-triangle intersection within chapter 2. This chapter will discuss the generation of ray-triangle grids which will be used for generating optimal waypaths. "Optimal" in this context may be different parameters, depending on usage. It may be distance traveled, time taken to conduct a scan, or energy consumed during the scanning process. These parameters may be defined intermittent between waypoints as "weights". On top of the concept of weights, optimal paths might also be the computation time to generate the waypath as well. For example, if the weight selection is to use time to conduct the scan, with the implication that the time to start the scanning procedure includes computation of the waypath, then computation time may be significant. Several methods for solving paths in terms of low computation time and sub-optimal path planning versus higher computation time and optimized path planning are discussed. For any waypoint p_w inside the list of waypoints W for any index i_w:

$$W = [w_0, w_1, w_2, ...w_{n_w}] \tag{5.1}$$

5.1 Basic Generation Algorithms

This section discusses basic generation algorithms that may be extended to the complex path planner. To start, a 2D configuration is discussed, in which later implementation will be used as a basis for the ray-triangle grid arrays. For any waypoint $p_{w,2D}$ in set W_{2D} within the 2D configuration:

$$p_{w,2D} = [T_{2D}, R_{2D}]$$

$$T_{2D} = [t_x, t_y, 0]$$

From equation 2.11, the normality $N_{2D,inv}$, which in this instance is pointing initially upwards in the z-axis, may be used to solve for the rotation $R_{2D,inv}$:

$$N_{2D,inv} = [0, 0, -1]$$

$$R_{2D,inv} = [0, 0, 0]$$

The rotation is "inverted", meaning instead of it pointing upwards, it should be facing downwards to the ground which is how the NDE probe will be oriented. Simply, a rotation of 180° applied to $R_{2D,inv}$ along the x-axis will reorient each waypoint:

$$R_{2D} = R_{2D,inv} * R_x(180) = [180, 0, 0]$$

See above equation 2.3 for the matrix of $R_x(\alpha)$. This rotation produces normal $N_{2D} = [0, 0, 1]$ which faces the ground, along the z-axis. The scanning environment, defined as the area in which waypoints are within, is defined as E_{scan} with:

$$E_{scan,min} = [0, 0, -\infty]$$

$$E_{scan,max} = [w, h, \infty]$$

As the z-axis is not defined, the boundaries go to infinity, though these are restricted by the workspace environment E. With w being the width of the scan along the x-axis, and h being the height of the scan along the z-axis. The potential points for E_{scan} are defined to be within the robot workspace E, so $E_{scan} \subset E$. Any points generated outside the bounds for E_{scan} should be ignored, which specifically points that are not generated for movement are counted as $null$. The scan should have an area defined by lengths L_x and L_y which extend from starting point $p_{w,2D,0}$. Considering boundaries:

$$E_{scan,min,x} < p_{w,2D,0,x} < p_{w,2D,0,x} + L_x < E_{scan,max,x}$$

$$E_{scan,min,y} < p_{w,2D,0,y} < p_{w,2D,0,y} + L_y < E_{scan,max,x}$$

The boundaries of along the z-axis are not needed in the current definition, as they are not defined within these any waypoint in the set. Resolutions rs_x and rs_y may be defined, which decides the number of points placed alongside the x-axis and y-axis respectively. Higher resolutions of rs_x increase the number of raster lines, which is significant for increasing the resolution and scan time of the output PTC, while rs_y increases the density of points on a raster, important for moving more smoothly along the input mesh surface. It should be noted that resolution of NDE data points on a raster path is figured out by the time taken to move along the line versus the data acquisition speed, meaning outside speed decreases due to an increase rs_y or only acquiring data at a point, rs_y doesn't increase the resolution along the y-axis. From resolutions, step-sizes s_x and s_y may be defined:

$$s_x = \frac{l_x}{rs_x - 1}$$
$$s_y = \frac{l_y}{rs_y - 1}$$

Inversely, rs_x and rs_y may be defined as $rs_x = l_x s_x$ and $rs_y = l_y s_y$. Figure 5.1 holds labels to describe the height and width of the sample (red), the boundary lengths (blue) and point step-sizes (magenta) the along the x and y axes, the waypoints (yellow) including the start point (cyan), and all possible paths (green). In this example, $rs_x = 4$ and $rs_y = 3$. It should be noted that this is a "point-by-point" waypath system, which entirely relies on waypoints to move towards. Ideally, increasing rs_x increases the resolution of the image while rs_y increases the resolution with respect to the curvatures of the surface. In a flat system, it is sufficient to leave $rs_y = 2$, which defines only the top and bottom of the length l_y. If using $rs_y = 2$, then $s_y = \frac{l_y}{2-1} = l_y$. The travel between points is assumed to be linear, following a ray between the two points. When dealing with a robotic arm system, this linear movement is decided by the joint sets needed per point via inverse kinematics. More advanced systems may incorporate "splines", which are curved lines that may move along the contours of the surface, removing the need for discretization of points. Specifically, b-splines have been examined in the past [184], however these methods had trouble in implementation due to reconstructed meshes having many faces that complicate movement of b-splines.

The current waypath is unorganized, meaning that no direct pathing between points has been defined. From the starting point w_0 to the last point w_{n_w} should be the order of points to move towards. It is helpful to generate the waypath to at once have a practical path on generation, rather

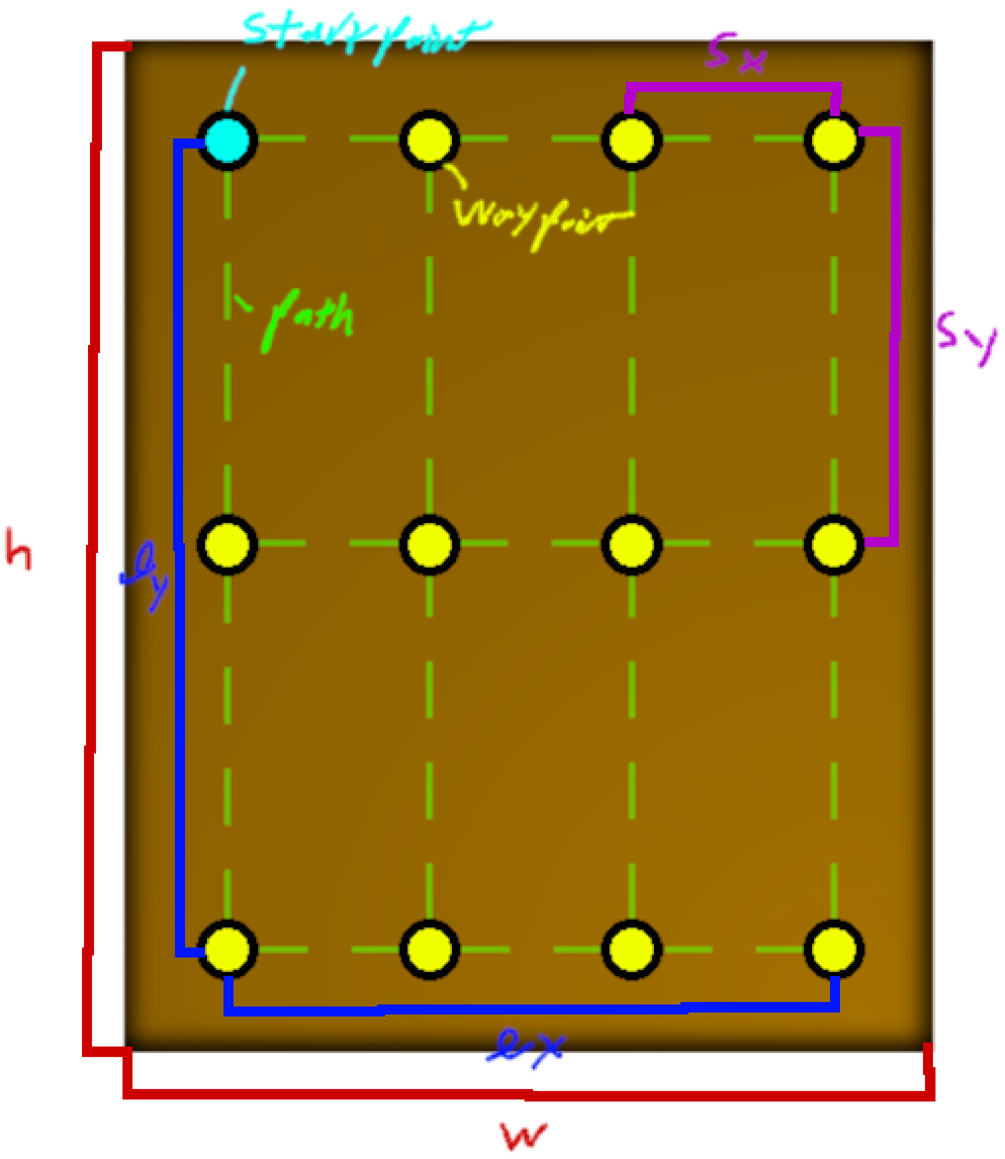


Figure 5.1 Way point example within 2D coordinates.

than each point being at random locations. One of the simplest algorithms is the "line-by-line" point generation algorithm, in which all points are generated via a double for-loop, creating a 2D grid of points along the x and y axes. This is shown in figure 5.2 and algorithm 5.1. The NDE probe will scan across a linear path, acquiring information along this path, then swing back to the next point in a diagonal manner. The total distance δ_{ll} may be defined, which may be useful in deciding total weight, such as if distance is the property used, or time which may be simplified to $t = \delta/v$ with v being static velocity value. This allows weight efficiency comparisons between distance and time to be analogous. Note that scanning time depends on more complex parameters than simple velocity from the robot joint system. This is relevant for point-by-point scanning, as the robot may stop shortly at each point, requiring deceleration and acceleration at each point. For the sake of optimization comparison, $t = \delta/v$ will be used. Total distance δ_{ll} is decided by the equation, adding the lines moved along length l_y and diagonals $\sqrt{s_x^2 + l_y^2}$:

$$\delta_{ll} = (rs_x * l_y) + [(rs_x - 1) * \sqrt{s_x^2 + l_y^2}]$$

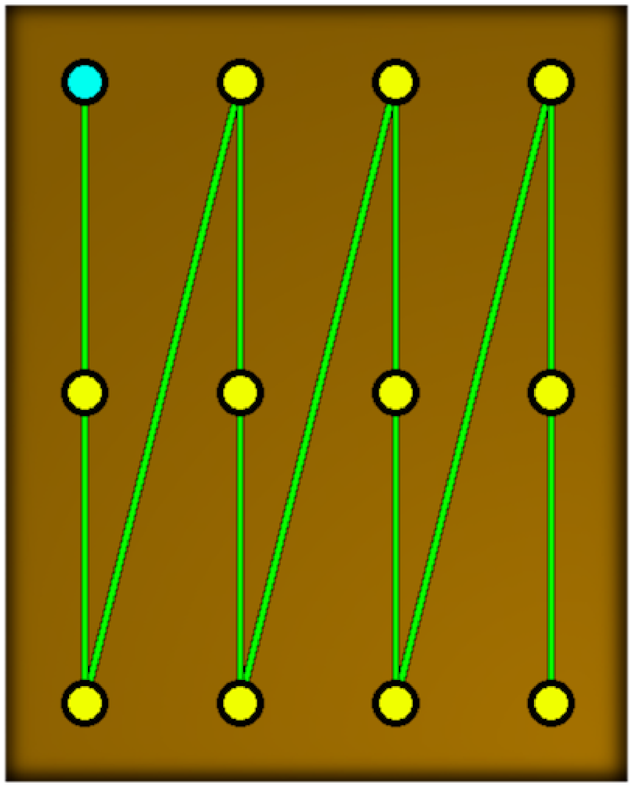


Figure 5.2 Line-by-line waypoint example.

It should be noted that in the 2D scenario that resolution rs_y has no effect in the total distance traveled. For 3D points that follow complex surfaces, rs_y plays a role as points may move upwards or downwards along the z-axis, increasing distance per point along l_y. The diagonal movement is not very efficient, as that is a large movement to move to the next index along the x-axis. It also tends to add distortion to final images from the diagonal movement interfering with the desired raster. Instead, it is better to move to the closest neighbor with only a slightly more complicated algorithm: zig-zag point generation. This algorithm is used instead of line-by-line due to its overall

Algorithm 5.1 Line-by-line point generation.

```
Input: Two real numbers $l_x$ and $l_y$, two non-negative integers $rs_x$ and $rs_y$
Output: List of points, $list$
  $s_x = l_x \,/\, rs_x$;                                          ▷ Define sizes
  $s_y = l_y \,/\, rs_y$;
  $list = new\ PointsList(s_x * s_y)$;       ▷ Define as $PointsList$ with capacity $s_x * s_y$
  for $x = 0; x < rs_x; x = x + 1$ do                ▷ Iterate through all lines along $l_x$
      $pos_x = s_x * x$;                                 ▷ Define the position on x
      for $y = 0; y < rs_y; y = y + 1$ do      ▷ Iterate through point on the line along $l_y$
          $pos_y = s_y * y$;                             ▷ Define the position on x
          $point = new\ Point(tx = pos_x, ty = pos_y, tz = 0, rx = 0, ry = 0, rz = 0)$;
          $list.Add(point)$;                   ▷ Create and add a point to the list
      end for
  end for
  $return\ list$;
```

better performance as a default point generation tool. The zig-zag point generation algorithm is shown in figure 5.3 and with supplied pseudo-code in algorithm 5.2. The difference between zig-zag and line-by-line generation algorithms is that a Boolean "zig" parameter is defined. If "zig" is true, then the probe running along the y-axis will move in a positive manner. If "zig" is false, this means to move along the "zag" path, or along the y-axis in a negative manner. What's important is organizing the points so shifted point on x-axis is neighbored to the last point of the y-axis, on either "zig" or "zag". Consecutive points along the zig or zag paths need to be organized to move in each direction. In terms of total distance δ_{zz} in the 2D configuration, the pieces that create the path are now a summation of zigs and zags traveled at l_y, and the summation of shifts as just the length l_x:

$$\delta_{zz} = (rs_x * l_y) + l_x$$

Compared to the δ_{ll} for line-by-line, scan, the difference between the two is the requirement to shift along the x-axis, with $\sqrt{s_x^2 + l_y^2}$ for line-by-line and s_x for zig-zag. Inequalities may be defined between the two:

$$s_x \leq \sqrt{s_x^2 + l_y^2}$$

This leads to:

$$\delta_{zz} \leq \delta_{ll}$$

It should be noted if $l_y = 0$, then $\delta_{zz} = \delta_{ll}$. If $l_y = 0$, then only a single straight line along l_x will be conducted, or in other words, a line-scan is conducted, which eliminates movement along y. Because such inequalities and similar computation requirements, zig-zag path generation is used over line-by-line. For zig-zag scans, there is also potential to round off the edges at the end of each raster iteration, leaving the edges to be less "square like". This is suggested but not implemented. One downside is that the edges while shifting may record information, which adds distortion on the edges of the retrieved scan. Fortunately, this is on the edges rather than through the image like line-by-line will conduct.

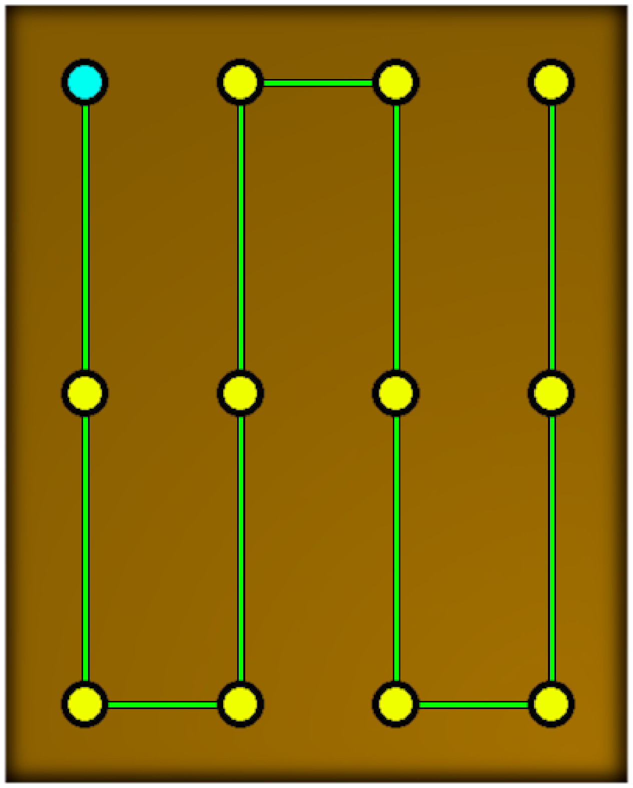


Figure 5.3 Zig-zag waypoint example.

3D surface configurations are well different, as movement along the z-axis is enabled. An example is shown in figure 5.4 and its mesh representation shown in figure 5.5. With the extra dimension, there is more to consider. For samples of higher complexity, higher resolutions for rs_y may be useful to follow the contour movements of the sample. This may be interpreted in an analogous way to Nyquist's sampling theorem, with the required geometry to move towards may

Algorithm 5.2 Zig-zag point generation.

Input: Two real numbers l_x and l_y, two non-negative integers rs_x and rs_y
Output: List of points, $PointsList$

```
1: $zig = true$;                                         ▷ Define the zig-zag variable
2:                          ▷ Note: zig => zag = true, and zag => zig = false
3:
4: $s_x = l_x/rs_x$;                                     ▷ Define sizes
5: $s_y = l_y/rs_y$;
6: $list = newPointsList(s_x * s_y)$;   ▷ Define as $PointsList$ with capacity $s_x * s_y$
7: for $x = 0; x < rs_x; x = x + 1$ do           ▷ Iterate through all lines along $l_x$
8:     $pos_x = s_x * x$;                                ▷ Define the position on x
9:     for $y = zig ? (0 : rs_y - 1); zig ? (y < rs_y : y >= 0); y = y + zig ? (1 : -1)$ do
10:                                  ▷ Iterate through point on the line along $l_y$
11:        $pos_y = s_y * y$;                            ▷ Define the position on x
12:        $point = new\ Point(tx = pos_x, ty = pos_y, tz = 0, rx = 0, ry = 0, rz = 0)$;
13:        $list.Add(point)$;                  ▷ Create and add a point to the list
14:    end for
15:    zig != zig;                             ▷ Flip the zig->zag or zag->zig
16: end for
17: $return\ list$;                                     ▷ Return the resultant list
```

be interpreted as a 1D signal defined by the mesh, with contours defining a required "frequency" f_M. As the resolution may also be interpreted as a frequency of points along this 1D path, then for effective usage [100]:

$$rs_y > 2 * f_M \tag{5.2}$$

This may be an oversimplification, as the choice for f_M should exist for every raster conducted throughout the scan, however it is a suggestion to follow to ensure scans properly follow the contour of the mesh. On top of this, the number of points as resolution should be minimized to prevent high computation time, either using more complex organization algorithms or during inverse kinematics computation plus collision detection per point in simulation. If spline methods are used to manipulate along the curves directly from the surface of the mesh, this may prove useful for raster scanning.

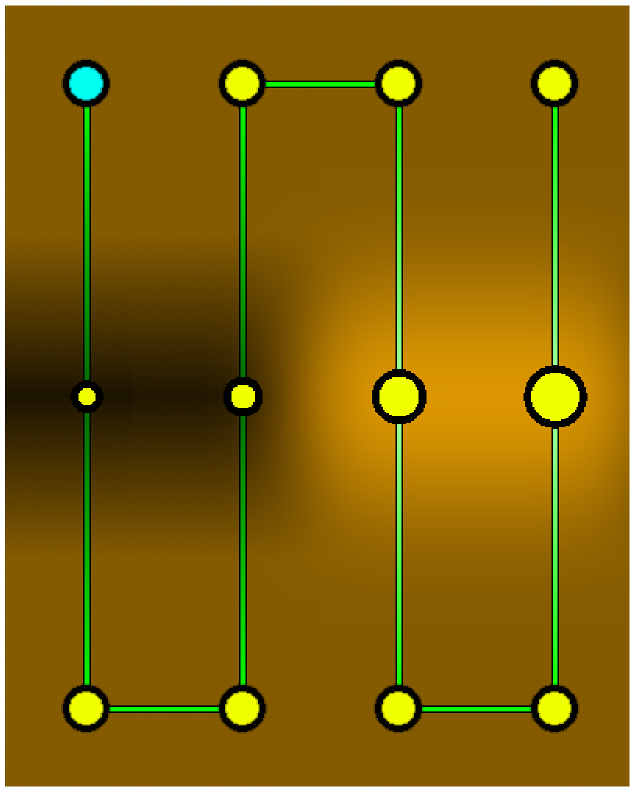


Figure 5.4 Zig-zag waypoint example within 3D coordinates.

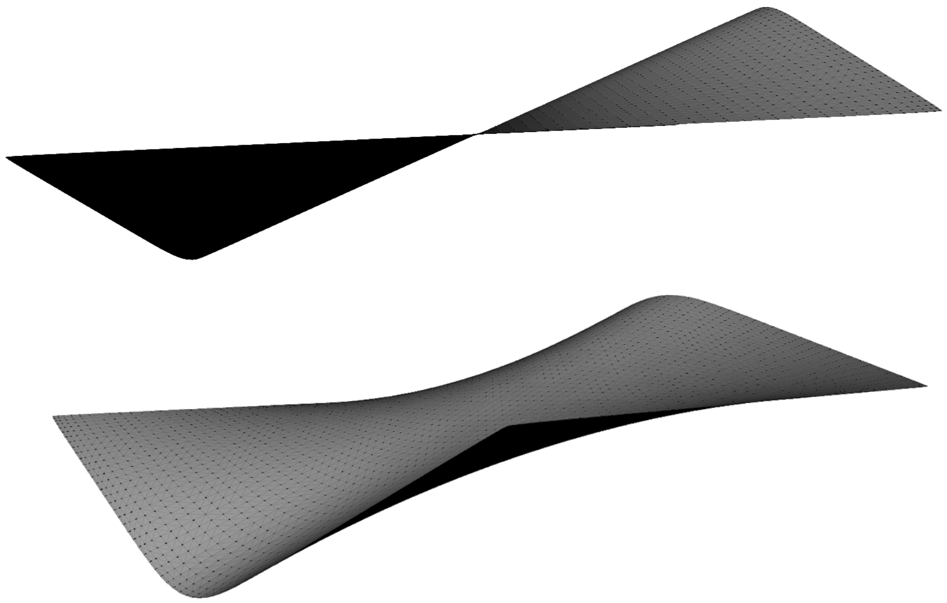


Figure 5.5 Mesh representation of figure 5.4.

5.2 Ray-Triangle Intersection Arrays

The custom path planner uses the concepts just discussed to parse a grid of waypoints to generate a waypath placed into the robotic simulation software. Ray-triangle intersection arrays that are organized in a zig-zag organized waypaths are used, which generates a path with 3D coordinates. The concepts of generating a point on a point via ray-triangle intersection was discussed in equation 2.12 for translation on the mesh, equation 2.11 for retrieving rotation from a face's normal, and equation 2.13 including for lift-off compensation. Instead of discussion theory and waypath implementation like past section, this section focuses on the customized path planner. The engine that runs the path planner was done with an open-source toolkit based on OpenTK which contains low-level controls for OpenGL bindings and works alongside the developed C-sharp code that runs the rest of the NDE acquisition software. For reference of this open-source engine, see [88]. From having low level control, it was possible to enable ray-triangle intersection to obtain any way point w_p on intersection and ignore potential mistaken waypoints from non-intersecting rays. The consequence of using such a system is that much implementation must be customized by hand, which is burdensome on development. For example, a spline or detection system based on line edges for waypath may be a potential option but requires development and debugger time to carry out. There are several optimizations within this engine for collision detection with respect to ray-triangle intersection, but these will not be covered.

To start, figure 5.6 displays a single ray against a flat plane. The intersection point w_p is on the surface of the mesh and holds a normal which extends from the z-axis away from the triangular faces on the place. The ray has a finite length, with a defined start point $p_{r,start}$ and stop point $p_{r,stop}$. $p_{r,start,0}$, not accounting for rotation, with respect to position x pos_x and position y pos_y from algorithms 5.1 and 5.2, may be defined as:

$$p_{r,start,0} = [pos_x, pos_y, 0]$$

Likewise, $p_{r,stop,0}$ without rotation may be defined with ray length l_r:

$$p_{r,start,0} = [pos_x, pos_y, -l_r]$$

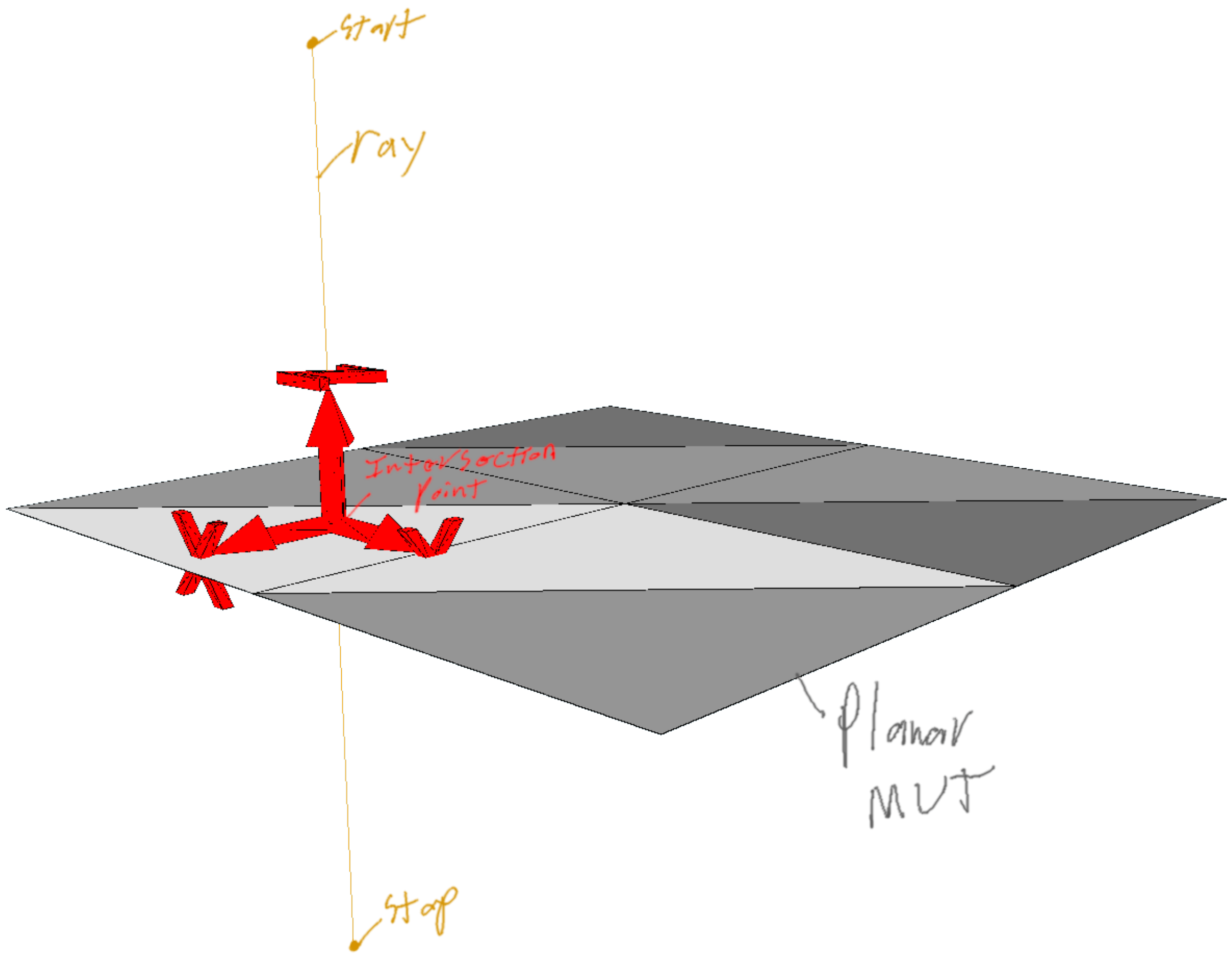


Figure 5.6 Ray-triangle intersection against a flat planar sample.

$p_{r,start}$ and $p_{r,stop}$ should allow for rotation along an axis to enable intersection along any mesh orientation. Using a rotational matrix $R_{r,4\times4}$, and functions $p_{4\times4} = PtoM(p)$, converting a vector into a 4x4 matrix, and vice-versa $p = MtoP(p_{4\times4})$:

$$p_{r,start} = MtoP(PtoM(p_{r,start,0}) * R_{r,4\times4})$$

$$p_{r,stop} = MtoP(PtoM(p_{r,stop,0}) * R_{r,4\times4})$$

The algorithms input the start position $p_{r,start}$ and direction normal vector d_r, instead of stop point $p_{r,stop}$. d_r may simply be solved by normalizing the difference between the two ray points:

$$d_r = \frac{p_{r,stop} - p_{r,start}}{|p_{r,stop} - p_{r,start}|}$$

About ray length l_r, it may be a useful tool to limit the length of l_r as a boundary condition of where to scan within the environment, to ensure that points are not generated far from the sample to scan. l_r may also theoretically be infinite, however in practice in a digital environment, a value of infinity is not possible as it can't be defined in the environment. If an "infinite" is needed, then the max value of the value's type may be used. Lift-off is important to parameterize, to support low and constant lift-offs. As mentioned previously, there are two approaches: a global lift-off which a single translation is allied to the complete set, or a local lift-off based on a constant distance away from the normal of the intersected face. These methods were discussed with resulting equations 2.2.6 and 2.13.

When a waypath is generated, then the rays will generate a "rectangular prism" in which faces within the prism may intersect with the arrays to generate waypoints. The start and stop of each array generate a plane from the different generated pos_x and pos_y values of the input waypath, holding width w and height h. These planes are considered as the "approach plane" and "end planes" respectively. The direction which this prism points towards is considered as the "approach angle" which all rays follow, defined by a single rotation $R_r = MtoPR_{r,4\times 4}$. Figures 5.7 and 5.8 display a simple example on a unit plane with width and height $w = h = 1$, resolutions $rs_x = rs_y = 5$, and lengths $l_x = l_y = 0.5$. Each generated point is normal to the surface, and the path generates a zig-zag pattern. When applying approach angles that are against the surface, there is a bit of distortion of the path placed onto the mesh, which may or may not be of use. Figures 5.9 and 5.10 show generate as similar path to figures 5.7 and 5.8, but at a 45° angle. The path stretches on the plane.

Another configuration shown in figures 5.11, 5.12, and 5.13 intersects a soccer ball-like mesh, with resolutions $rs_x = rs_y = 9$ and lengths $l_x = l_y = 2$. This shows the capability of adjusting the rotation per point on surfaces with different normals per face. Figure 5.13 shows the path (orange) alone without the waypoints and approach and end planes.

5.2.1 Optimal Waypath Organization Algorithms

Waypaths to this point have been generated from an unorganized algorithm which creates a raster pattern. This section focuses on some options to perfect the total weight, decided by total distance, by reorganizing the waypoints within the waypath. Calculations about ray-triangle

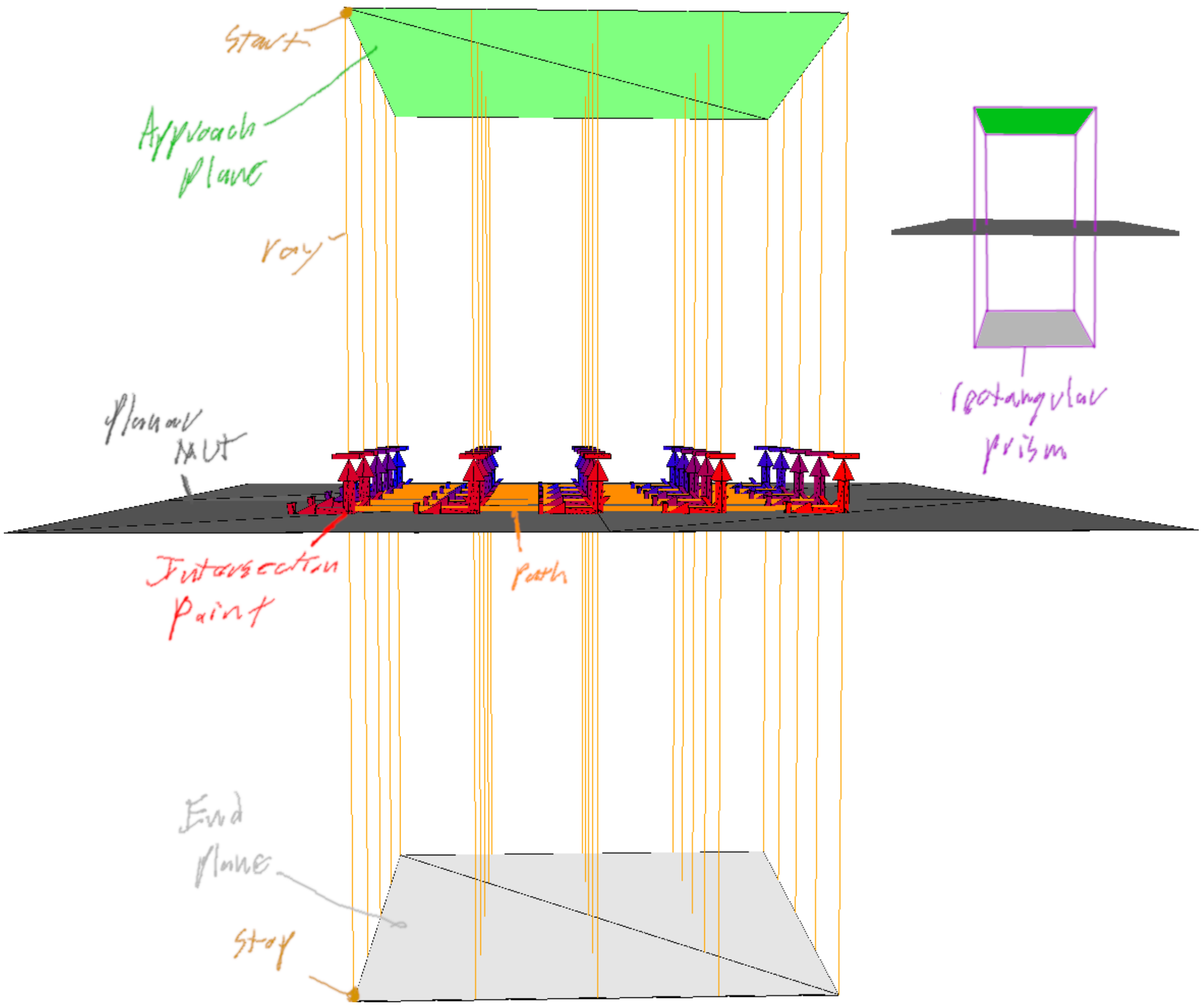


Figure 5.7 Plane-triangle example against a unit plane showing the approach planes.

intersection have already been calculated, which saves time in computation. Collision detection along the environment is decided afterwards, since it is a computationally expensive function that may be performed after solving the waypath. One quick disadvantage to note is that when applying organizational algorithms, the data may distort based on the path taken from the organized path, which raster scanning prevents. As discussed before, the path of distance δ is analogous to time, considering if time calculation is abstracted from a constant velocity v so $t = \delta/v$. The solution for total weights is considered by the famous traveling salesmen problem (TSP) [90] [9]. When dealing with TSPs, it is important to juggle two parameters related to time: computation time t_c and travel time t_t. This version of the TSP requires all points to be found, excluding any points that

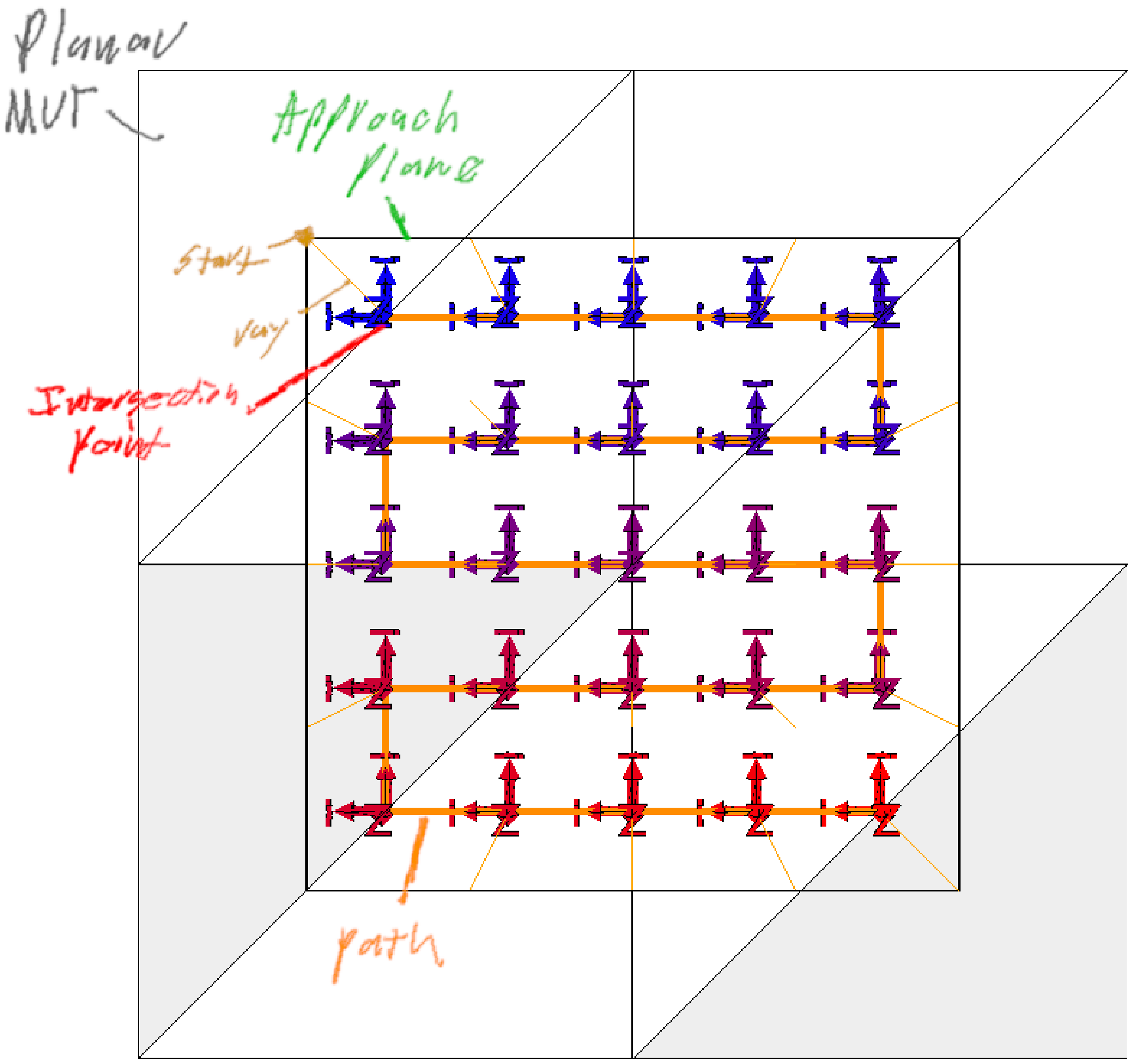


Figure 5.8 Plane-triangle example against a unit plane with a bird's eye view.

are *null*. This is known as a Hamiltonian path or circuit [66], which have been historically difficult to solve for best solutions. As it turns out, solving the best weight solution supplies exceptionally long computation times that are impractical for scanning. Such methods such as brute force supply exact locations by calculating all possible waypaths or "tours" and outputs the shortest tour of the bunch [152]. The problem is that calculating every solution is burdensome on performance, even if simple calculations ignoring collision detection are used. Now it is proper to introduce the "Big O Notation", which is used to evaluate performance of algorithms. The big O notation is used to

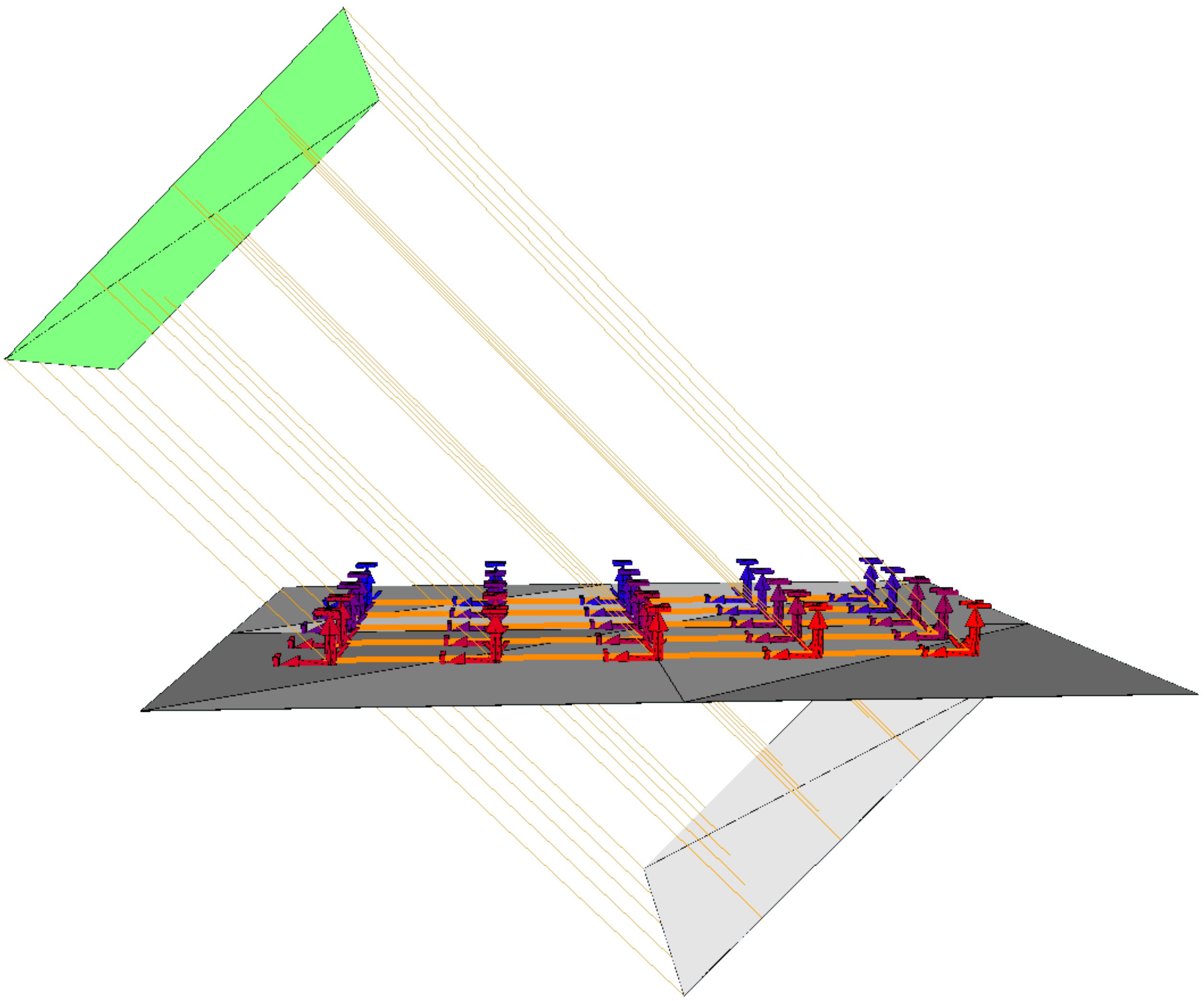


Figure 5.9 Plane-triangle example against a unit plane showing the approach planes when the approach is tilted.

classify the performance of algorithms based on the size of the input data placed into the system [36]. In this case, the input size n is uses the area of input resolutions into the default path planner algorithm, rs_x and rs_y:

$$n = rs_x * rs_y$$

Big O notation evaluates the number of computations done on the input set of data based on the algorithm used. Brute force does the following procedure: [152]:

1. Find the total number of tours

2. Connect and list all tours possible from input vertices

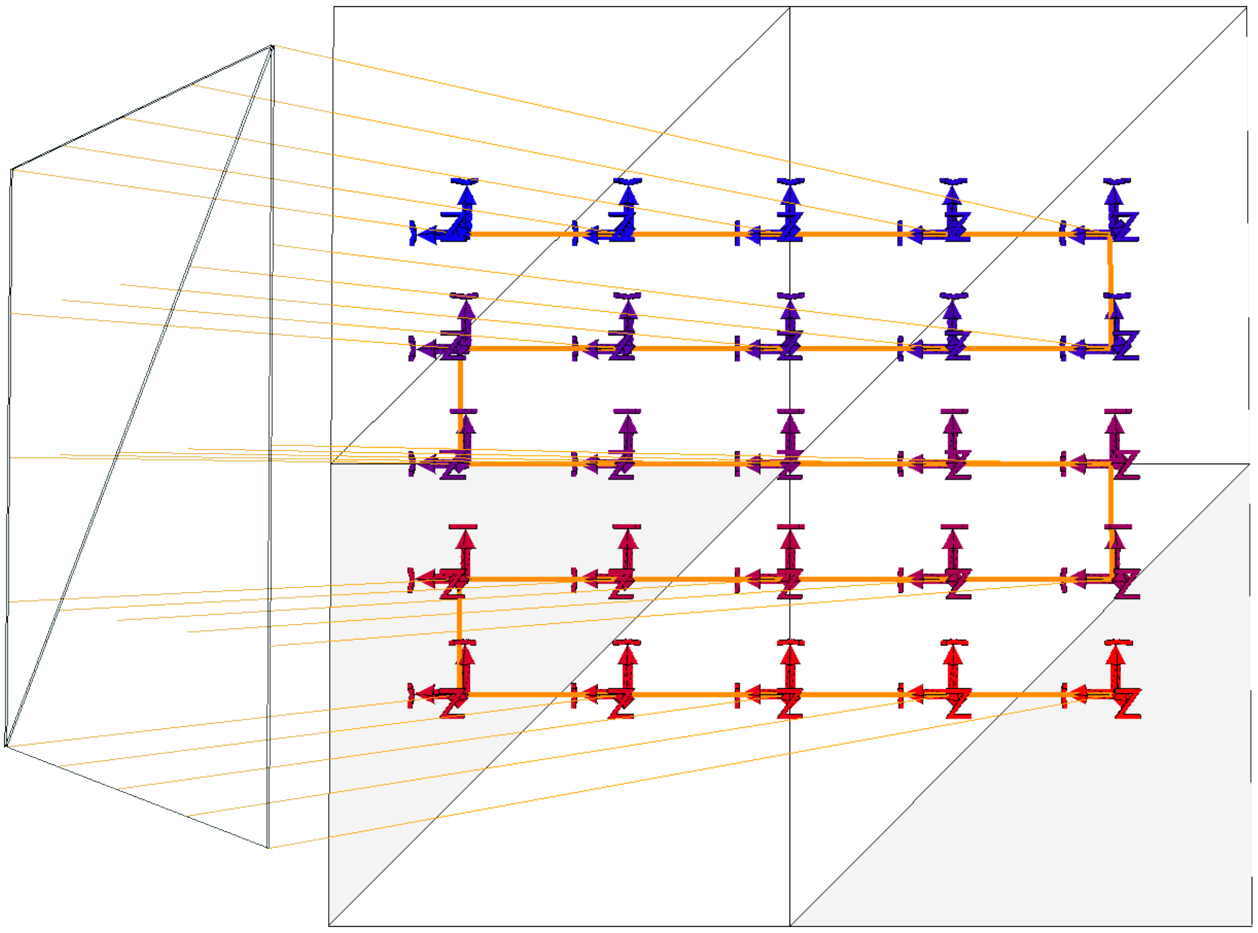


Figure 5.10 Plane-triangle example against a unit plane with a bird's eye view when the approach is tilted.

3. Determine the weight t_t from each tour

4. Return the list of vertices that provided the shortest tour as the best waypath

This simple solution supplies a monumental consequence of $O(n!)$ performance [71]. To understand the significance of this abyssal computational performance, take a graph with $rs_x = rs_y = 10$ using the brute-force function. Assume that a computer can calculate $f_c = 1GHz$ calculations per n. The time to perform this operation, with a relatively small set of points will provide:

$$t_c = \frac{n!}{f_c} = \frac{(rs_x * rs_y)!}{f_c} = \frac{100!}{10^9}s = \frac{9.333 * 10^{157}}{10^9}s = 9.333 * 10^{148}s$$

In other words, to complete this computation, it would take $t_c = 2.959 * 10^{138}$ millennia to finish! Clearly, this is unacceptable. This example shows the clear difficulty when considering

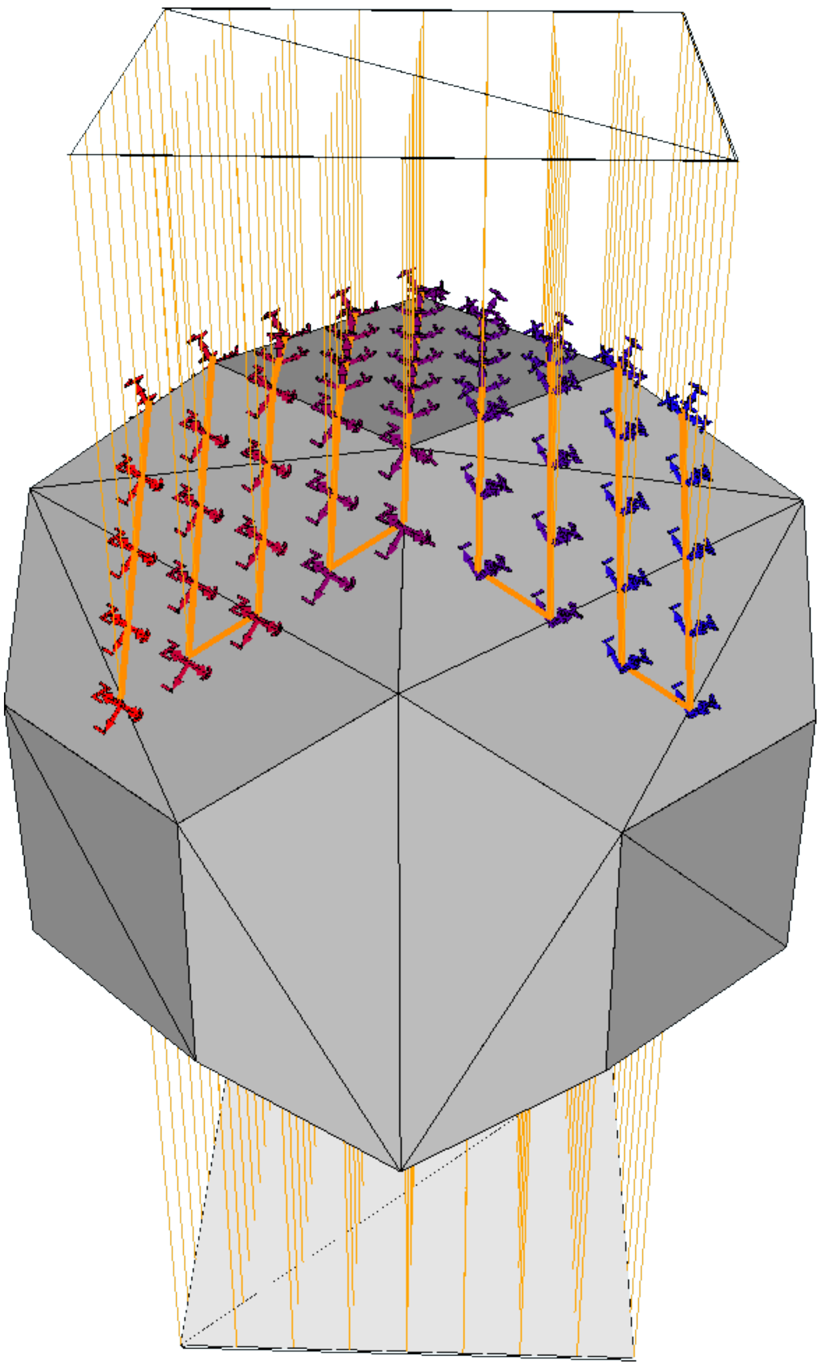


Figure 5.11 Plane-triangle example against a soccer ball showing the approach planes.

perfecting computational time. There exist other exact solutions, such as the deterministic approach branch-and-bound, which breaks the problem into several sub-problems to solve to eliminate the factorial nature of brute-forcing [152] [107]. Branch-and-bound still had an expensive complexity, with exponential complexity for worst-case computational scenarios [164]. For comparison, the

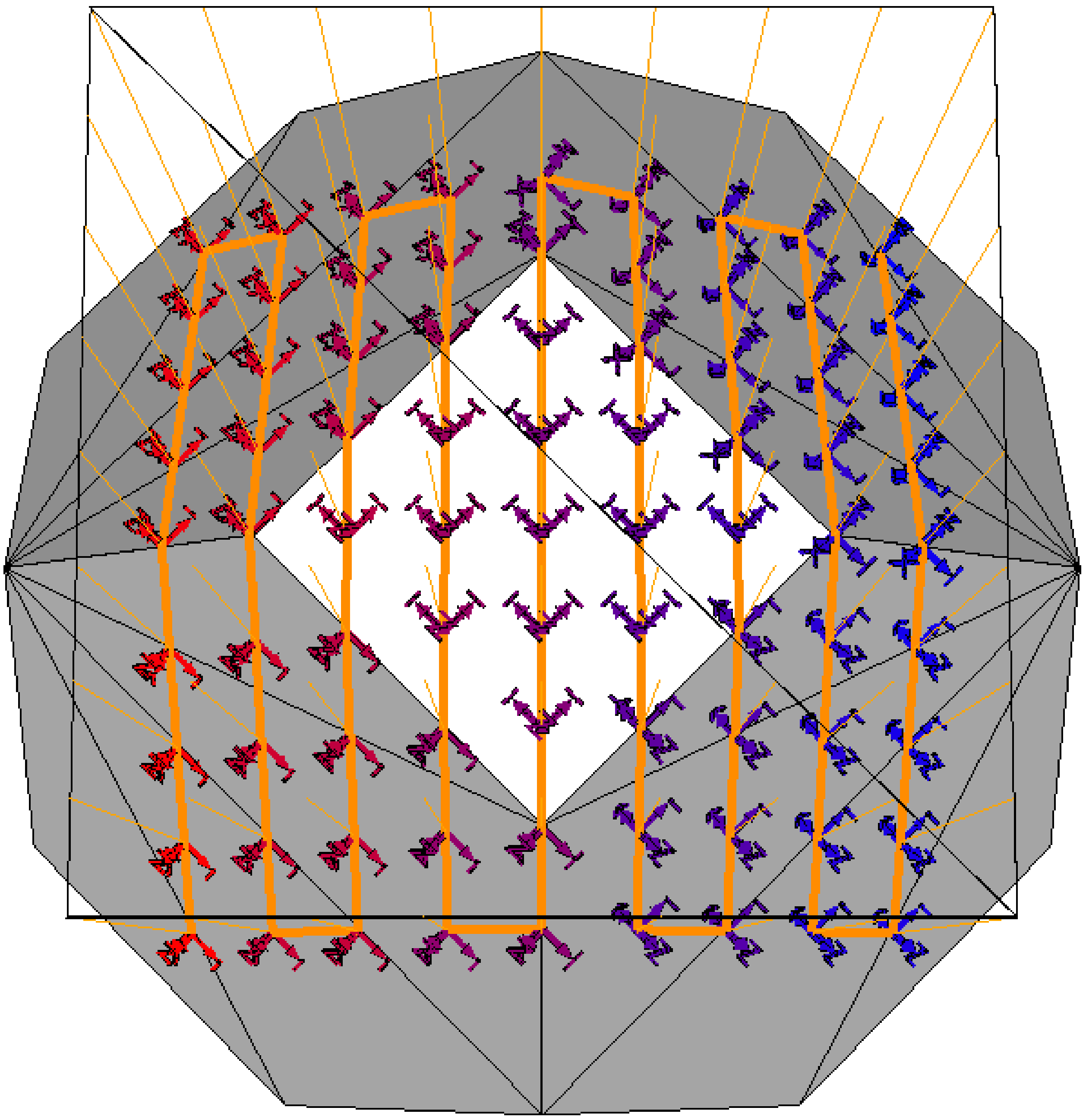


Figure 5.12 Plane-triangle example against a soccer ball with a bird's eye view.

generation algorithm using zig-zag, that is prerequisite for making the points before organizing, has a complexity of $O(n^2)$, with computational time using the same parameters as before:

$$t_c = \frac{n^2}{f_c} = \frac{(100^2)}{10^9} s = 0.00001s = 10\mu s$$

The organizational algorithm used in this report is the greedy approach, which minimizes computation time in attempt to find a better immediate solution [125]. The complexity of the greedy

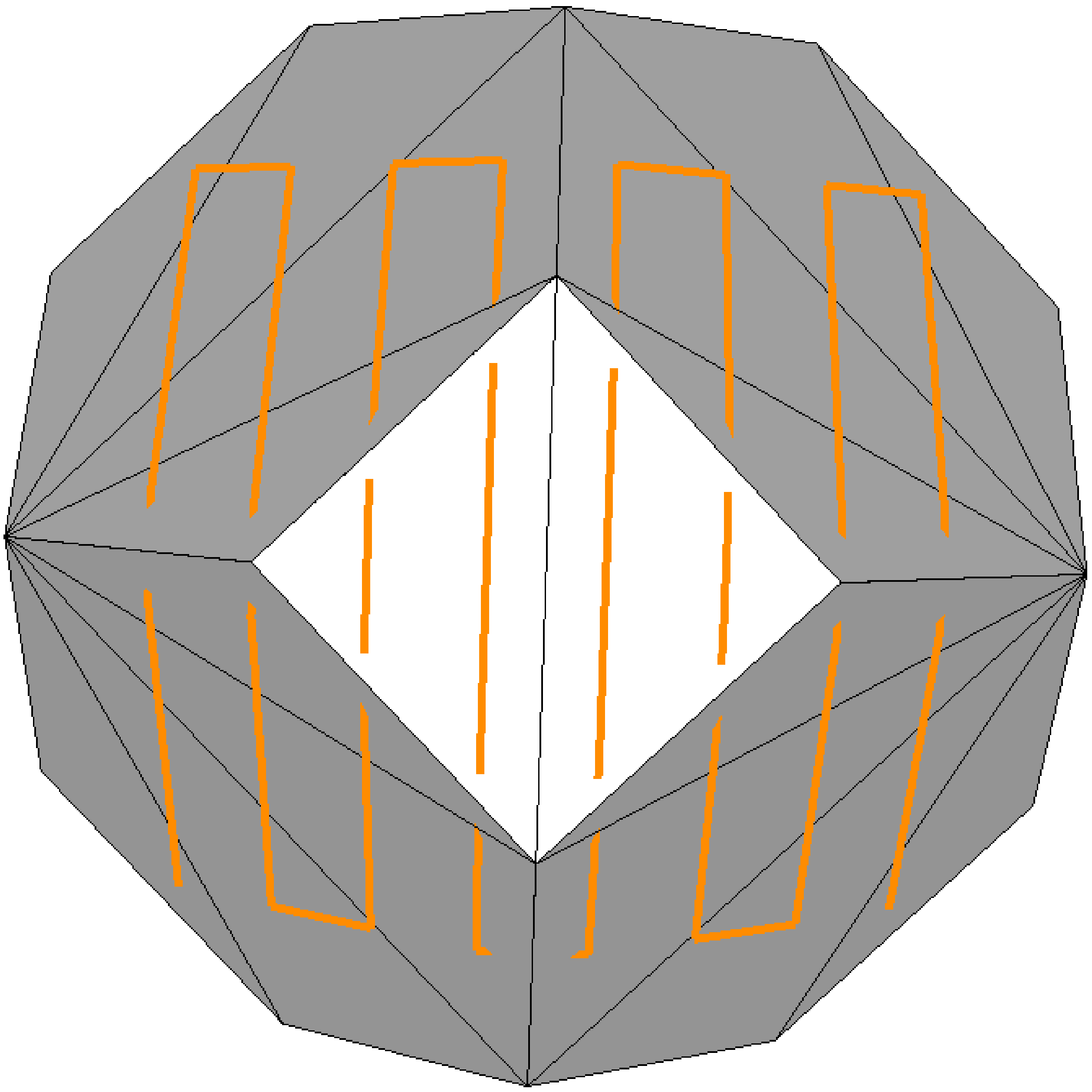


Figure 5.13 Plane-triangle example against a soccer ball showing improper collision.

algorithm itself is $O(n^2 log_2(n))$, and while it has the potential to find the best solution for weight t_t, it isn't guaranteed to do so [3]. There is also a possibility the solution may be equal to the input zig-zag default waypath list, or even worse. Despite this, the greedy approach tends to solve best tour yields around 10 to 25 percent less than the best possible yield from exact methods [89], meaning it's for travel time $t_t$improvement per computational time t_c is rather impressive.

Consider that the greedy algorithm will use the zig-zag algorithm as a base, so applying greedy afterwards will add the complexities together. For usage, then the complexity using greedy will be:

$$O((n^2) + (n^2 + n^2 log_2(n))) = O(n^2 * (1 + log_2(n)))$$

Solving t_c using previous parameters:

$$t_c = \frac{n^2 * (1 + log_2(n))}{f_c} = \frac{10000 * (1 + 6.644)}{10^9} = \frac{7.644 * 10^4}{10^9} = 76.44\mu s$$

In comparison, the greedy approach is approximately 7.644 percent more expensive than the base zig-zag algorithm, approximately due to simplification of computational complexity algorithms, however, is still effective on it's closer approximation to the best solution for t_t. The process for greedy is shown in algorithms 5.3, which uses function 5.4 to solve best points. To be noted, if the best weight as t_t found is worse than zig-zag, then the waypath return will be zig-zag instead. This is done through the function in algorithm 5.5.

Algorithm 5.3 Greedy TSP

Input: The enumeration value of which weight mode to use, $WeightMode$, a list of waypoints, $InputPoints$

Output: A list of waypoints organized using the greedy algorithm, $OutputPoints$, the non-negative value for total weight of movement, $TotalWeight$

```
1:  $length = InputPoints.Count$;        ▷ The number of points inside of the input points list
2:  $OutputPoints = new\ PointsList(length)$;        ▷ Set up the output path as an empty point list
3:                                          ▷ with the capacity at $length$
4:  $currentPoint = InputPoints[0]$;        ▷ Select the current point as the starting point
5:  $currentPoint.Reached = true$;        ▷ Set as reached by default
6:  $TotalWeight = 0$;        ▷ Set up the total weight traversed as zero
7:  **for** $i = 1; i < length; i = i + 1$ **do**    ▷ Iterate through all points after first, solve the best weight
8:      $weight = 0$;        ▷ The local weight solved from the previous
9:      $point = BestGreedyPoint(InputPoints, currentPoint, out\ weight)$;        ▷ Solve
10:     **if** weight $\neq 0$ **then**        ▷ If the weight is solved (it's not equal to zero)
11: ‘                                        ▷ then a valid point was found. Do the following:
12:         $point.Reached = true$;        ▷ Mark the current point as reached
13:                                          ▷ Note: by default, $points.Reached = false$
14:         $OutputPoints.Add(point)$;        ▷ Add the point to the path
15:         $currentPoint = point$;        ▷ Set the current point as the point selected
16:         $TotalWeight = TotalWeight + weight$;        ▷ Append the local weight to total weight
17:     **else**
18:         $break$;        ▷ If no point found, then leave the loop to exit the function
19:     **end if**
20:     $OutputPoints.ResetReach()$;    ▷ On the final list, restore default reached configuration
21:     $return\ OutputList$;
22: **end for**
```

Algorithm 5.4 Best point solver within the Greedy TSP

Input: The enumeration value of which weight mode to use, *WeightMode*, a list of waypoints, *InputPoints*, the current point to compare with the rest of the list, *CurrentPoint*
Output: The best point solved (if not solved, this it is *null*), *BestPoint*, the non-negative value for best weight solved (if not solved, then is equal to positive infinity), *BestWeight*

```
BestWeight = ∞;                               ▷ Set the best weight to positive infinity
                                              ▷ as any other value selected will be better
BestPoint = null;      ▷ Set the best point to null, which is the default if no point was found
length = InputPoints.Count;                   ▷ The number of points in the input point list
for j = 1; j < length; j = j + 1 do           ▷ Iterate through the list of points
    point = points[j];                        ▷ Get the local point with the index of this loop
    if localpoint.Reached == true then
        continue;      ▷ If the local point has been reached, immediately go to the next point
    end if
    weight = GetWeight(CurrentPoint, point);  ▷ Compare the weight of the
                  ▷ input point (CurrentPoint) and the local point, and obtain the weight value
    if weight < BestWeight then   ▷ If the local weight is less than the current best weight
                                              ▷ then set the best parameters as the local ones
        BestWeight = weight;
        BestPoint = point;
    end if
    if weight = 0 then
        break;                ▷ In the case that the weight is zero, then the current point and
                ▷ local point are the same. In this case, then leave the loop to exit the function.
    end if
end for
```

Algorithm 5.5 Best weight selector

Input: The enumeration value (of type *WeightModes*) of which weight mode to use, *WeightMode*, and the two waypoints to compare, *WaypointA* and *WaypointB*
Output: The weight between the two points, *Weight*

```
switch WeightMode do     ▷ Switch based on the weight mode, either being distance or time
    case WeightModes.Distance                 ▷ Simply, the distance between points using
                                              ▷ translational $T_{xyz}$ parameters
        return WaypointA.DistanceBetween(WaypointB);
    case WeightModes.Time                     ▷ Time is to be implemented
                                              ▷ would be a separate function due to complexity
        return SolveTime(WaypointA, WaypointB)
```

5.3 SprutCAM Toolpath Generation

This section discusses waypath generation through the commercial software SprutCAM which specializes in usage with robotic arms. This software works well with predefined CAD models but had trouble with reconstructed models. As explained before, the importance of using reconstructed models is to ensure the positioning and geometry in physical space correlates with the environment in virtual space. If approximation is done, for registering the CAD and assuming the geometry is the same to the CAD in physical space, then usage of software capable of complex path planning may prove useful. Due to the focus of reconstruction in this thesis, SprutCAM will not be examined later, but will be mentioned quickly here, as there is plenty of documentation on SprutCAM available.

SprutCAM enables linear and curved movements along the y-axis on the mesh by selecting the predefined edges of the model, while still generating raster scans. This cuts the fine tuning from point-by-point generation algorithms, with rs_y. SprutCAM also generates "jumps", which is not implemented currently in the custom path planner. A jump allows the probe to move towards a "safe plane" whenever a complicated movement or one that causes a collision is detected. SprutCAM avoids generating paths that cause any collisions, with the robot and its tools and the environment.

Two operations were examined with SprutCAM. The first provides 6-DOF surfacing results normal to the surface of the virtual CAD. This requires a defined CAD model to run, and if a mesh file is desired, then it must be converted to a CAD file such as STEP. This requires an input of faces to be scanned and start and stop curves. This process is shown in figure 5.14. Several strategies are possible which supply different results. The second strategy is 6-DOF morphing, allowing a user-defined region for a surface scan to be conducted within a 2D boundary. This prevents scanning unwanted sections of the model for areas wanted to be examined. Either may be helpful depending on different applications, but both were difficult to work with while using reconstructed mesh inputs.

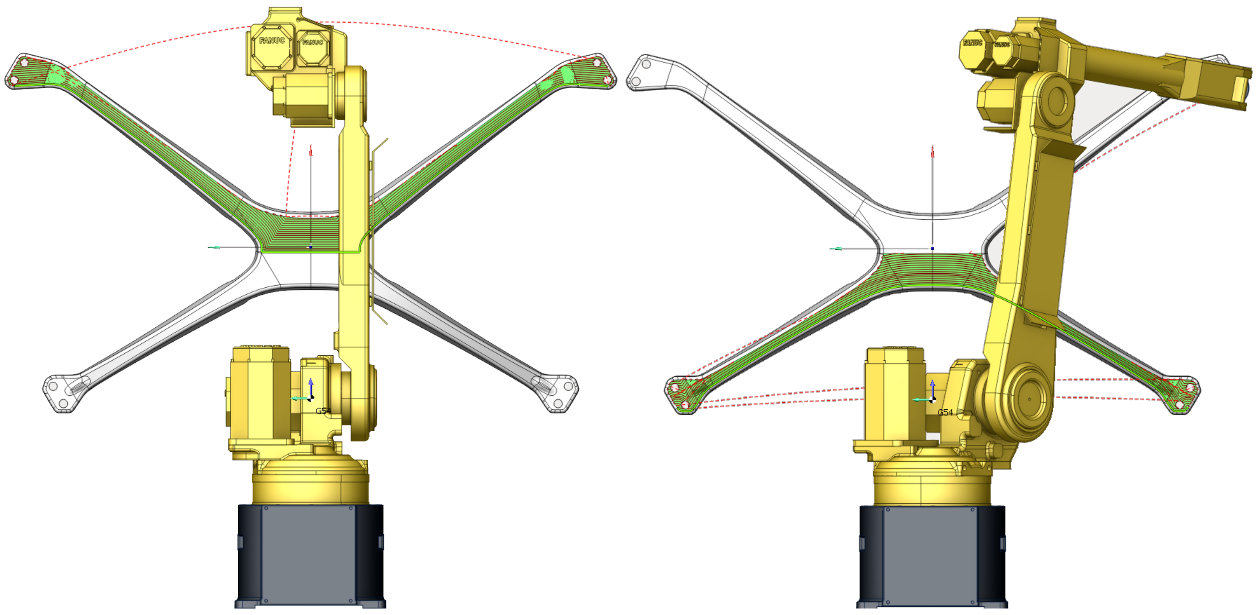


Figure 5.14 SprutCAM path planning using a predefined x-brace model, partitioned twice from top to bottom.

CHAPTER 6

ECA SCANNING AND POST-PROCESSING

Now that the processes prior to scanning have been discussed, which enable surface scanning for complex-shaped objects, it should be appropriate to discuss the scanning procedure, the effects of errors in scanning, and how to deal with these errors in post-processing for Eddy current arrays (ECAs). The parameters used for figures regarding ECA data are the same as shown in the experimental results, with an Ectane 2 and Eddyfi I-flex probe (ECA-IFG-034-500-048-N03S) utilizing 32 channels with a $34mm$ coverage area and $1.25mm$ between rows of coils, running at $2Mhz$ with 5V excitation, with a goal lift-off of $1mm$ away from the scan surface. Complex impedance is obtained through absolute mode for several 3D+NDE points on the image, with the robot running at 25mm/s at 500hz acquisition speed for each set of 32 coils, obtaining an along resolution of $0.05mm$ however this varied. Only the real component of the impedance is evaluated. See figure 7.1 for the system diagram. Flat sample A will be used in these examples as well. With these parameters, corrosion defects at a minimum of $10\mu m$ may be picked up from the ECA system. However, there are unique phenomena that occur when scanning with a robotic arm system. The raw data is a PTC of 3D+NDE points that follows the shape of the material under test, though it is converted later to 2D for simplification of post processing. Fast and full mode scanning will later be discussed as procedures to enhance understanding of errors and post processing requirements, in which only coil 16 will be used. High frequencies are desired for picking up small surface defects, in which the skin depth equation may be used to determine the max penetration of the coils on the defect:

$$\delta = \frac{1}{\sqrt{f\sigma\pi\mu}}$$

Where σ is the conductivity in S/m, μ is the magnetic permeability, and f is the frequency of the probe.

6.1 Errors from Complex Robotic Scanning

The largest concern with ECA scanning with defects requiring such precision was sensor orientation error, however this was not the only factor found through testing. With the high required sensitivity of the ECA to pick up such small defects, errors must be minimized to increase SNR. The lift-off selected was $1mm$, so to prevent collision, the goal was to maintain lift-off errors that do not exceed $1mm$ from either direction. Rotational errors occur on the entire probe, meaning that coils away from the center point will be subject to more lift-off error. For example, along the $34mm$ coils, a 1 degree tilt error along the edge will cause a $0.314mm$ change in lift off in either direction along the tool z-axis, while the coils near the $0.75mm$ away from the center would be significantly less affected by the tilt at $0.0109mm$. The two coils in this array next to the center are coils 16 and 17, which are on opposite rows. The coils on the far edges are coils 1 and 32. A 3 degree tilt error on the end coils would cause a $0.942mm$ error! The center coils would be affected $0.0327mm$ by the same tilt. In contrast, a 1 degree error between the points at $1.25mm$ only causes $0.0109mm$ change between rows, and a 3 degree error causes $0.0327mm$ lift-off in either direction. Tilts also cause changes in interaction between the coil's electromagnetic field and the metallic surfaces, however this phenomenon was not examined.

It was found the most complex error came from robot mastering error due to such rotation issues. This error is complicated as each malignment of the probe itself will be related towards several different rotational errors on the robot arm. This error significantly affects the raw data, as shown in figure 6.1. This was a significant issue in testing, as due to the age of the robots used, reducing this error proved to be difficult as errors below $1degree$ per joint could still add around $1mm$ worth of orientation error in the scan. With this error removed, it is possible to recover locations and intensity of defects with post processing, though there may be loss of signal from the lift-off error. A unique algorithm "array subtraction" was developed to counter these unique patterns using the 3D+NDE PTC.

Another source of error would be any miscalculations from the mesh used for path planning. Coarseness of the face used in a ray intersection causes differences in rotation that would affect the tilt of the probe along a linear path. Figure 6.2 shows the changes in tilt online side the sample, which it may be seen that some lines on the raster path tilt more than others. Likewise, error versus expected lift-off location-wise would be appended to these issues. This may be fixed in post

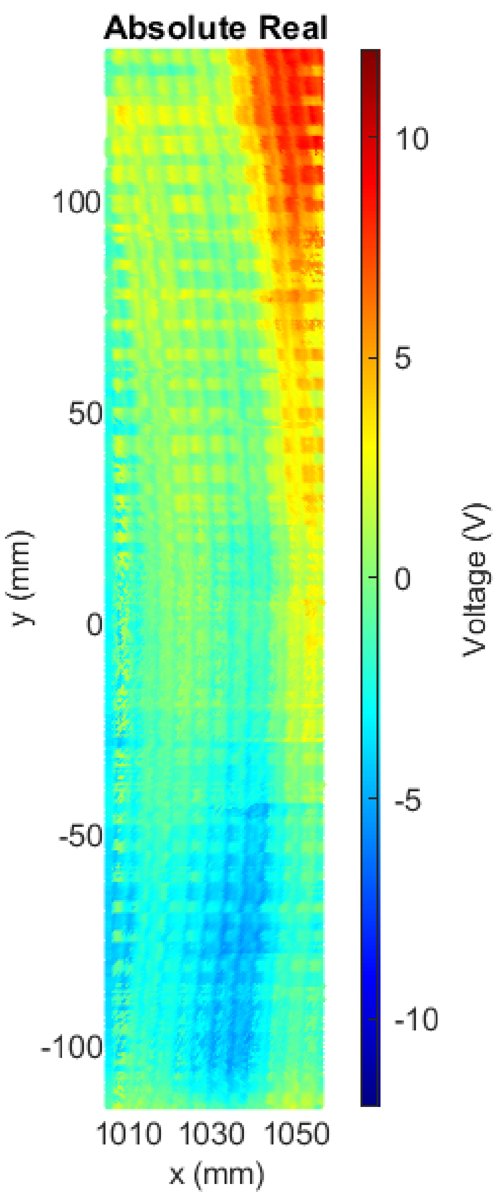


Figure 6.1 Raw PTC data from full scan.

processing through detrending the scan lines. To simplify this process, detrending is applied along each raster scan. There may also be errors from the alignment process, which would be a consistent orientation error across all points. This may also be resolved by detrending along the entire sample.

Since absolute mode scanning for ECT was used, it was found that effects of temperature fluctuations on the coils would significantly affect the acquired voltages [65]. These temperatures could be from the surrounding scan environment, or from the coil heating up from voltage being pushed through the system, or through any other system components generating heat. Figure 6.3

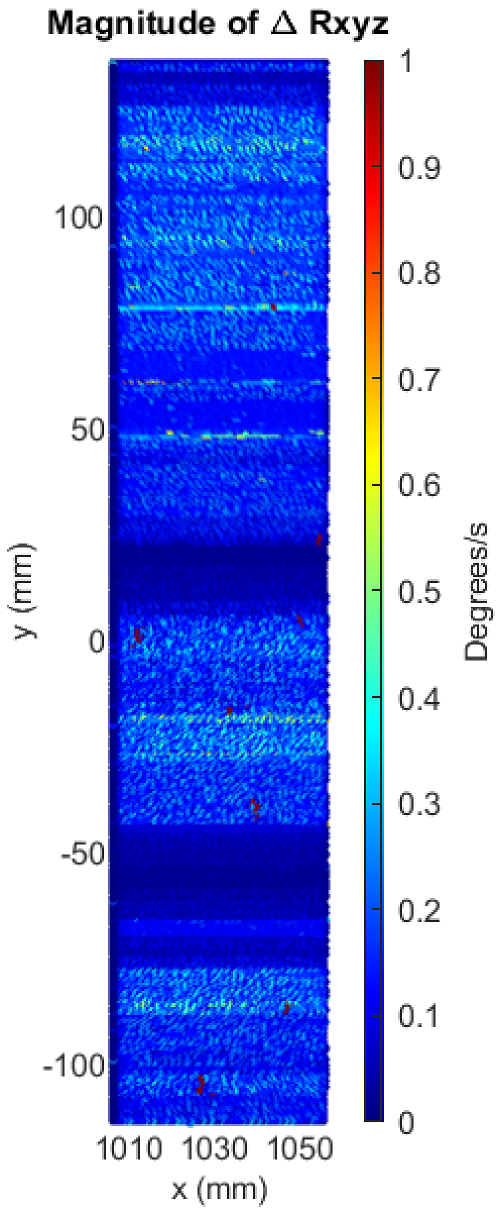


Figure 6.2 Change in rotational tilt for each rotation on the probe, full scan.

shows heat fluctuations, with a coil placed statically above a steel sheet sample. After 7 minutes, the returned voltage for each coil saturates towards the maximum voltage settings! This may be decreased through waiting a significant period while exciting the coils, such as 1 hour at minimum which was applied across all scans later. Afterwards, utilizing coil gain calibration tools from the Ectane 2 the signal may stabilize near $0V$ on the background. However, since it is assumed that over a longer scan the temperature may still change, detrending of the signal per coil over time is utilized.

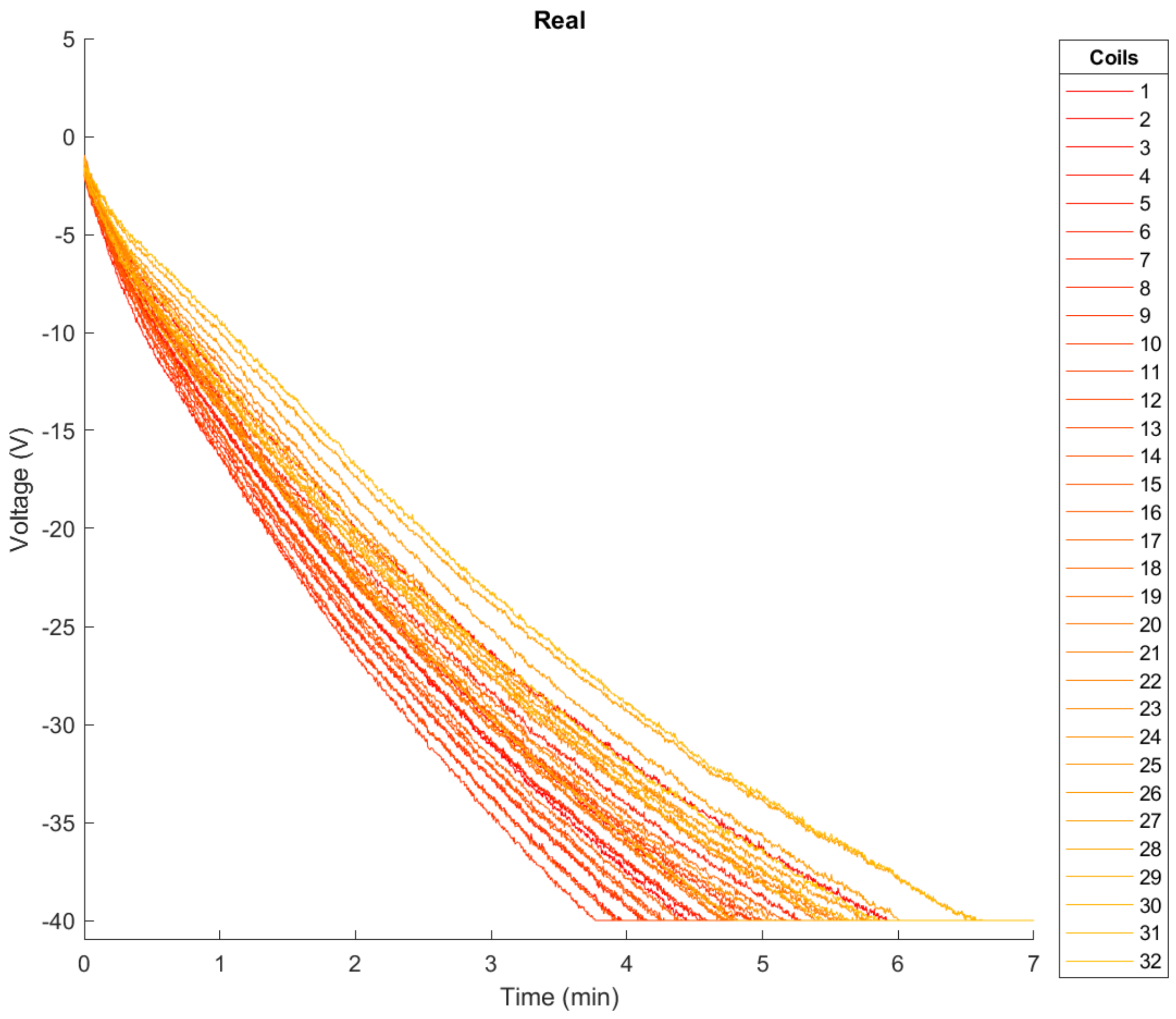


Figure 6.3 Heat fluctuations for every coil.

As the returned signal of the coils run asynchronously from the toolpath returned from the robot, there are delays which affect the actual location of the signal with respect to space. While the coils maintain a $500hz$ acquisition rate, the robots tended to fluctuate between $10hz$ and $70hz$. While this does not affect the gain of the coils, effecting the quality of intensity per defect, it does affect the placement of the data which is later used for quantifying the ECA signals with the ground truth results of the digital microscope. The solution was to scan at a slower speed of 25mm/s to reduce synchronization error. A proper solution towards this synchronization error may allow for much faster scanning of samples, as the selected robots can move up to $2m/s$ if so desired!

6.2 Scanning Procedures to Examine Errors

It was selected to use two different scanning movements of the array to better understand and utilize post processing, which will be later used in the experimental results. The typical usage of an ECA is to scan with several coils to increase coverage of a scan while significantly reducing scan times. For the sake of this work, this is considered a "fast" scan. However, it is advantageous to examine the effects of errors for each coil individually for a scan. A "full" scan was considered, meaning that each coil will obtain an image of the samples through redundancy. The full scan, while being slower due to redundancy, also requires more coverage so that several coils may hit the same defective locations. Both types of scans are conducted using the same aligned mesh but will have different waypoints for pathing. Despite the assumption that a full scan should provide a better or similar quality image in terms of SNR. This is due to where the points end up intersected and the interaction they have on tilt and lift-off as previously mentioned. Another is not that due to limitations of the robot's capacity to hold a large number of points; specific path planning was utilized to allow only 2 points for a linear movement. This switches directions of rasters, with full scans appending rasters along the y-axis while fast scans append along the x-axis, swapped due to the nature of the scan types. In this configuration, the required appending will follow along the curved section to provide more points for the curved sections intentionally. The differences in errors should be for direction of synchronization errors and locations of the way points. For simplicity, only coil 16 which is near the center of the array will be demonstrated for full scans, however later full analysis of coils will be done in graphs instead of figures of the image.

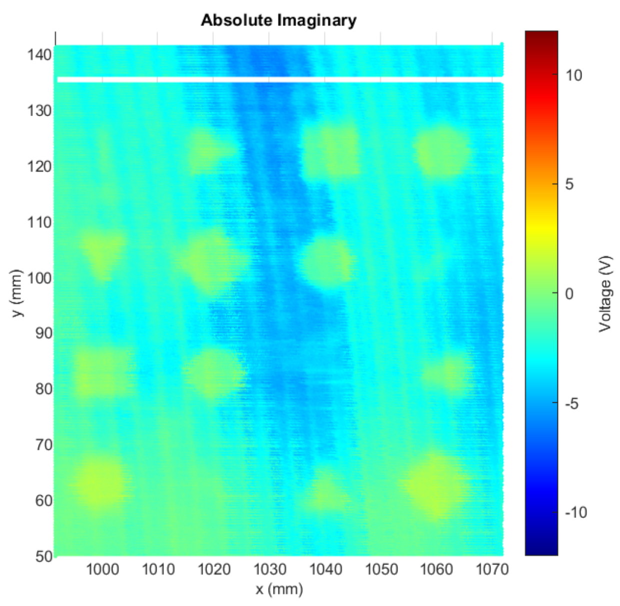
6.3 Post Processing

There are several errors that have been discussed, this section covers processing procedures to minimize such errors. Post processing to compare both the digital microscope data from a Keyence VHX-7000 (considered as VHX) and the ECA results are also discussed, in which common defect regions are compared on a pixel-by-pixel basis between defect and background. The total procedure in order of algorithms used will be in the experimental section.

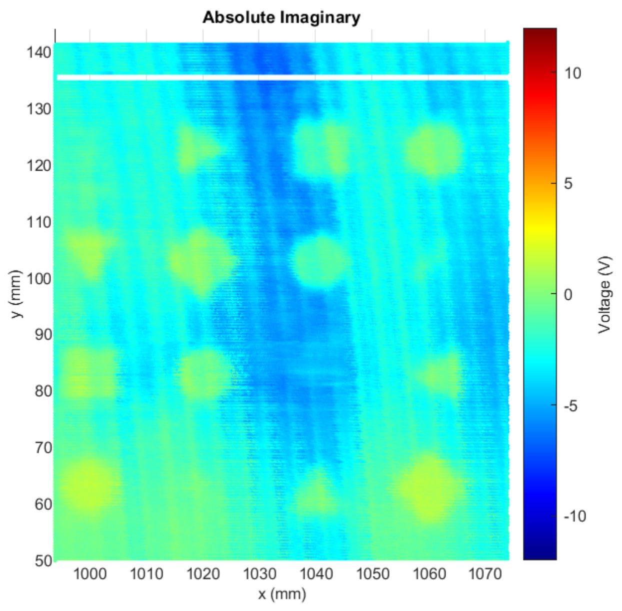
6.3.1 Array Subtraction Algorithm

This section covers how to reduce errors that are common along all coils of the array. This is important for removing complex errors that occur due to robot mastering miscalculation. ECA scanning has different modes to conduct scanning. One of these mods is "differential mode", in which two adjacent coils on a row are subtracted. This can be done either through analog with a circuit to subtract the signals such as a Wheatstone bridge, or digitally by simply subtracting the signal between the two. By subtraction, common errors between the two signals may be removed. This is a method that can be used to remove the error from robot mastering. However, a differential image will obtain an "edge-like" pattern. Let us say there is a coil A and B, with B lagging A. A reaches a defect while B is in the background. There should be a measurable difference between A and B in this scenario. Let us say B reaches the defect while A is still on the defect. The subtraction of A and B should yield a zero result. Likewise, if both A and B move to the background, the subtraction will yield a zero result. In those scenarios when A and B are above the same defect, they will give the same result as the background! This may be seen in figure 6.4, which sample has several sand-blasted defects with shapes such as a triangle, square, circle, and line of 1mm in length and measured depths of $10\mu m$. Note that imaginary is used in this case. The edges of the shapes are picked up, but the area of the defect in between is identical to the background output. This makes comparison between the ground truth complicated, as the ground truth shows the near exact volumes of defects from a surface perspective rather than an edged image. The background is not perfect, as the loss of data due to lift-off still occurs.

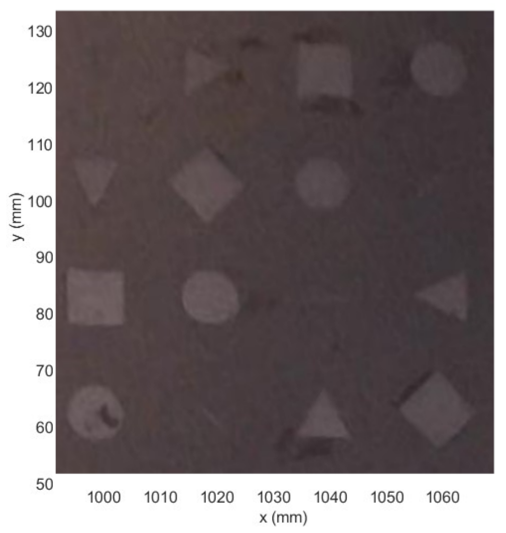
To maintain the areas of inside the defects, an algorithm inspired by differential mode scanning was used. This algorithm determines the normalized mean per coil and subtracts each voltage value with similar normalization from this mean per each 3D+NDE point. Normalization is used to reduce biases between more saturated coils. The subtracted normalized value is then scaled back to the voltage prior to normalization with the common signals removed. This process significantly reduces the effects of the robot mastering error, as shown in figure 6.6. The algorithm is shown in algorithm 6.1 and shown in figure 6.5. This algorithm tends to work better on surfaces with



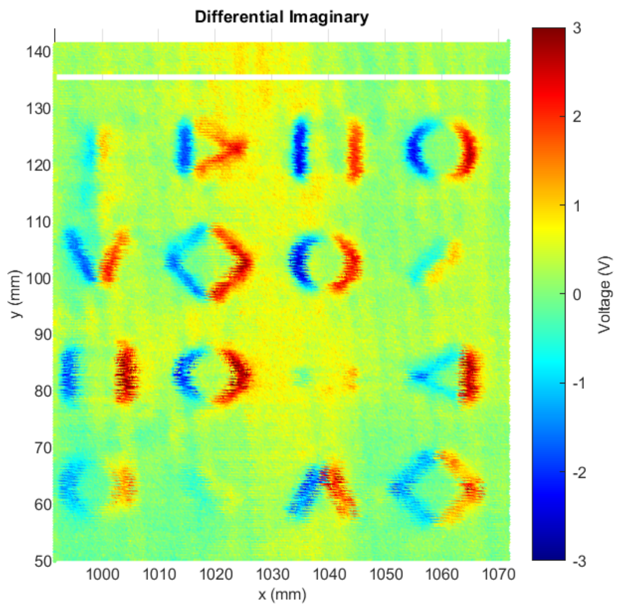
(a) Raw coil 14 data



(b) Raw coil 16 data



(c) From camera



(d) Differential 14 and 16

Figure 6.4 Differential mode on sandblasted shapes sample.

variable defects, such as corrosion. Large defects with constant values tend to overbias and cause "ghosting" from the array.

6.3.2 Detrending

This section will discuss detrending and the entire process of post processing up to cross correlation. In can be seen in figure 6.6 that the top part is more saturated than the bottom. This

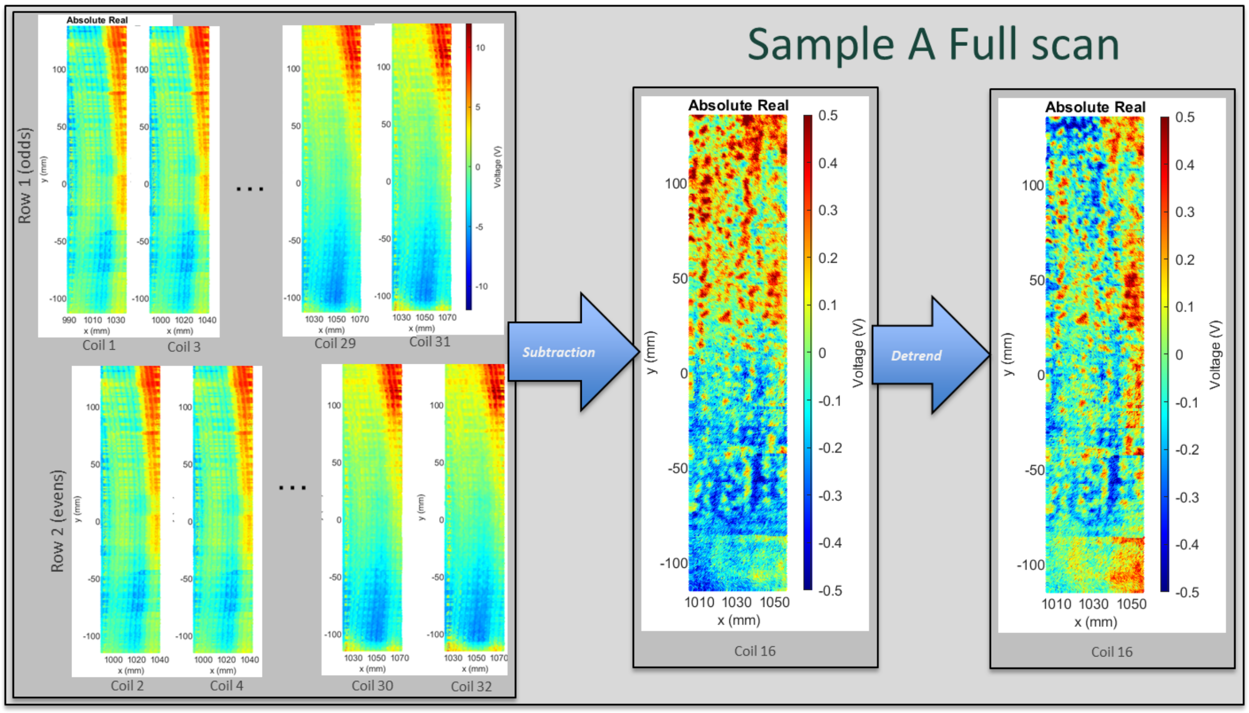


Figure 6.5 Array subtraction procedure.

Algorithm 6.1 Array Subtraction Algorithm

Input: ECA voltage data with all coils, $ecaIn$
Output: ECA voltage data post subtraction with all coils, $ecaOut$

```
1:  $normVals = zeros(size(ecaIn));$                          ▹ initalize values holding normalized vals
2:  $ecaOut = zeros(size(ecaIn));$                            ▹ initalize values holding the output
3:  for $c = 1; c < size(ecaIn, 1); c = c + 1$ do             ▹ c is the coil index
4:      $N = norm(ecaIn(c, :));$                              ▹ normalize the voltage time signals per coil
5:      $normVals(c, :) = n$                                  ▹ place time signal per coil
6:  end for
7:  $normVals = normVals - mean(normVals);$                   ▹ zero the normalized values
8:  for $c = 1; c < size(ecaIn, 1); c = c + 1$ do             ▹ c is the coil index
9:      $ecaOut(c, :) = revertNorm(normVals(c, :), ecaIn(c, :));$   ▹ normalize to voltage
10: end for
11: $return\, ecaOut$
```

may be due to the registration error and heating variables shown in figure 6.3. There are also factors such as rotation that would need to be detrended per line, as shown in figure 6.2. Detrending is the process of zeroing regression to "squish" down results, reducing these effects. The regression may be linear or of higher orders. Mean or median subtraction may be used to zero a section of the data. For a different view, figure 6.8 shows the subtraction algorithm with respect to time, and figure 6.9

Algorithm 6.2 Normalize Vector

Input: ECT data, $ectIn$
Output: Normalized ECT data, N

```
1: $V = ectIN - mean(ectIN)$;                          ▷ zero the voltage
2: $N = (V - min(V))/(max(V) - min(V))$;               ▷ normalize
3: $return\, N$;
```

Algorithm 6.3 Revert Normalization

Input: Normalized ECT data post subtraction, N, original ECT data, $ectIn$
Output: Reverted ECT data, $ectOut$

```
1: $mEctIn = mean(ectIn)$;                             ▷ obtain the mean
2: $V = ectIN - mEctIn$;                               ▷ zero the original voltage
3: $V = (N * (max(V) - min(V))) + min(V)$;             ▷ revert voltage from normal
4: $ectOut = V - mean(V) + mEctIn$;                    ▷ undo zeroing
5: $Return\, ectOut$
```

is the front scan.

The first order of detrending occurs on the 3D+NDE PTC after subtraction. Detrend is first applied along the scan with respect to time for each coil to remove heating effects. To reduce tilt alongside a coil row, for each time instance of row data, the data is detrended. The results are shown in figure 6.7. There is still tilt from the orientation. To simplify this, the PTC is converted into 2D through scatter interpolation. Once converted, planar detrending is used using first then third order detrending on the 2D regression. This reduces complex phenomena from poor tilting. The result is shown in figure 6.10 with trend from 2D included.

6.3.3 Cross Correlation between ECA and VHX

This section will discuss cross correlation (xcor) to position the VHX data, which has a smaller area, to the larger ECA. As the two data sets will be aligned differently, an algorithm was developed and will be shown that approximates this rotation through iteration and comparing the best xcor value. Errors may occur if either the data of the ECA are obstructed, or the defect is simply outside the image itself. For later results, if the defect is shown to be outside the ECA data, then it is ignored in performance analysis.

Xcor is the process of multiplying two images together that are scaled with respect to eachother. Keep in mind that the data at this point have been converted to 2D instead of PTC. When two

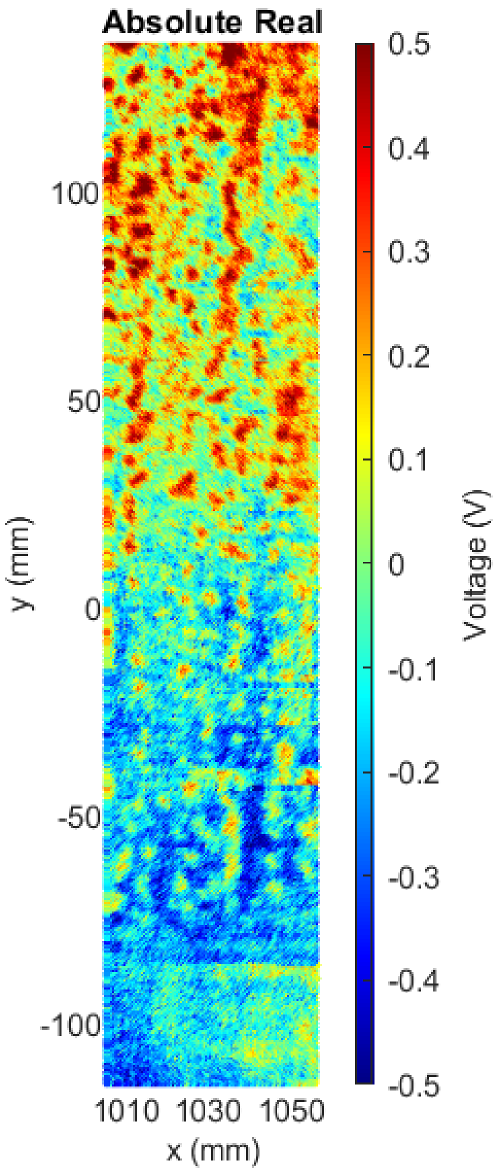


Figure 6.6 PTC data after the array subtraction algorithm is used, full scan.

patterns match, their xcor value should increase. The multiplied value for our voltage values from the ECAs and depth values from the VHX would be $V \times m$, which is not a usable value. To counter this, the values are normalized between 0 and 1. The maximum value from cross correlation will determine the best match, which in turn, also gives the position in between the two images.

An issue that occurs from comparing two images from different scans together is that they will also have a tilt along XY and need to be counted for. To approximate this rotation, an algorithm was made that checks the cross correlation per each rotation. This was done here rather than later, as it

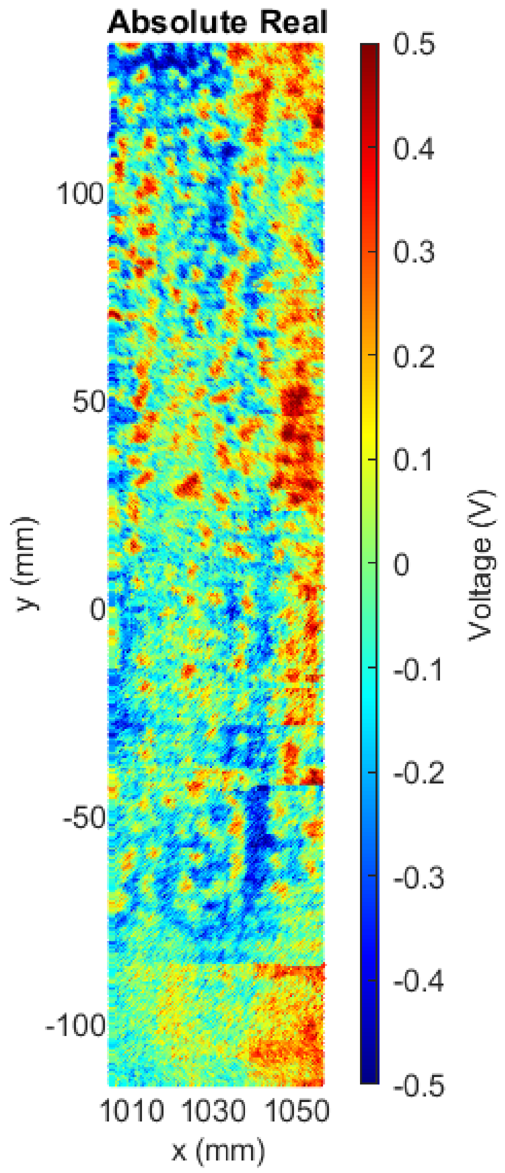


Figure 6.7 Detrended data on the PTC, full scan.

would be less taxing left outside the intersection testing. When the rotation is found, it is applied to the VHX image with the cross-correlation location found. This value, as well as the area from the VHX image, then crops out the ECA data. Both cropped images are then used for intersection testing to obtain objective information about the locations of the defects per each. These cropped images are shown in figure 6.11 using defect d1 from sample A. The patterns of each can be seen but are much sharper in the VHX image. This is due to convolution and other factors of processing for the ECA image. Figure 6.12 shows the ECA and Xcor full images with respect to figures 6.11.

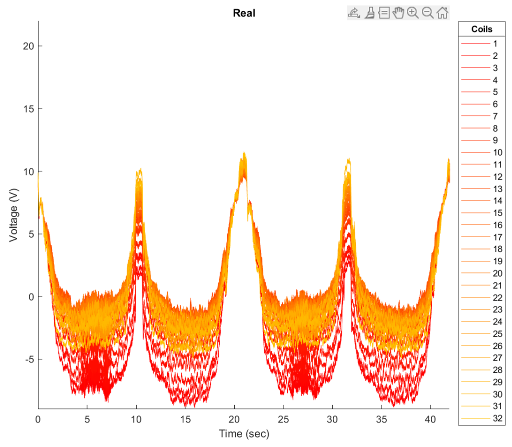
6.3.4 Intersection between ECA and VHX

This section will discuss intersections to match localized ECA and VHX values. This intersection requires the masking between defect and background for both ECA and VHX, and simply compares the two. The intersected image will have three enumerations: defect, background, and no intersection. This is done on a pixel-by-pixel basis, in which the percentage of intersected pixels versus total pixels is calculated. As voltage and depth are not perfectly aligned so 0V and 0mm due to the nature of them being different scanning procedures, an algorithm was developed that will determine the best intersection through iterating different voltages. Unfortunately, due to voltage changes due to gain calibration as required to reduce temperature errors, this voltage fluctuates between different scans. It will also discuss why intersection does not reach 100% due to errors obstructing data, convolution of the ECA, and synchronization errors between the ECA and robot toolframe data streams.

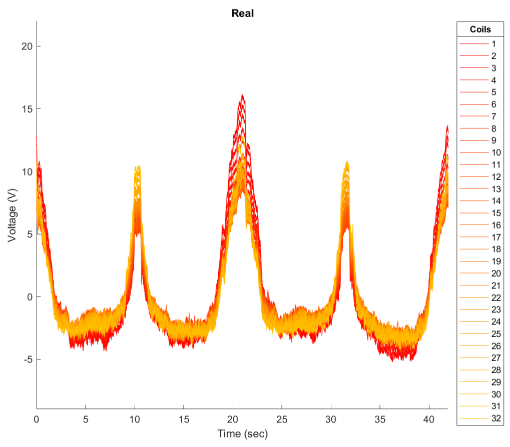
For pixel-to-pixel intersection, both the ECA and VHX samples require alignment together, which is done previously through xcor as discussed. The ECA and VHX images are placed into a threshold filter to obtain masks between "defect" and "background". For VHX, values below 0mm are considered as defects, while above are considered as background. For ECA, voltages over a variable amount are considered as defects, while under are considered as background. Examples are shown in figures 6.13 from the ECA and VHX images from figure 6.11. The masks are then intersected to find common regions between defects and backgrounds. Figure 6.14 shows the results of this intersection, with the white regions being the defect, the black regions being the background, and the gray regions being no intersection. For the fast scan, figure 6.15 shows the ECA data from figure 6.8, along with the translated VHX data, and with xcor and intersection information parsed from ECA vs VHX. The final intersection percentages, which is the ratio between the intersected pixels versus all pixels, is used for performance testing. Despite issues from error due to convolution, data synchronization, and some post processing, the results are promising for efficient detection of corrosion using robotic inspection.

6.3.5 Future Prospects for Voltage to Depth

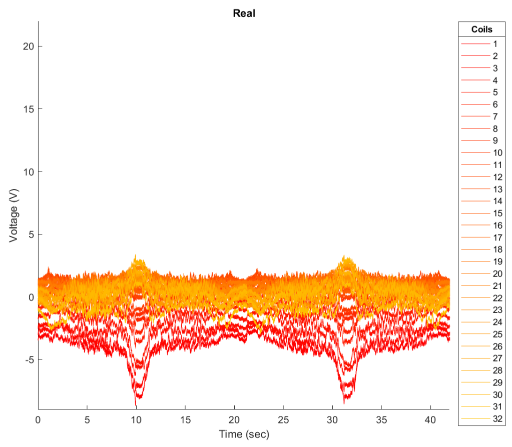
This section will briefly discuss the difficulties of voltage to depth. There are procedures that make it difficult to obtain this information, such as the array subtraction algorithm and detrending which recovers information at the cost of this difficulty. To obtain voltage to depth, a calibration sample is used in which voltages are mapped to depths. This can be done through fitting algorithms, in which exponential decay or tangential models may be recommended. Unfortunately, with the variables due to calibration and variance in lift-off, this was ineffective for obtaining quality results from testing which are currently ignored in this work. There may be prospects using improved calibration methods, different instrumentation allowing for more control of gain calibration, or using AI tools or simulation to retrieve the lift-off and desired depths for ECA.



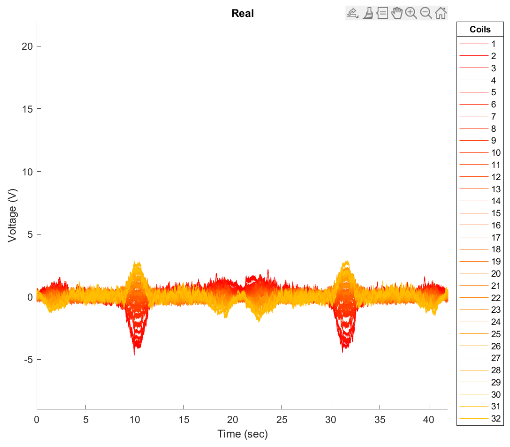
(a) Raw



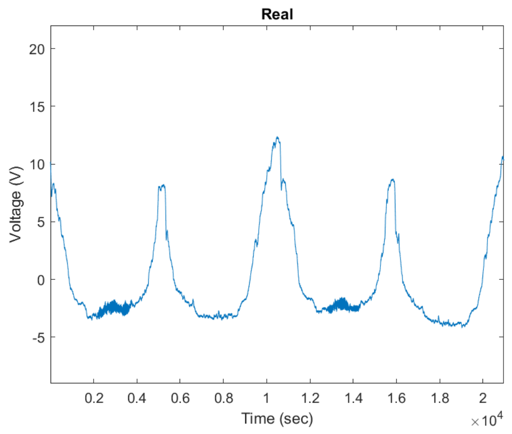
(b) Detrend only



(c) Subtract only

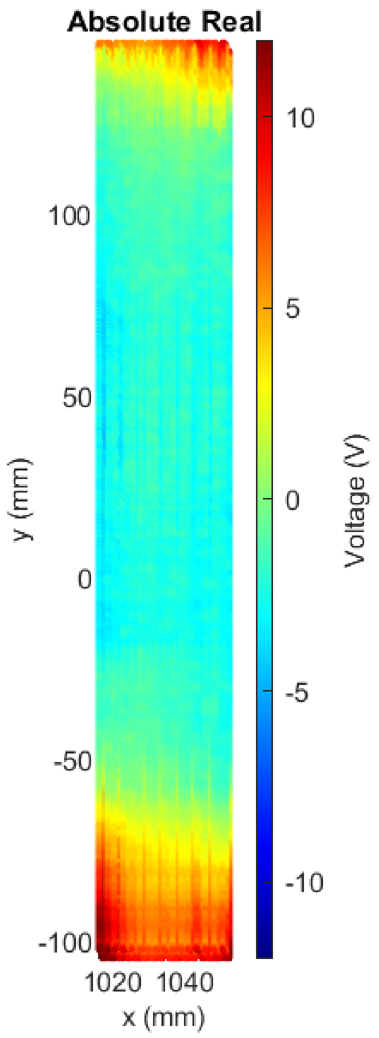


(d) Detrend then subtract

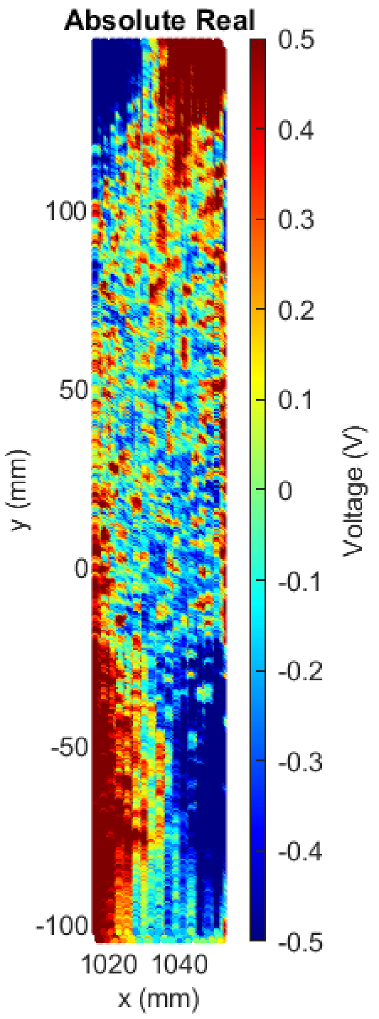


(e) Subtraction amount

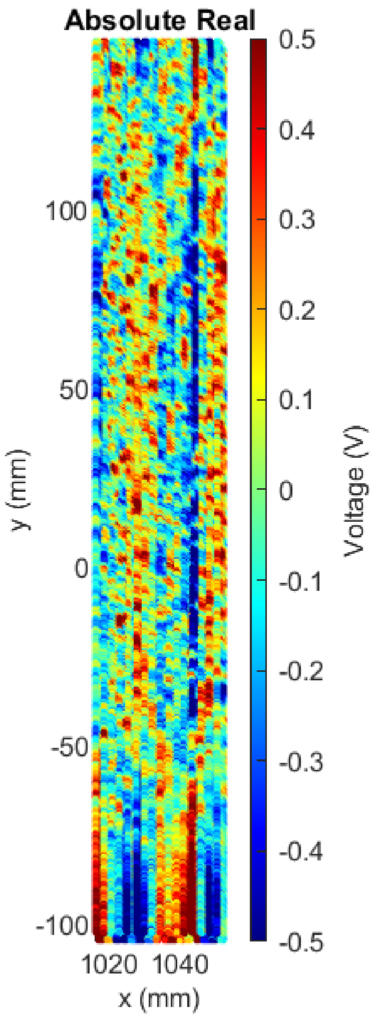
Figure 6.8 Side view of each coil w.r.t. time, fast scan.



(a) Raw



(b) Subtraction



(c) Subtraction and detrend

Figure 6.9 Front view of figure 6.8.

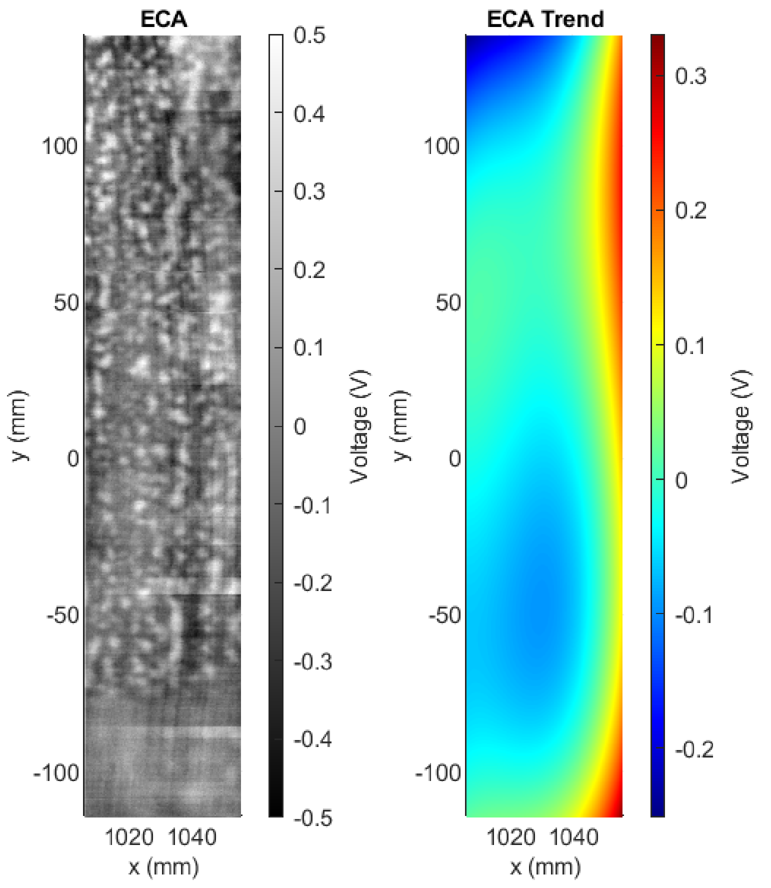

(a) Processed (b) Trend

Figure 6.10 Final processed ECA after 2D trending, full scan.

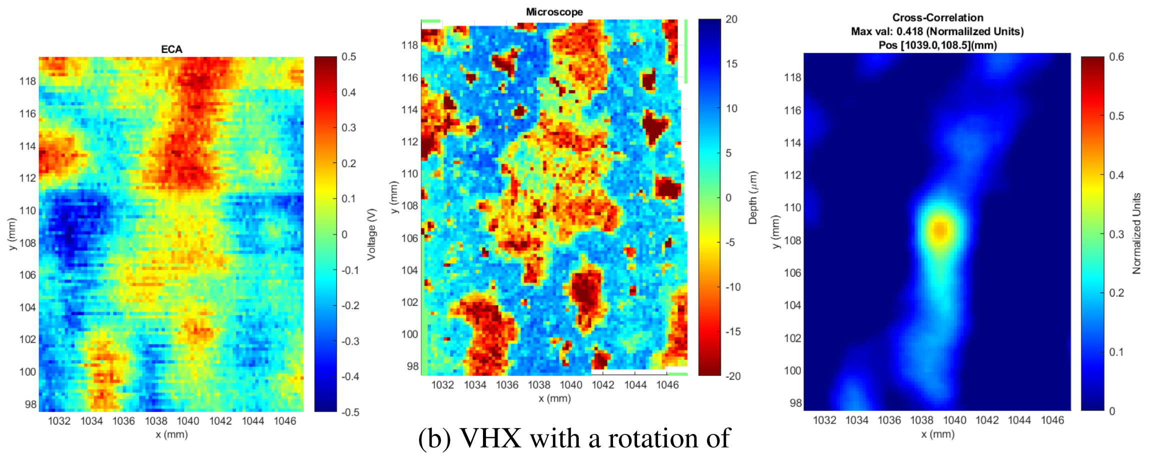
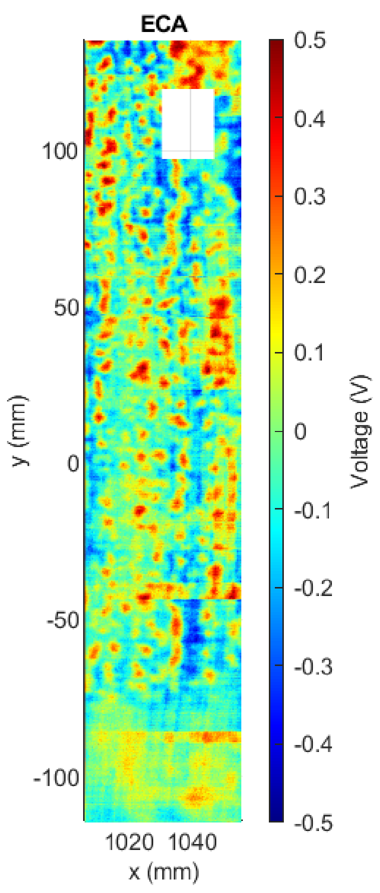
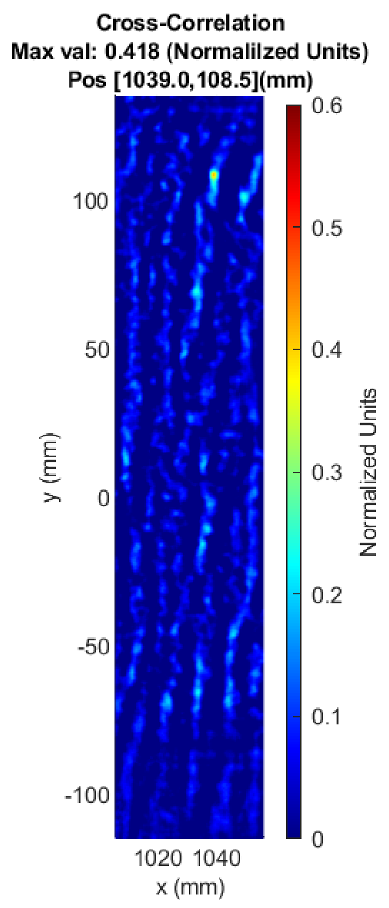

(a) ECA (b) VHX with a rotation of −2.6° (c) Xcor at 0.418nu

Figure 6.11 Cropped xcor information for ECA compared to VHX, full scan.

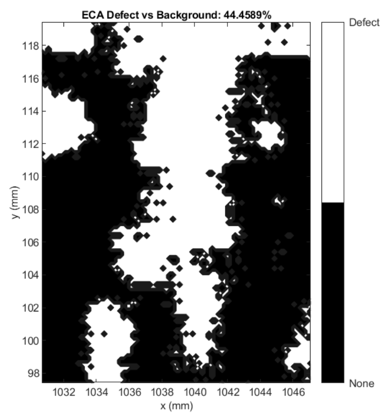


(a) ECA with croppedA cut

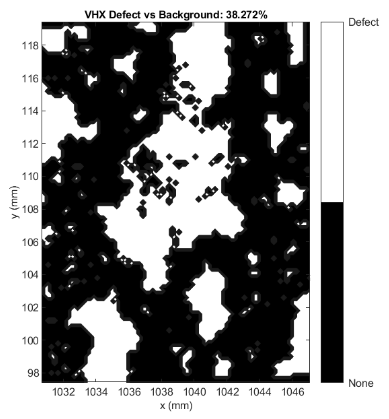


(b) Full Xcor image

Figure 6.12 Full scans showing cut ECA and full xcor images.

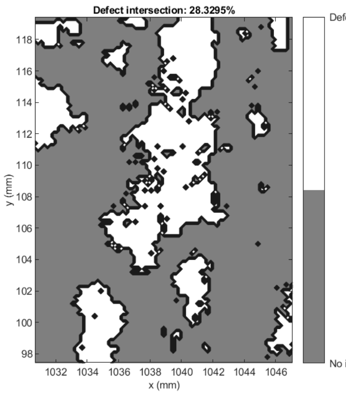


(a) ECA mask at -0.05V

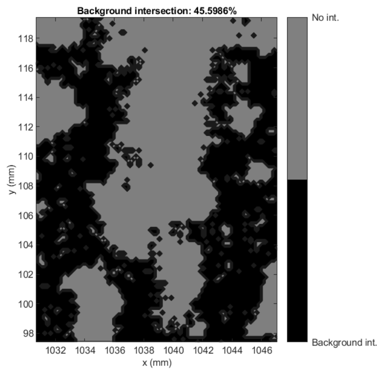


(b) VHX mask

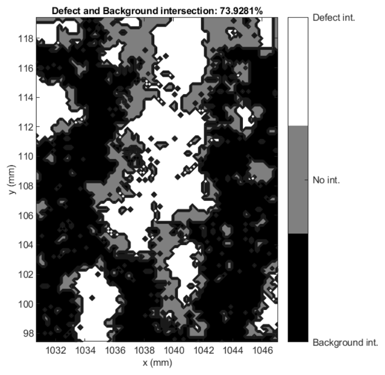
Figure 6.13 Masks for defect versus background using figure 6.11, full scan.



(a) Defect intersection

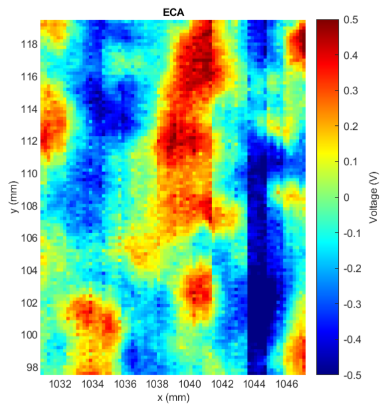


(b) Background intersection

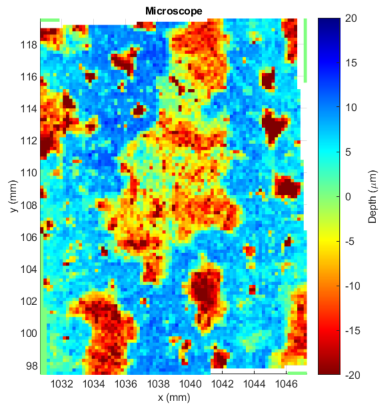


(c) Both combined

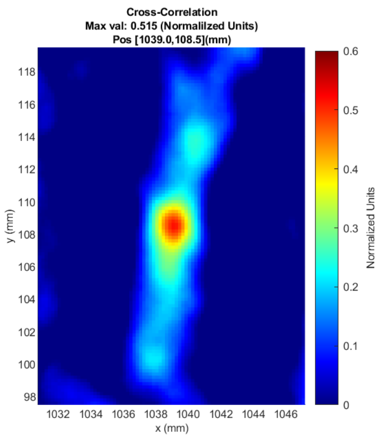
Figure 6.14 Intersection between defect and background, with the number of intersections at 73.9%, full scan.



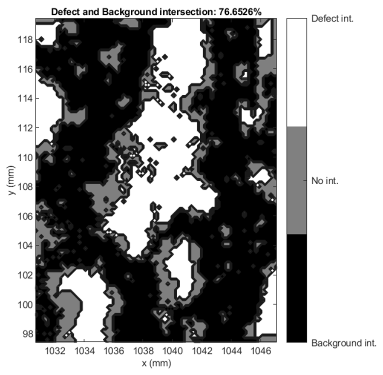
(a) ECA thresh at $-0.1V$



(b) VHX rotation at $-2.5°$



(c) Xcor at 0.515nu



(d) Intersection at 76.7%

Figure 6.15 Fast scan showing ECA, VHX, xcor, and intersection.

CHAPTER 7

EXPERIMENTAL RESULTS

The results shown will mostly cover the complex robotic scanning setup and ECA results for corroded samples. Voltage to depth through ECA is briefly discussed. In support for objective 1, UT inspection on a curved CFRP car piece is also conducted, using the same methods for path planning until NDE is black boxed. The topics discussed will be sample preparation, the reconstruction and registration results, ECA versus digital microscope (VHX) results, and the performance of each scanning mode for full and fast configurations. See 7.1 for the system diagram. A refurbished Fanuc ARC Mate 100ib robot arm is used to manipulate the ECA and calibration laser. The robot arm was calibrated so the joints would be less than 1 degree of error. A CR-Scan 01 structured light (SL) reconstruction device is used to obtain reconstructed images of the samples, which were placed in a static location alongside the robot. To register the reconstructed mesh from the SL frame to the robot frame, a Banner LM precision measurement sensor (LM150KUQP) was used as the calibration laser. this laser would be pointed towards several calibration marks that would also appear in the reconstructed mesh that would measure the location of those points with respect to the robot. The length of the laser is output as an analog voltage signal converted to depth, which has a range in between 50 to 150mm and an accuracy of $4/mum$. There was an error due to joint error from robot mastering miscalculation and slight tilt from placement on the toolframe itself. To reduce error from tilt, the laser was separately calibrated so when lifted upwards from a calibration mark, the laser's XY position would remain the same at a large distance. This rotation was $0.40°$ and $-0.24°$ alongside x and y axes of the tool. The laser was lifted $60m$ away from the calibration marks while depth information was obtained while the robot tool was at a standby to prevent jitter. For information about how the laser is used, see section 4.1. The layout of the ECA probe with respect to the environment is shown in figure 7.9.

An Ectane 2 Eddy current instrument was used alongside an Eddyfi I-flex probe (ECA-IFG-034-500-048-N03S). This probe operates at a base $500khz$, however, using the strong gain calibration tools of the Ectane 2, $2Mhz$ is utilized in order to pick up smaller defects. 32 channels are utilized on the probe in absolute mode, which obtains the impedance voltages for real and imaginary. Only real is considered to reduce the number of figures in this work. The coils are exciting with $5V$ The probe has a $34mm$ coverage, with $1.25mm$ between coils. The desired lift-off was 1mm away from the samples, with enough room to prevent collision from rotational errors. The probe was also calibrated to prevent major tilt from the placement on the tool along the coil rows, at $-1.815°$ along y. During scanning, the information from the ECA obtaining defect information is synchronized with the robot's toolframe position to localize the defect information to the robot's frame. The robot ran slowly at 25mm/s, even though the maximum velocity of the robot is around 2m/s. This is due to synchronization issues between the ECA and robot toolframe which would displace expected information. The ECA ran at 500hz for every coil, or 16000 impedance data points per second including every coil acquired. The coverage while moving along a line would obtain a resolution of approximately $50/mum$, however this varies due to changing rates of the robot tool orientation information. It was selected to have a shift resolution of $0.25mm$ between lines. These parameters are sufficient to pick up corrosions with similar resolution requirements in area, and $10\mu m$ in depth. Full and fast scanning procedures were used per dataset as discussed previously, in which figures for full scanning shown will only be for coil 16 for simplicity.

A VHX digital microscope from Keyence was used to obtain high accuracy for small defects as a ground truth. This is different from the laser, which is used for understanding defects within the robot's frame and validation of the structured light reconstruction and registration. The digital microscope cannot serve the same function, due to limitation of space on the microscope and it not being with respect to the robot's workspace. Being able to use precise depth reading from the digital microscope to obtain fitting curves is especially useful, especially considering the sizes of the defects involved. What needs to be considered for the microscope's usage is that it will obtain the surface profile, which also includes surface rust. Rust is barely visible to the ECA probe versus steel but will be counted the same as steel. This means to improve the microscope results; the samples must be removed of corrosion, or considered as "cleaned," then rescanned with the assumption that the rust will not significantly change between the uncleaned and cleaned samples.

Subsurface is possible with an elemental analyzer, which would provide composition between the steel, rust, and residue salt or other contaminates, but this is technically destructive testing as it will burn a hole into the sample to obtain this information and not considered useful for surface depth detection.

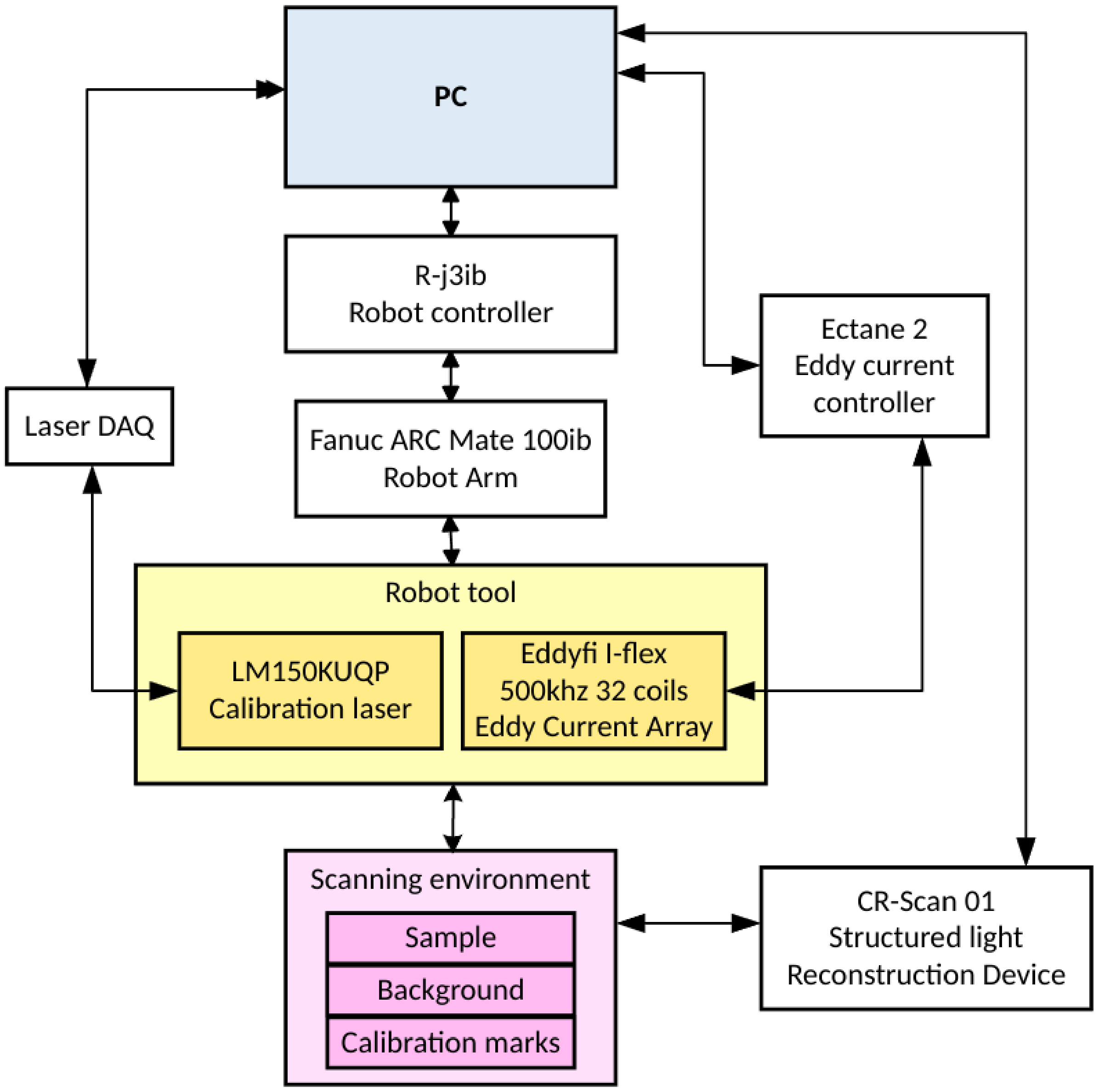


Figure 7.1 ECA robotic scanning diagram.

7.1 Curved Steel Corrosion Detection

For the corrosion detection, the ECA scanning system shown in figure 7.1 is used. Nine samples were made using steel sheets that were corroded for seven days each. 27 defect areas were examined in total, which were measured using a VHX-7000 digital microscope. Each was scanned using the procedures mentioned above.

7.1.1 Sample Preparation and Defect Selection

Nine samples in total were prepared for ECA scanning: 1 flat sample, 4 increasingly concave samples, and 4 increasingly convex samples. Each sample is $12 \times 4in^2$ or $304.8 \times 101.6in^2$ in area, with a depth of 0.75mm. Each sample is $12 \times 4in^2$ or $304.8 \times 101.6in^2$ in area, with a depth of 0.75mm. The reconstruction and path planner for the robots need to compensate for the angular changes of each, demonstrating the complex movements of the probe with curved surfaces along several different curved intensities. There is also a need to see what complications may arise due to poor orientation, specifically for bad rotation alignments. The standard used is ASTM B117 [11] using a salt fog chamber. The samples would be within the fog chamber during a 1-hour operation, which was conducted daily for seven days. The brine solution consisted if deionized water and sodium chloride (NaCl) using the ratio:

$$\frac{Mass\,of\,NaCl}{Mass\,of\,Water} = 0.053$$

The samples in the corrosion chamber are shown in figure 7.4. The back portion of the samples and the bottom inch were coated with corrosion inhibitor during these days. The back was coated to prevent leaching of the back to the front corrosion areas, which would potentially make results more complex than desired. the bottom inch was coated to provide an unaffected background for each sample. Initially, the scans were scanned each day for corrosion growth, however these results were corrupted by issues from heating. Importantly. There was also slight scratching of the corrosion inhibitor on the calibration area for the bottom inch of the samples. This allowed some creeping of corrosion to occur on these components that can be seen in the ECA data. Despite this, most of the calibration regions were unaffected and still allowed for background reading for each. Post processing techniques such as detrending end up obscuring this region, as will be seen later. The samples were placed upwards in the salt chamber. Each day, the samples were removed and cleaned of salt. After seven days, the samples were soaked using a rust remover solution for

1 day. This was done so when the VHX was used to obtain depths that rust would not obstruct the depths that would be picked up from the ECA. The ECA will pick up the differences between the conductive steel surface and nonconductive air. Rust is nonconductive with negligible effects. However, these would be picked up as significant from the VHX, which is not desired. After the samples were soaked, the corrosion was cleaned off using rags.

Prior to corroding, each were curved using a sheet roller at desired curvatures. These curvatures were required to be within the ASTM B117 standards of $10 - 30°$ at any location on the curved sheet. The sheet roller process was manually done on previous samples, then recorded using circular segment calculations. The metric used will be in curvature $k = 1/R$ in units of m^{-1} with R being the radius of the "circle" which the curved sheet is placed along. These curvatures are shown in table 7.1. The flat and concave samples after 7 days have "speckled" defects across the sample with unique patterns throughout. The convex samples removed large components as dripping from the top occurred during these scans, leaving large lines draining along the sample. Each finished sample is shown in figure 7.5, 7.6, and 7.7. Note that from rust removal that some regions were stained darker than usual but have no effect on the ECA or VHX results.

The importance of each sample's curvature is to ensure that it does not go over the B117 requirements of tilt for the samples, between 10 and 30 degrees. To ensure this, each curvature of the sample much be within this range. To simplify the problem, the total curvature would remain constant, following the path of a circle. This was done by recording the chord length C between the edges of the curved sheet, and the height of the arched length M [179] shown in figure 7.2. It should be noted that in relation to arch length S, $C \approx S = 12in$. Using C and M, a solution for the radius R of the circle in which the sheet metal is curved along using the equation:

$$R = \frac{C^2}{8M}$$

With this radius and the chord C may be used to solve the curvature θ of the sample:

$$\theta = 2 * sin^{-1}(\frac{C}{2 * R})$$

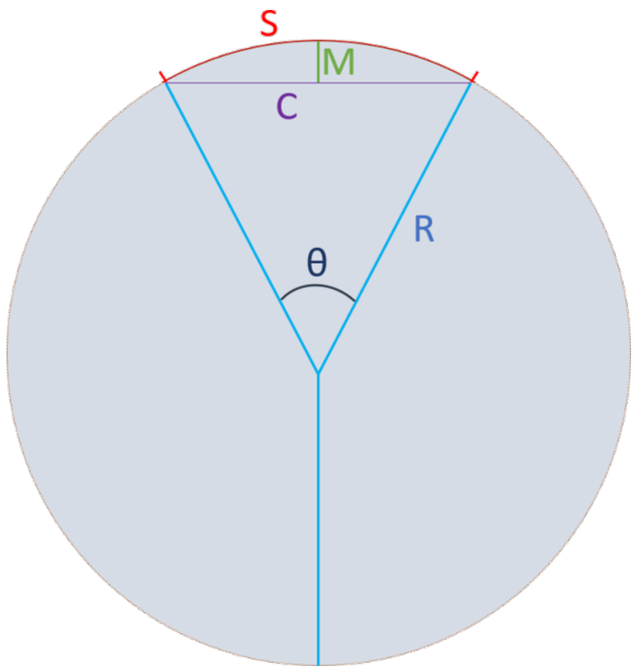


Figure 7.2 Chord-angle model for sample curvature preparation.

A duty sheet metal roller was used to curve the samples, however as this was entirely mechanical, it was difficult to obtain an exact rotation without trial and error. The corrosion salt spray chamber places a flat-shaped sample at a 20 degree angle, in the middle of the 10 to 30 degree range of sample tilt. Figure 7.3 shows the configurations for flat, convex, and concave setups. The sample holder evaluates about 10.3cm diagonally from the bottom. Regions along the 12in length of the sample were partitioned into two sections: d_{12} and d_{23}, with d1 being at the top of the sample, d2 being at the fulcrum resting point of the sample, and d3 at the bottom. These points are measured using the arch length S prior to bending, though because $S \approx C$, each point distance is approximated to be the same between the arch and chord. It is desired to obtain the maximum and minimum amount of curvature that a sample may have while ensuring each piece of rotation of the sheet metal falls within 10 and 30 degrees. This is done by partitioning the angles with the previous, one at the top at θ_{t1}, at the fulcrum θ_{t2}, and at the bottom θ_{t3}, all tangential to the sample. In the flat example, each tilt will be same ideally at $\theta_{t1} = \theta_{t2} = \theta_{t3} = 20$. In the concave configuration, let us assume the sub rotation at θ_{t3} is 10 degrees and the tilt at θ_{t2} is 20 degrees. The known distance $S = 12in \approx 304.8mm$ and the distance $d_{12} \approx 103mm$, so $d_{23} \approx 201.8$. The ratio between d_{12} and

$S = d_{13}$ is 0.338, and the ratio between d_{12} and $S = d_{13}$ is 0.662. Solving for the total curvature θ:

$$Abs(\theta_3 - \theta_2) = 20 - 10^\circ = 10^\circ = 0.6621\theta$$

$$\theta \approx 15.103^\circ$$

Now for θ_{t1}:

$$\theta_{t12} \approx 0.338\theta = 5.103^\circ$$

$$\theta_{t1} = \theta_{t2} + \theta_{t12} = 20 + 5.103^\circ = 25.103^\circ$$

As $10 < theta_{t1} < 30$ degrees and $10 < theta_{t3} < 30$ degrees, the convex configuration is sound at a maximum of 15.103 degrees curvature. The minimum amount of curvature would be at 0 degrees. Similarly, the convex configuration follows the same procedure, with $\theta_3 = 30$ instead with the same results. Likewise, the tilt at $\theta_2 = 20$, the ratios between d_{12} and $S = d_{13}$ is 0.338, and the ratio between d_{12} and $S = d_{13}$ is 0.662. Solving for the total curvature θ:

$$Abs(\theta_3 - \theta_2) = 30 - 20^\circ = 10^\circ = 0.6621\theta = 10^\circ = 0.6621\theta$$

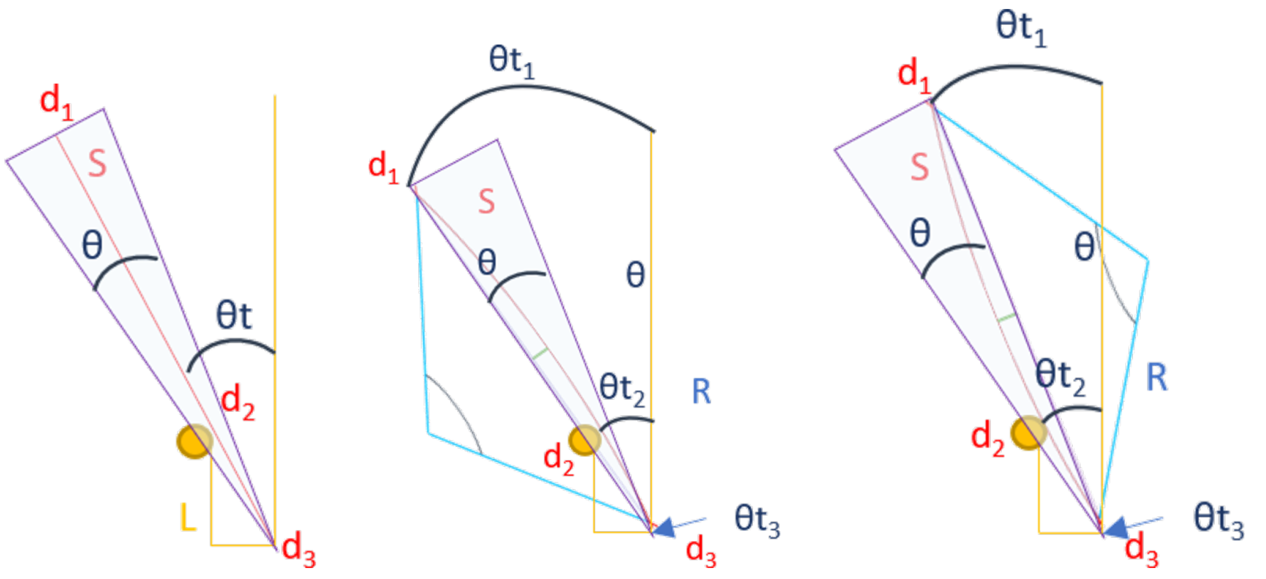
$$\theta \approx 15.103^\circ$$

Now for θ_{t1}:

$$\theta_{t12} \approx 0.338\theta = 5.103^\circ$$

$$\theta_{t1} = \theta_{t2} + \theta_{t12} = 20 - 5.103^\circ = 14.897^\circ$$

As $10 < theta_{t1} < 30$ degrees and $10 < theta_{t3} < 30$ degrees, the convex configuration is sound at a maximum of 15.103 degrees curvature. Both convex and concave configurations are symmetric towards each other, meaning that when creating the curvature on the sheet metals, it is preferred to use the same settings on the duty roller. Unfortunately, despite using the same settings, there was still some variation between sheet metals using the same duty roller configuration, however all measurements were deliberately under 15 degrees to ensure B117 standards were met when corroding the shaped pieces.



(a) Flat (b) Convex (c) Concave

Figure 7.3 Sample curvature diagrams with respect to placement inside the salt chamber.

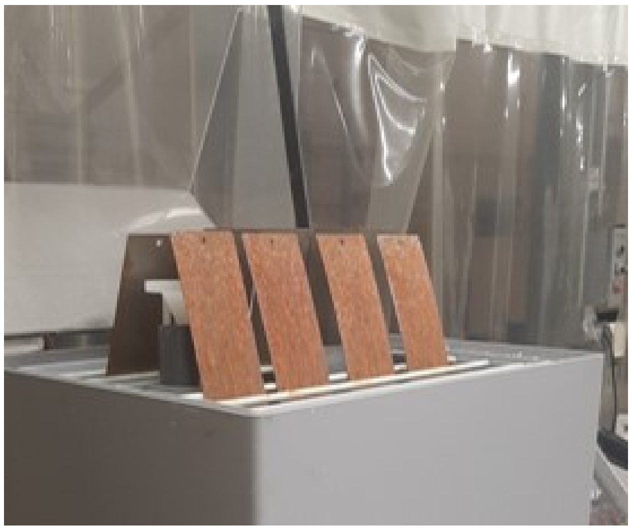


Figure 7.4 Corroded samples inside the salt spray chamber before the 7th day of corrosion.

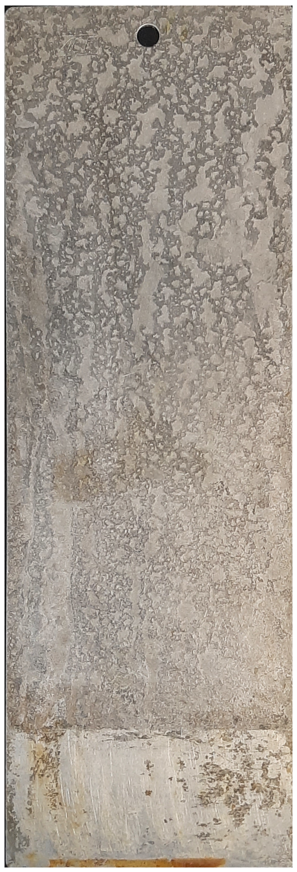
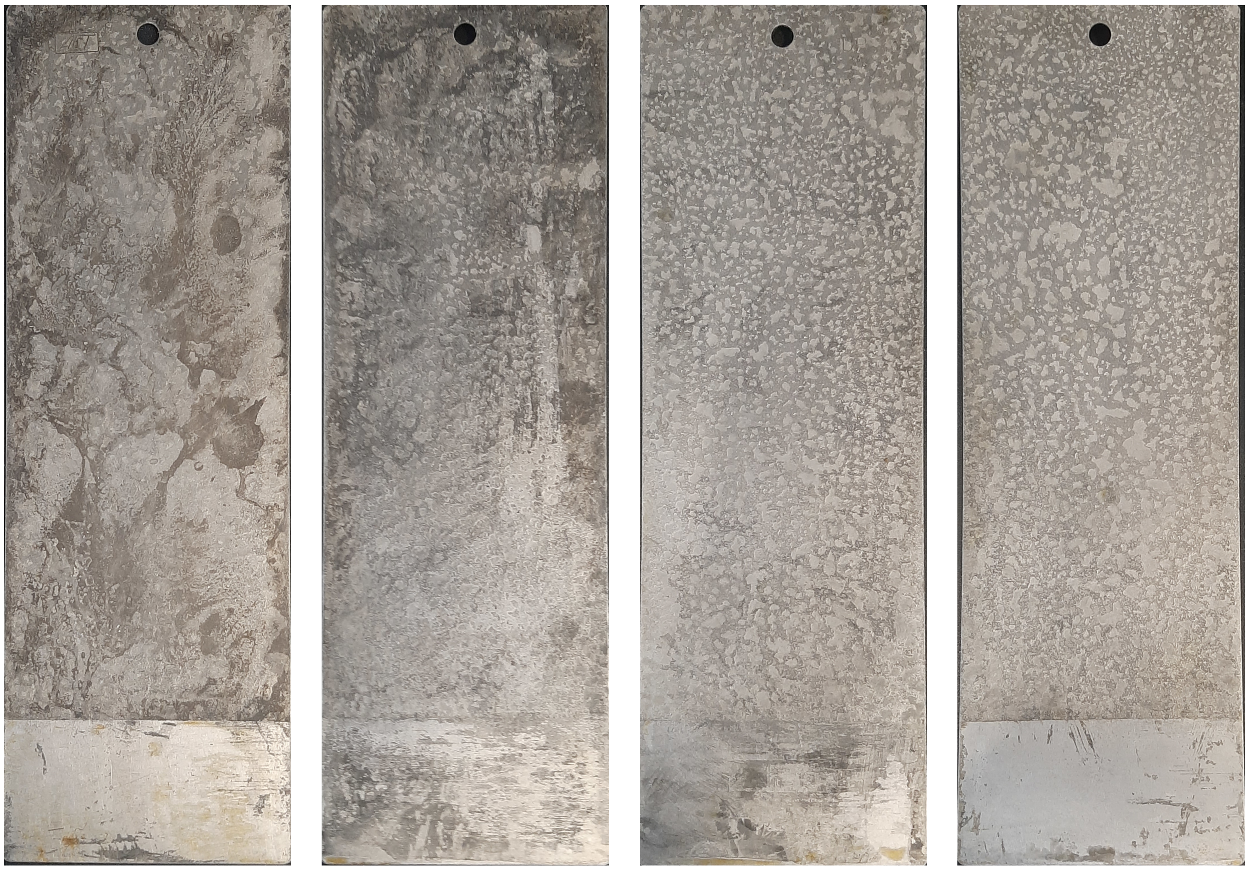


Figure 7.5 Flat sample a.

Sample	a	b1	b2	b3	b4	c1	c2	c3	c4
Curvature k	0.000	0.304	0.397	0.527	0.842	0.304	0.451	0.528	0.842

Table 7.1 Curvatures of each sample under corrosion.

To apply accurate cross correlation, defects were required to be unique across an approximate $2 \times 2cm^2$ area per defect, as limited by the VHX microscope. These defects may be unique based on the pattern of corrosions in the area, or one large but uniquely shaped pattern in terms of XY. Each defect was manually selected with this in mind, with blind knowledge of the ECA results. As a result, some defects end up being outside the bounds of the scans for the full scans for the coils. These results are removed from the results. The VHX results for each corrosion will be down in section 7.1.4 alongside the ECA results. Four defects (D1-4) were chosen on the flat sample, while three defects (D1-3) were chosen for every other sample for concave and convex sets. A larger number of defects would be preferred but would be very taxing on time due to requiring VHX



(a) b1 (b) b2 (c) b3 (d) b4

Figure 7.6 Concave samples b1-4.

scans that take over 20 minutes a piece to obtain, as well as over 20 minutes for processing each. Figures 7.8 show the defect locations for each "critical" sample, or the flat sample and the max curved samples for concave and convex sets, with the defects for each shown later.

7.1.2 Applying the Framework

Reconstruction and registration are used to localize each sample's geometry with respect to the robot's frame. The results from this process are placed into the path generator using the ray-triangle intersection array algorithm discussed in section 5.2. This path is then placed into RoboDK to generate the movements for the physical robot through a simulated environment. The movements are then run while obtaining ECA and robotic toolframe information, discussed in the later sections. This section will discuss and show the mesh and orientation results for each sample. Later, how to orient the data later used in post processing is also discussed.

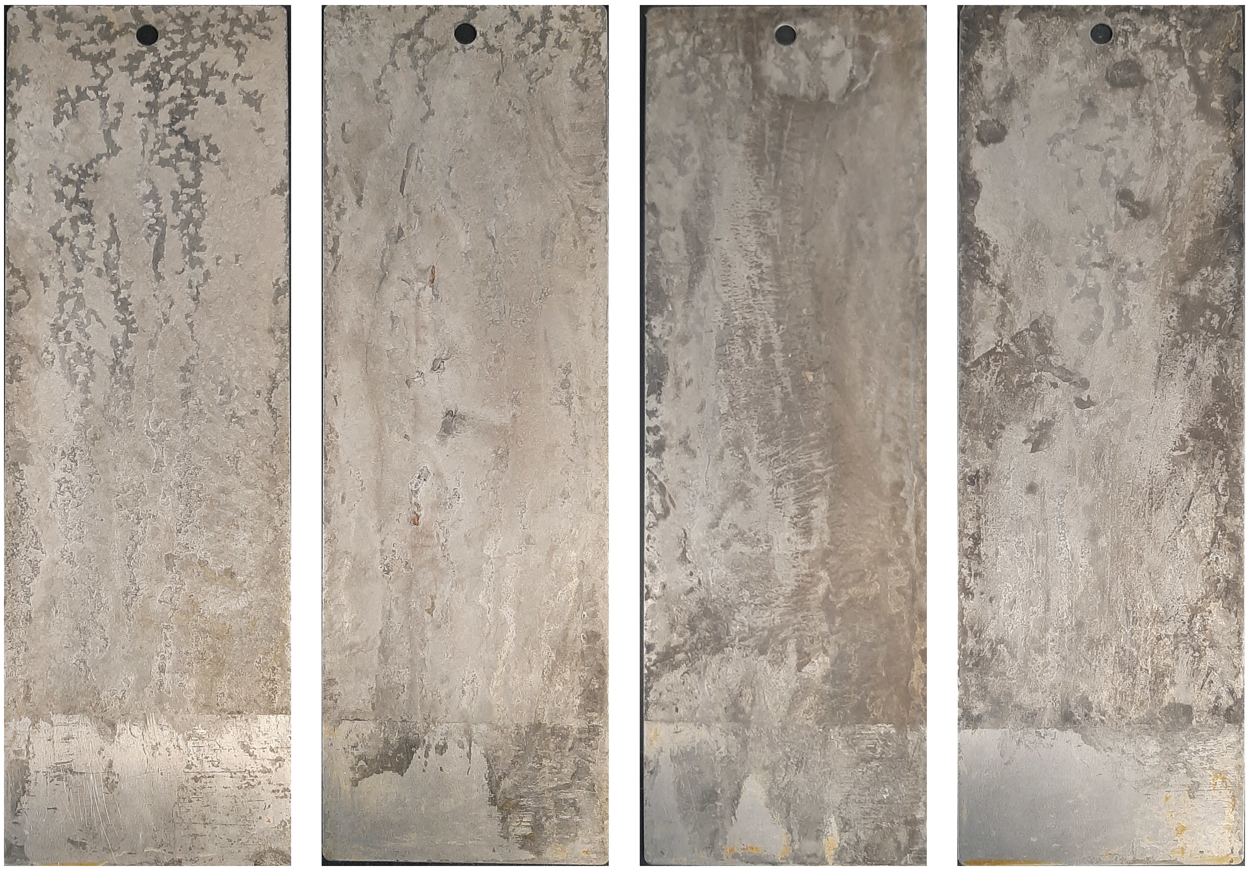
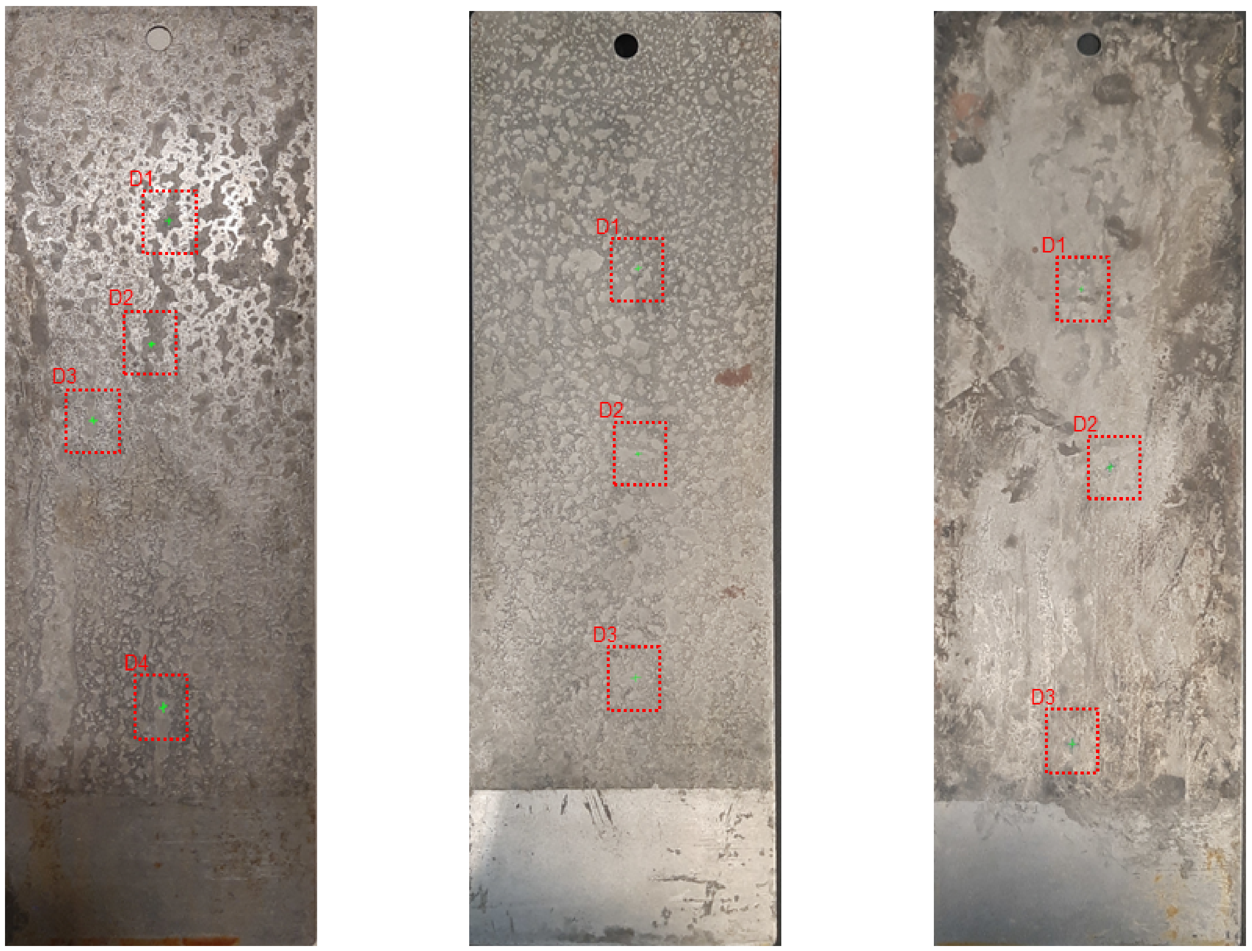

(a) c1 (b) c2 (c) c3 (d) c4

Figure 7.7 Concave samples c1-4.

Prior to scanning, the calibration targets were measured using the laser profiler on the robot arm. This was only required once, as the calibration targets will remain in static locations. A Creality CR-01 structured light camera was placed on a stand that would acquire the reconstruction of the sample and environment for every scan. The CR-01 can obtain reconstructions with 0.1mm depth accuracy within a 536 x 378 mm area. This would be done for each sample for each day, with the reconstruction result being used for both the laser scan and ECA scan. For the earlier scans, the reflection of the laboratory's overhead lights would cause glares on the structured light readout, which would lead to holes in the reflection. To combat this, a large piece of foam was placed strategically to remove this glare. The reconstructed mesh generated from the CR studio software was exported into CloudCompare, in which point pairs picking was used to align the mesh to the robot's base frame. This required human intervention, both in the initial location of the



(a) Sample a with defects (b) Sample b4 with defects (c) Sample b4 with defects

Figure 7.8 Defects localized for critical samples.

laser physically onto the calibration targets initially and on the virtual mesh. This method allows sub-millimeter accuracy between the two sets, though the inaccuracy due to human elements is unknown. Hence why laser profiling is used, to measure inaccuracies between the structured light aligned set. It is advantageous to ensure that the texture mapping of the mesh shows the proper location of each calibration mark. From testing, it is better to keep the camera in a static location as textures may shift with several merged point clouds, giving an improper location of the calibration mark on the mesh. This is shown in figure 7.10. For larger meshes which the SL camera may be used in a handheld mode, giving a larger reconstruction of the area, either geometric-based point selection or more autonomous methods such as checkerboard patterns or April tags may be used. The samples were taped down to prevent any movement during the scan. The calibration marks

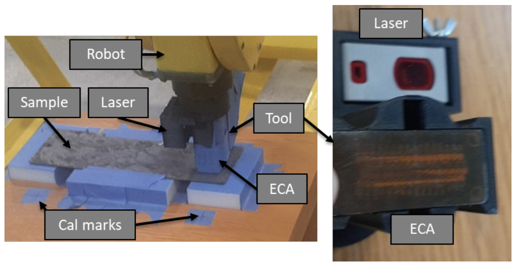


Figure 7.9 ECA tool layout with respect to the scan environment.

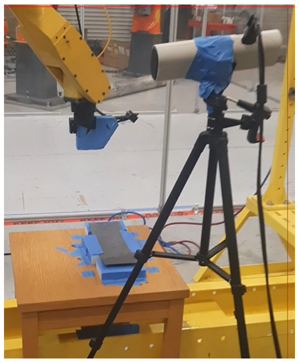


Figure 7.10 Structured light setup with respect to the scan environment.

Point	X (mm)	Y (mm)	Z (mm)
R0	1123.009	148.944	-481.882
R1	1127.309	-104.545	-481.271
R2	937.195	-103.592	-479.255
R0	939.350	148.592	-479.841

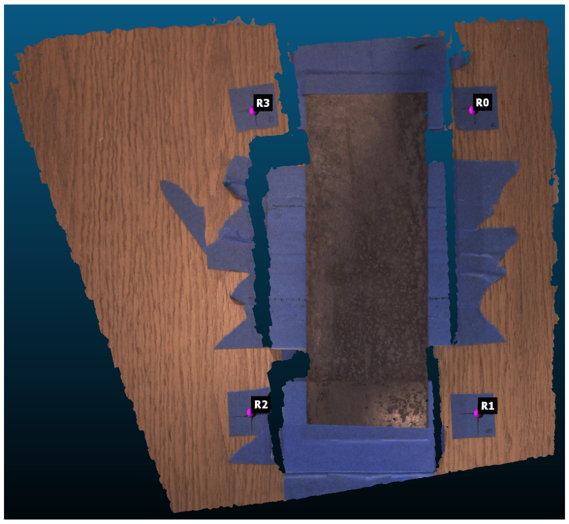
Table 7.2 Recorded physical point positions R0-3 to match with virtual.

used for point-pairs picking (PPP) were also taped onto the scanning environment which were kept in static locations throughout the scan to reduce issues in remeasuring the recorded location with laser to the robot environment. In virtual space, the marks were manually selected, in which table 7.2 shows the information for the physical points, and accuracy in alignment afterwards. The alignment errors with respect from virtual to physical are shown in table 7.3, in which root-means square (RMS) error across the four calibration point-pairs are used. Figure 7.11 shows the meshes and location of each point. Note that the points may drift away from the center of the calibrations due to texturing on the mesh does not fully correlate with the actual position on the mesh.

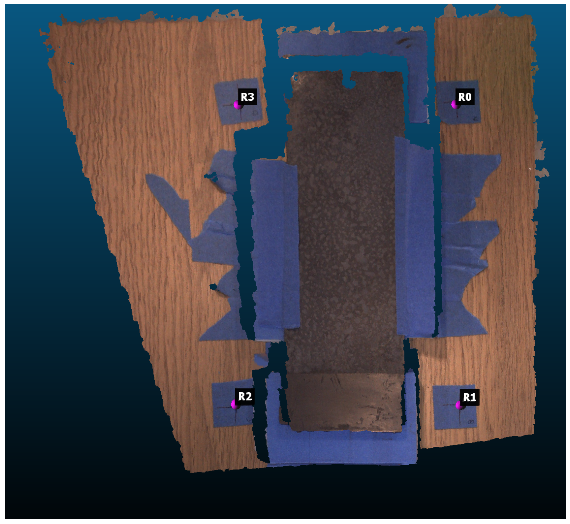
Meshlab is used to post-process the aligned set post point-pairs picking. The operations

Sample	a	b1	b2	b3	b4	c1	c2	c3	c4
PPP RMS (mm)	0.516	0.558	0.770	0.625	0.578	0.722	0.820	0.643	0.587

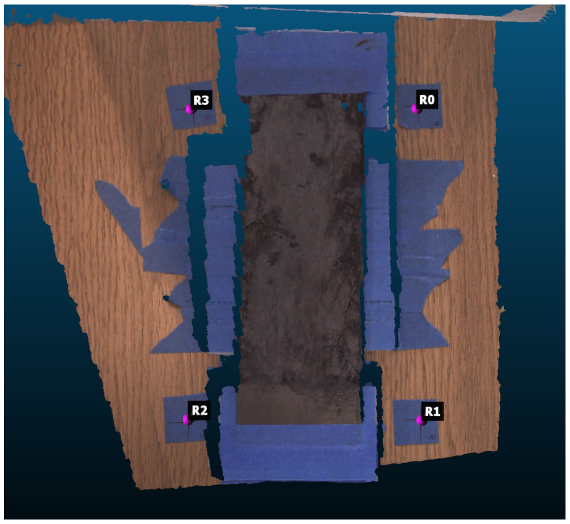
Table 7.3 RMS errors for each sample.



(a) Sample a



(b) Sample b4



(c) Sample c4

Figure 7.11 Meshes post alignment and PPP for critical samples, containing the sample and background.

are: separate the sample from the background (done manually), pre-decimation smoothing via 10 iterations of Laplacian smoothing, quadratic edge decimation to 1000 faces, and post-decimation smoothing via 10 iterations of Laplacian smoothing. The strong amount of processing is used as corroded samples will end up creating a rough surface that is no desired for path planning, so it is heavily smoothed. Decimation is also desired, as through testing, larger face counts will end up causing small "bumps" that may have orientations overreact for rotations which causes problems when scanning. Pre-decimation helps smooth the mesh prior to decimation enables the decimation to place in a smoother location, while post-decimation Laplacian smooths the decimated result which may generate sharp edges. If the mesh generates a hole inside the area of the mesh, then hole repair is used. If an area of the mesh follows a poor rotation, such as if an undesired irreparable hole on the edge is generated, then the effected faces are removed, which is a rare occurrence but is done to prevent damaging the probe or sample. The rectangular area of the sample also tends to become more "oval like" after the smoothing operations, which has been examined to show no effect on the orientation outside of path planning, where intersected arrays may miss the sample on the rounded regions. The cleaned mesh is then sent to the intersection array path planning, which parses the orientations on the cleaned mesh as waypoints. The plane is maintaining constant rotations for each type of scan, though the xy translation on the plane is minorly adjusted to ensure all points are on the mesh. The waypoints are sent to RoboDK, which performs a simulation ensuring that there are no collisions or singularities while scanning. RoboDK then parses a list of robot commands that are sent to the robot for scanning. The path used is deliberately the same between the laser and ECA, with the only difference being the toolframe. This is to ensure intersected coverage of the two scans, while determining any malorientation mostly due to bad rotation.

Each scan obtains both the Fanuc's current orientation and the information of the sensor. Fanuc's PC Development Kit is used alongside its *C#* libraries to obtain the current orientation of the robot around 70 times per second, although this varies and may increase to 100hz or shrink to 30hz. The output is the 3D Cartesian coordinates for translation and rotation. These are saved in a separate file, containing the orientations and time acquired using the acquiring computer's clock. It is more advantageous to use a sync signal in more ideal setups, as there are some inaccuracies using the computer's clock such as desyncs between the robot and the computer, but this was difficult to incorporate. The second set of data per scan comes from the sensor. The eddy current array will output the complex voltages for real and imaginary, as well as the time acquired using the system's clock. The frequency used to obtain each set is at 500hz, with the excitation frequency being set at 500kHz, excitation voltage at 10V, and gain at 60dB. Each coil's excitation is saved, with 32 sets of complex voltages per acquisition. For the laser profiler, only the analog voltage holding depth information time is saved. Each set of data obtained is 1D alongside with time. To equate the Fanuc's orientation and the data of each sensor, interpolation is used from the time information of both sets. This mostly equates between the two sets, however, there tends to be a slight mismatch between the clocks used in the threads of PCDK and the sensor, with a time delay required. This time delay is potentially non-static, but a 50ms delay for the ECA and 70ms for the laser are used. The output of interpolation will be a point cloud with floating orientations holding sensor data. The eddy current array requires an additional translation per coil, which is done by obtaining the local orientation of each coil respective to the toolframe and translating each coil point. As the alignment of the probes used 2 rows of 16 coils each, with the difference between each adjacent coil being consistent between each, the following algorithm is used:

Rxyz is a function which converts the rotation values into a 4x4 rotation matrix. Likewise, Txyz converts translation values into a 4x4 translation matrix. Wristflip is used to flip the orientation to ensure proper placement of each coil, which is determined by the tool orientation. The laser profiler calculates the analog signal to depth. To ensure the ECA data and Laser data is aligned correctly, we run the inverse of the lift-off algorithm to place the data sets onto the location of the sample, rather than floating above it from the lift-off amount. This is done by simply applying the rotation transform per element:

$$OLift = Rxyz(rx(i), ry(i), rz(i)) * Txyz(0, 0, liftOff(i))$$

Then applying the translation of the rotation to the original:

$$ox(c, j) = OLift.X + x(j);$$

$$oy(c, j) = OLift.Y + y(j);$$

$$oz(c, j) = OLift.Z + z(j);$$

Algorithm 7.1 Coil alignment for a 2 row array

Input: Lengths between each ECA coil dx, dy, number of coils $numCoil = 32$, Robotic orientation data x, y, z, rx, ry, rz

Output: Aligned coil data ox, oy, oz.

```
1: $ox = zeros(numCoil, length(x));$                    ▷ initialize the outputs
2: $oy = zeros(numCoil, length(x));$
3: $oz = zeros(numCoil, length(x));$
4: $wristFlip = Rxyz(0, 0, -90);$   ▷ obtain 4x4 rotation matrix to flip the data to the sample
5: $zigdx = (dx/2);$                        ▷ define zigzag amounts to along x
6: $zagdx = -(dx/2);$
7: $shiftY = ((dy * (numCoil - 1))/2);$                ▷ define shift amount y;
8: $zig = 1$
9: for $c = 1 : c <= numCoil; c++$ do                    ▷ for each coil c
10:     $ddy = -((dy * (c - 1)) - shiftY);$                   ▷ append to shift y
11:     if $zig == 1$ then          ▷ chose x value from zig-zag, then flip zig value
12:         $ddx = zigdx;$
13:         $zig = 0;$
14:     else $zig == 0$
15:         $ddx = zagdx;$
16:         $zig = 1;$
17:     end if
18:     $coilPoint4D = [ddx; ddy; 0; 1];$                ▷ calculate the local point
19:     for $i = 1 : i <= length(x); i++$ do   ▷ for each data point i, adjust the orientation
20:         $adjustCoil = wristFlip * Rxyz(rx(j), ry(j), rz(j)) * coilPoint4D;$
21:         $ox(c, j) = adjustCoil(1) + x(j);$
22:         $oy(c, j) = adjustCoil(2) + y(j);$
23:         $oz(c, j) = adjustCoil(3) + z(j);$
24:     end for
25: end for
26: $returnox, oy, oz$
```

The rotation per point is maintained, this only translates the point along the normal or rotation from original point. After orienting the data onto the original mesh positions, any mismatch between calibrated ECA and laser data may be detected. As the probes used were flex coils, some of the coils were mismatched, however as the coils were set in a rigid formation, this was mostly approximated as a body translation along x and y w.r.t. the toolframe rather than per coil. Simple, the ECA data set was aligned with the laser to obtain the calibration. Any drift along the z axis w.r.t. the tool frame was assumed to be zero, though this may not be the case in practicality.

7.1.3 Processing Procedure

The processing procedure from the reconstruction to path generation goes as such:

1. Obtain reconstruction, use Creality software to output a mesh.
2. Place mesh into CloudCompare for PPP: input physical point locations, then manually select the equivalents on the mesh. If RMS is above 1mm, then redo the point selection with the better oriented mesh.
3. Place the mesh into MeshLab.
4. Cut out the sample from the background.
5. Pre-decimation smoothing from 10 Laplacian smoothing operations.
6. Quadratic edge decimation to 1000 faces.
7. Post-decimation smoothing from 10 Laplacian smoothing operations.
8. Place into ray-triangle intersection algorithm software:
9. For full scans, rotate the array by 90° along the plane's z-axis (normal to the sample), set up an area of $250 \times 50mm^2$ and 1000×2 points along Y and X respectively (X faces away from the robot, Y is left-and-right, Z is up-and-downwards from the ground).
10. For fast scans, do not rotate, set up an area of $250 \times 1mm^2$ and 2×4 points along X and Y respectively.
11. Center the location of the scan for the scan to ensure the scan is centered on the sample.
12. Apply array intersection in a zig-zag pattern using algorithm 5.2.
13. Send the generated path from intersections to RoboDK, which will simulate the scan.
14. Send the path generated from RoboDK to the physical robot for scanning.

Run the scan, which will place the information of the ECA, including system timing information, asynchronously with the robot's toolframe using Fanuc PCDK, and its processes system timing information. After the scan, both sets of information are placed into MATLAB for post-processing. First, for ECA 3D+NDE PTC information:

1. Read ECA information for real, imaginary, and timing.
2. Read Robot information for pose and timing.
3. Apply a static 50ms delay to the robot timing info to help match ECA and robot tool info.
4. Set the start times of both to zero.
5. Delete redundant points before interpolation.
6. Interpolate the robot pose data to the faster acquired ECA data using both timing information.
7. Apply a global shift from calibration: 0.988mm along x, and 0.593mm along y.
8. For each coil, apply 1mm lift-off toward the sample to undo lift-off, along the normal determined by rotation per point.
9. Place each coil to the robot frame using algorithm 7.1.
10. Detrend along time and per coil, fast: 2nd order for both, full: 2nd order for 1st, ignore per coil.
11. Apply the array subtraction algorithm using algorithm 6.1.
12. Another similar detrend along time and per coil, fast: 2nd order for both, full: 2nd order for 1st, ignore per coil.

This will process each ECA PTC for both full and fast modes. After this, each defect will be examined with respect to VHX:

1. Read the 2D VHX data, decimate to $0.2 \times 0.2mm^2$ pixel areas.

2. Apply 2D detrending of order 1, then order 2.

3. For each coil:

4. Scatter interpolate the ECA PTC to a 2D image with $0.2 \times 0.2mm^2$ pixel areas and using natural interpolation method.

5. Apply 2D determine of order 1, then order 3.

6. Apply xcor, which checks for the best rotation by the best xcor value output in normalized units.

7. Cut out the ECA data using the area of the VHX set.

8. Obtain the defect and background intersection ratios between ECA and VHX. Iterate to determine the best related voltage to VHX from the best intersection.

Poor data is determined from full scanning if the bounds of a data set are not within 20mm of the radius of the median of the data sets. Poor data for fast scans are compared similarly, but with the median of the full scan data. From these median data, if the coil's is determined to be outside of the scan, 20m away from the radius of the median value, then its information is discarded in performance as it would be determined not to provide information about the defect.

7.1.4 ECA versus VHX Results

Using the framework provided, ECA point clouds are obtained then processed into an output 2D image. This image will be compared with VHX to align the ground truth VHX data with the ECA image. If they are aligned, then the defects and background are intersected to have a performance metric for the ECA. If they do not align, they are considered undetected. The exception is if the defect would not have been in the image, which is checked by comparing the median values of correlation per coil with the expected location. If the defects are outside the bounds, they would not generate any data. These pieces of data are discarded in this scenario. Scanning is both done in fast and full configurations. This section will show the results of each sample. Afterwards, the performance of detection is discussed. For full scans, only coil 16 is shown. For the processes, the resolution for both should be around $0.1 \times 0.25mm$ while using scatter interpolation. The pixel areas after interpolation were set to $0.2 \times 0.2mm$. The area of each full scan should be $50 \times 250mm$, while the area of each fast scan should be approximately $37 \times 250mm$. Note that because an ECA figure for full scans shows poor results does not mean that another coil was unable to pick up the defect instead. For the bulk of data comparing ECA, VHX, xcor, and intersection data for each defect, coil 16 only, for both fast and full scans, see the appendix.

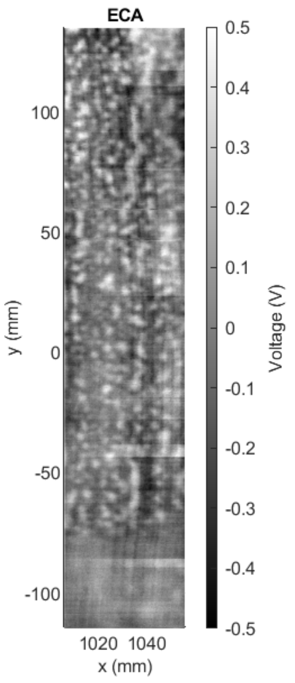
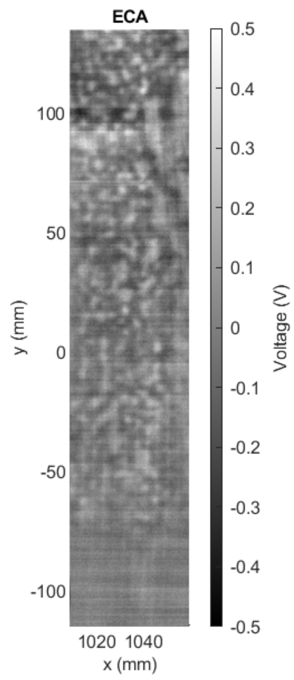
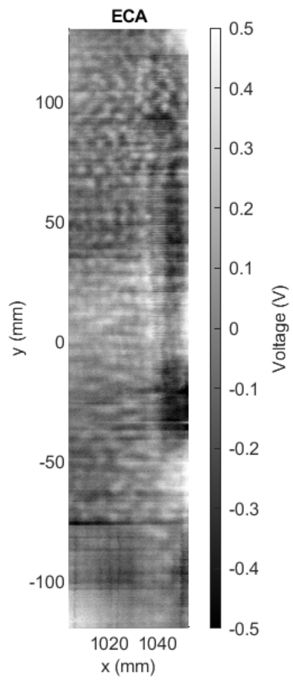


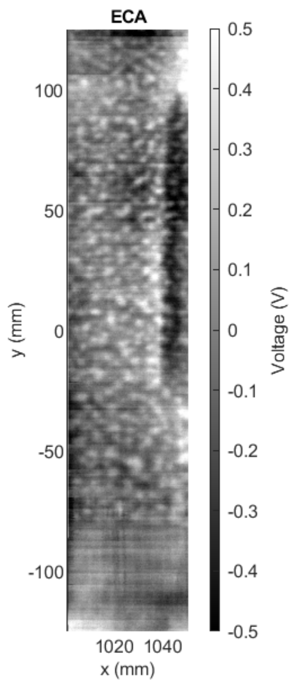
Figure 7.12 Flat sample a full scans.



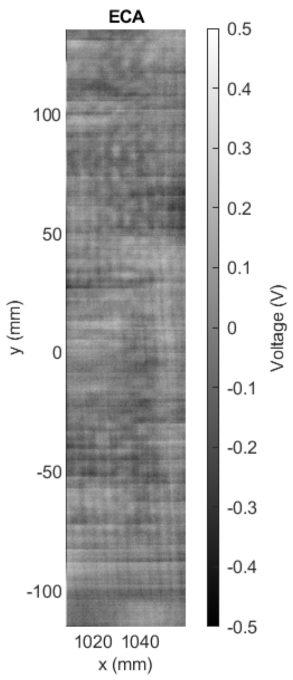
(a) b1



(b) b2

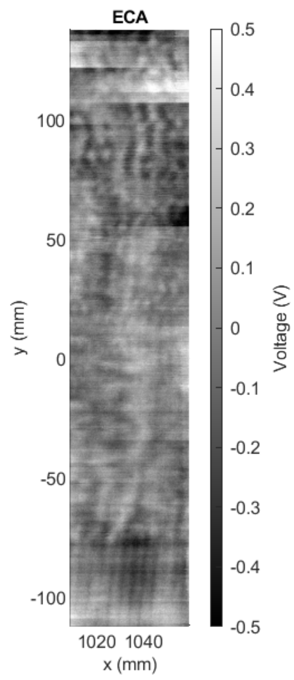


(c) b3

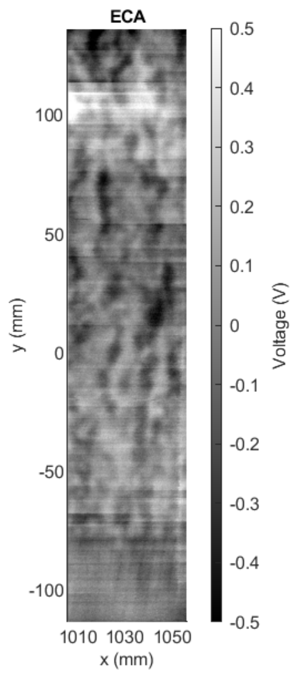


(d) b4

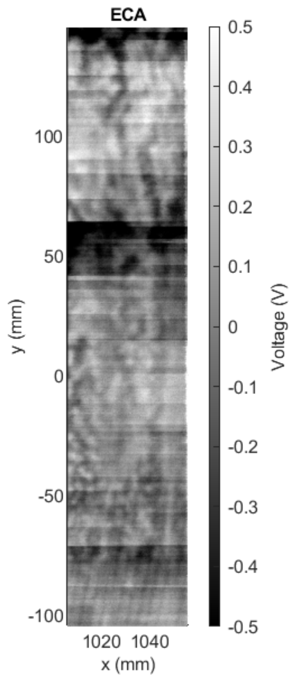
Figure 7.13 Concave samples b1-4 full scans.



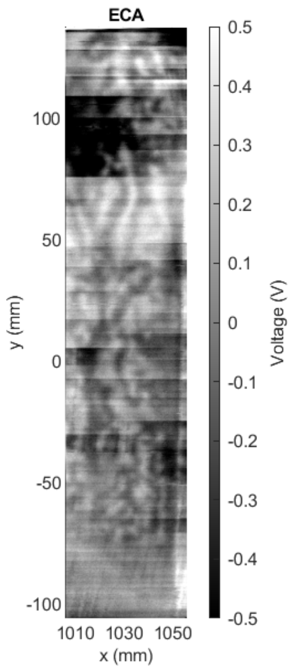
(a) c1



(b) c2



(c) c3



(d) c4

Figure 7.14 Concave samples c1-4 full scans.

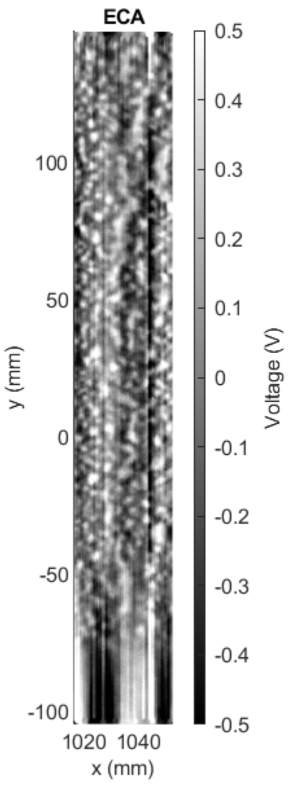
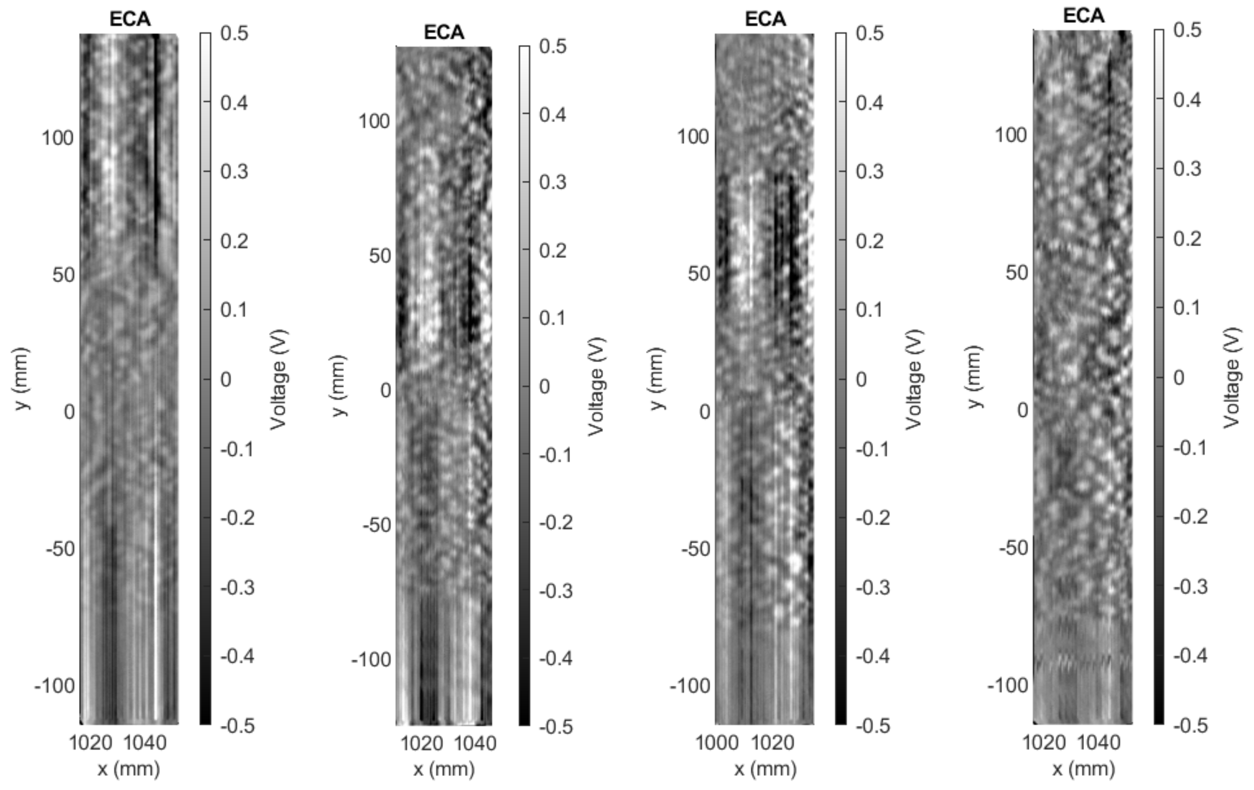
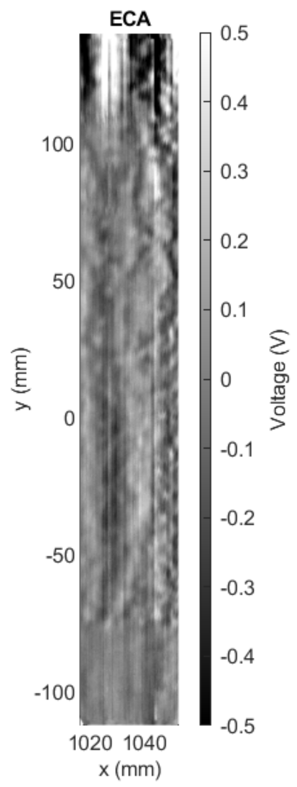


Figure 7.15 Flat sample a fast scans.

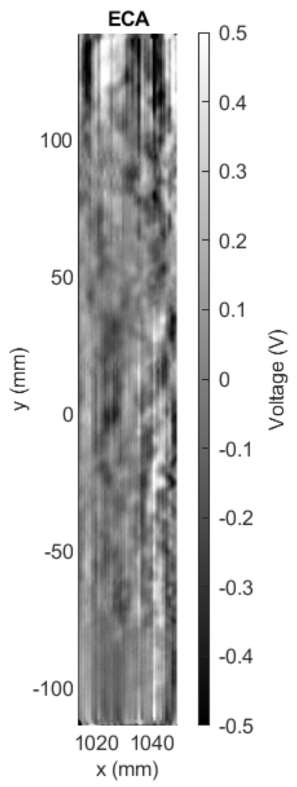


(a) b1 (b) b2 (c) b3 (d) b4

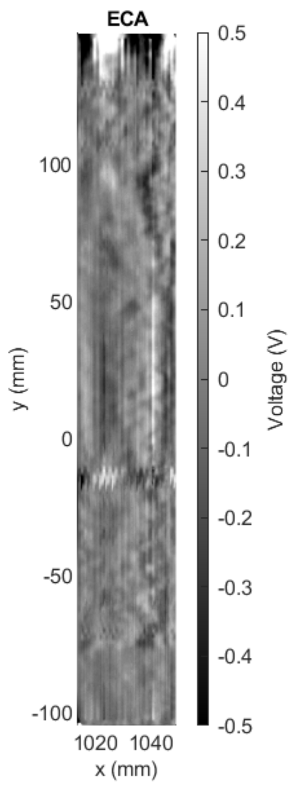
Figure 7.16 Concave samples b1-4 fast scans.



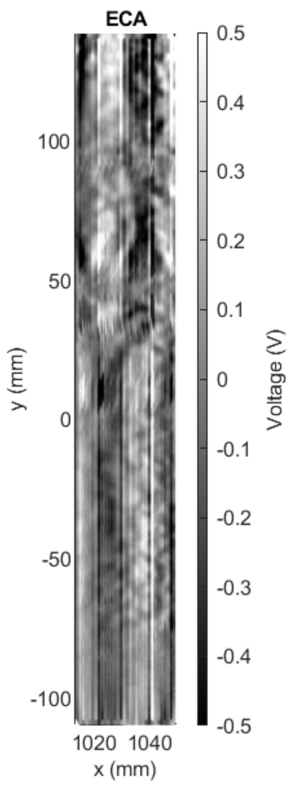
(a) c1



(b) c2



(c) c3

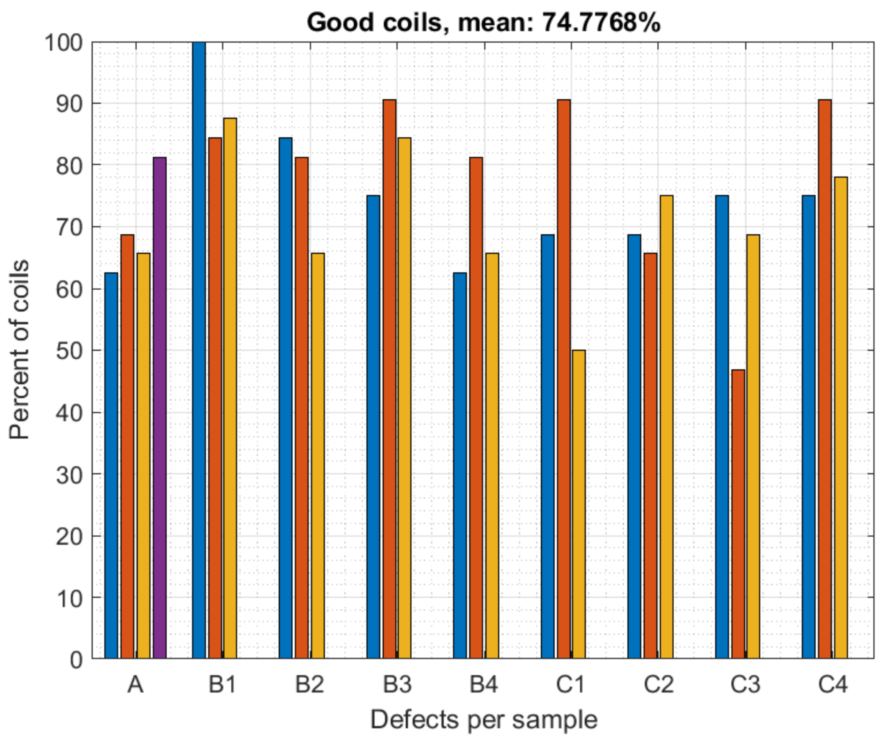


(d) c4

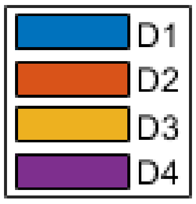
Figure 7.17 Concave samples c1-4 fast scans.

7.1.5 Performance

The performance metrics for both fast and full scans are discussed in this section. For full scans, the percentage of coils with a detection in figure 7.18, the xcor values comparing orientation between ECA and VHX for average and best cases in figures 7.19 and 7.20, and the average and best intersection values are considered in figures 7.21 and 7.22. For fast scans, only the number of detected defects per scan shown in figure 7.23, the xcor values in figure 7.24, and intersection values in figure 7.25 are considered.

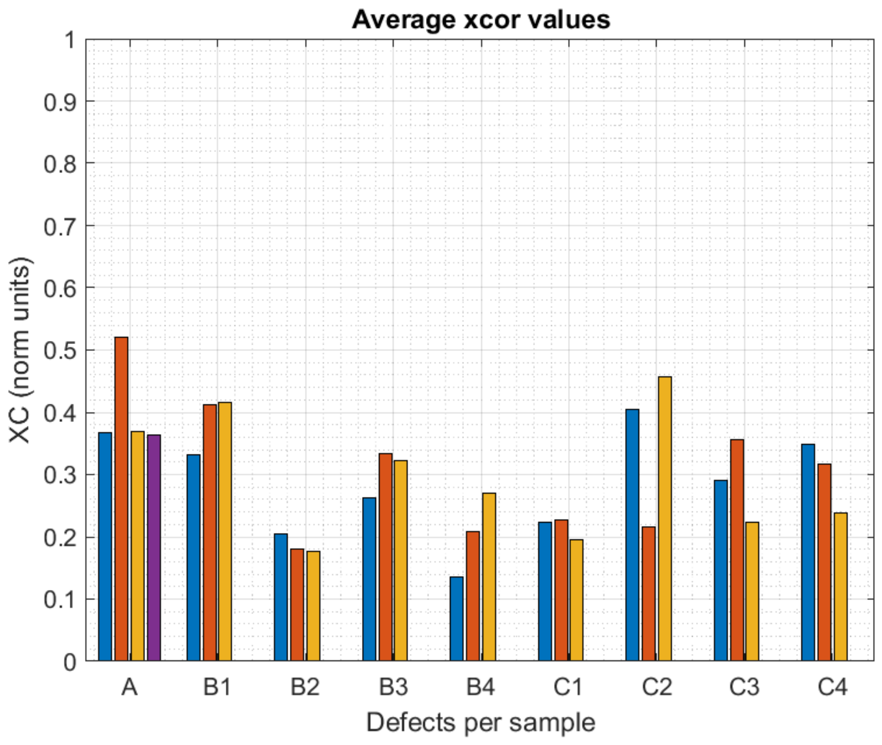


(a) Defects per sample vs coil results

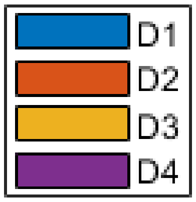


(b) Legend

Figure 7.18 Full scan, percent of coils detected per defect per sample.

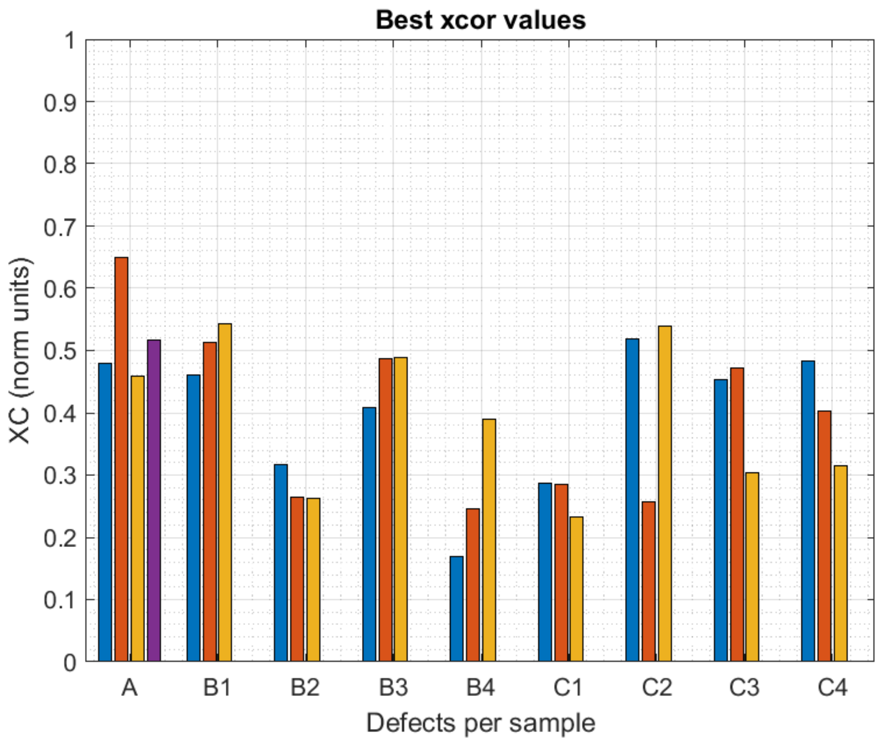


(a)

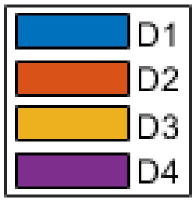


(b) Legend

Figure 7.19 Full scan, average xcor values detected per defect per sample.

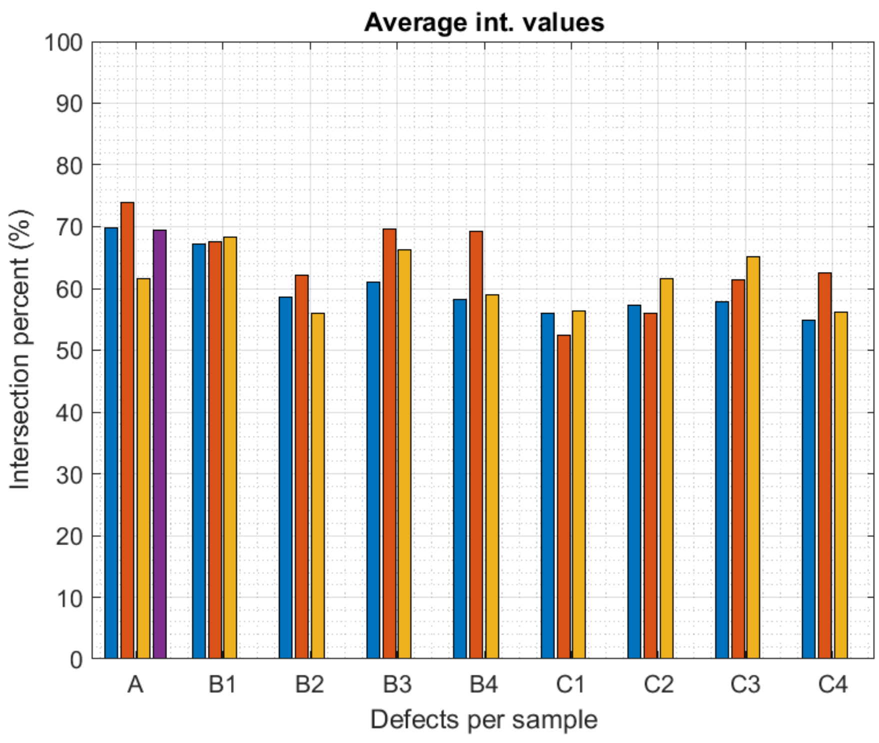


(a) Defects per sample vs coil results

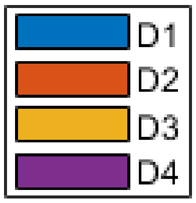


(b) Legend

Figure 7.20 Full scan, best xcor values detected per defect per sample.

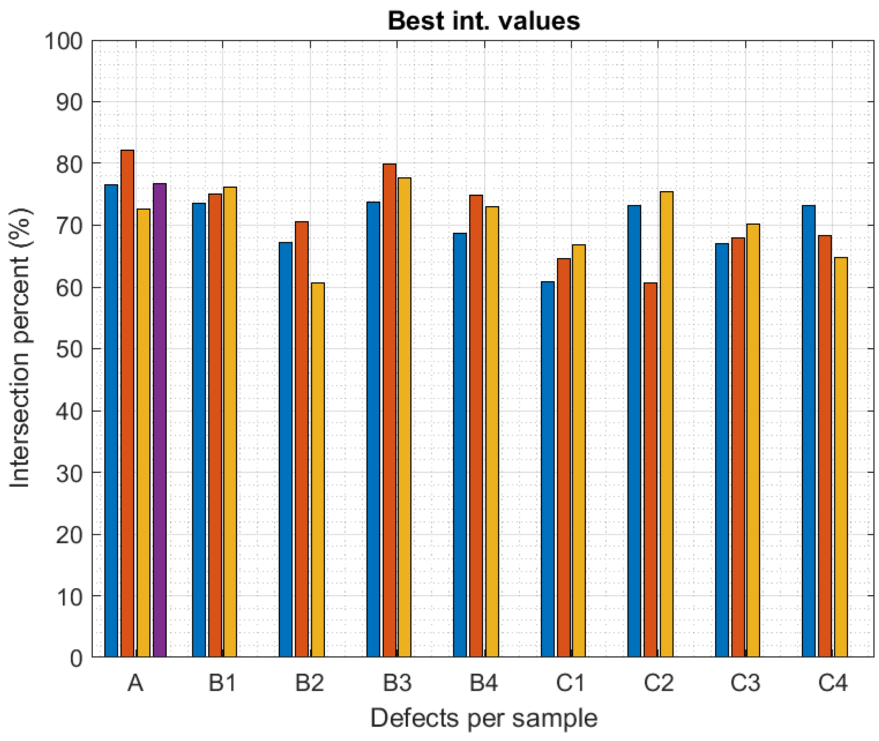


(a) Defects per sample vs coil results

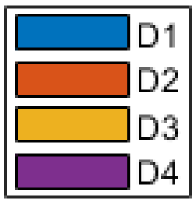


(b) Legend

Figure 7.21 Full scan, average intersection values detected per defect per sample.

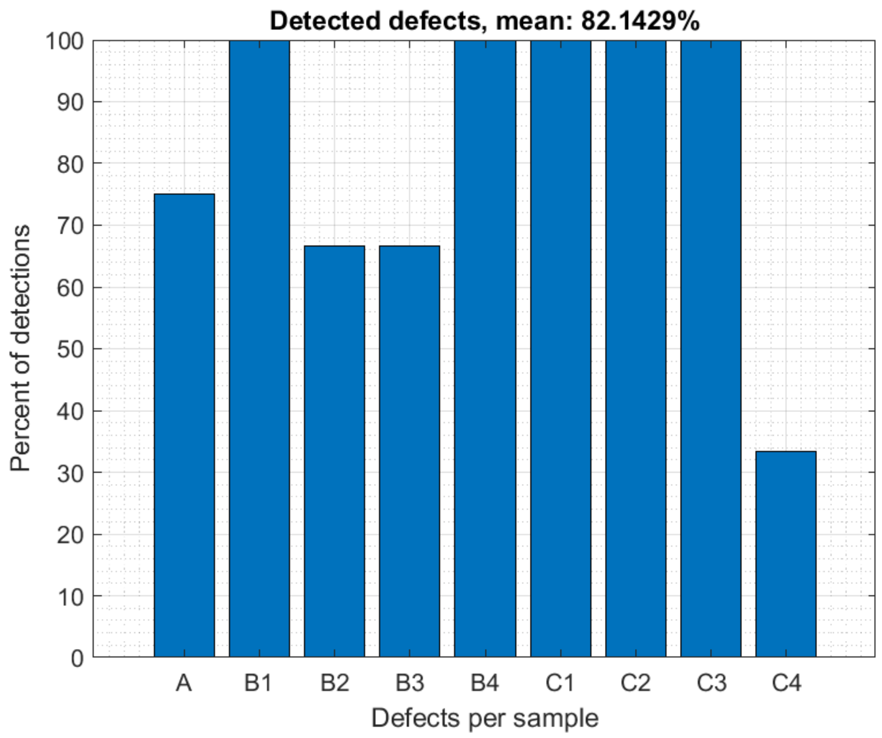


(a) Defects per sample vs coil results



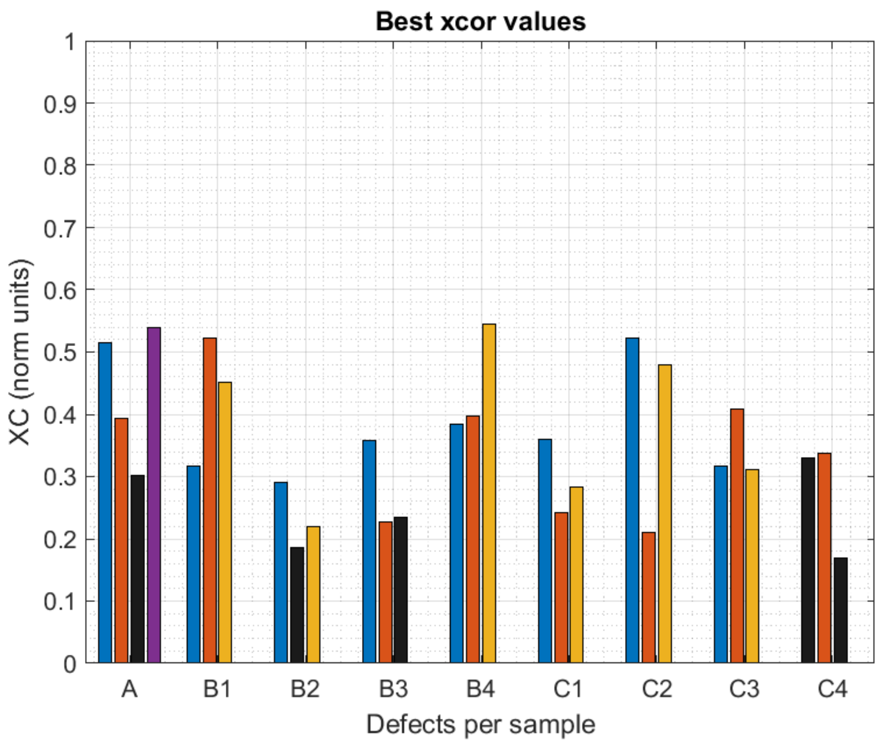
(b) Legend

Figure 7.22 Full scan, best intersection values detected per defect per sample.

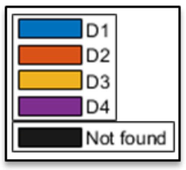


(a) Defects vs detections

Figure 7.23 Fast scan, number of defect detections per sample.

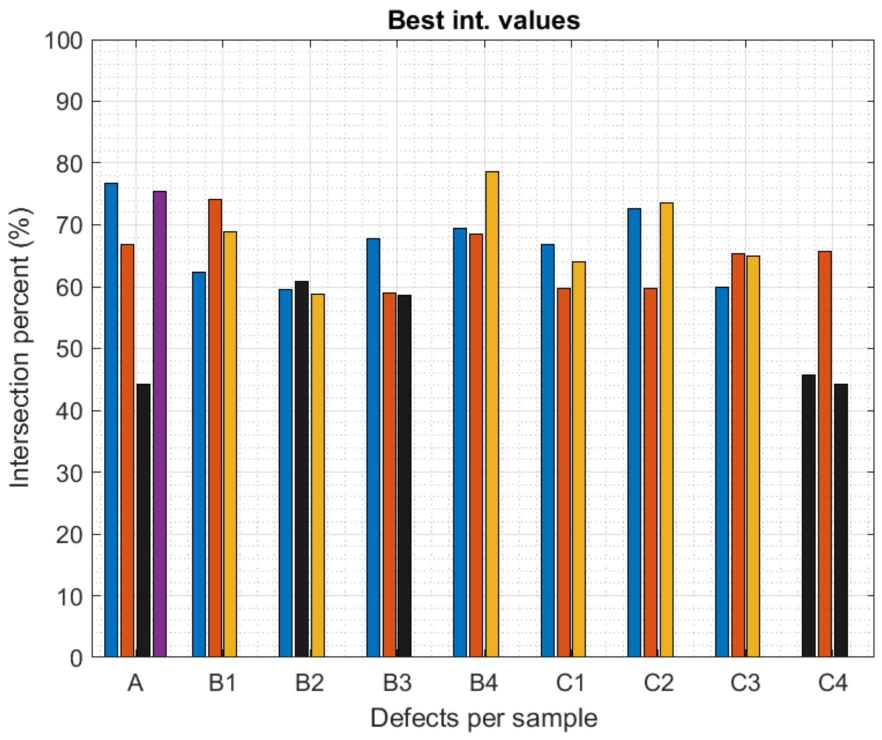


(a) Defects vs detections

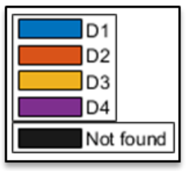


(b) Legend

Figure 7.24 Fast scan, xcor values per sample.



(a) Defects vs detections



(b) Legend

Figure 7.25 Fast scan, intersection values per sample.

7.2 Voltage to Depth

Voltage to depth would be the next process to compare depths calculated from ECA voltage parameters from impedance to the depth values from the VHX sets. However, due to the hefty amount of post processing to remove lift-off and heat variation, with some of the effects still appearing in the data sets, this is unfeasible for this work. This section will briefly show voltage to depth for diverse types of samples still scanned with the robotic system.

The samples, shown in figure 7.26, contain 10 plates corroded for different weeks (indicated by the number on the sample), as well as two calibration samples. To avoid temperature or other gain fluctuations, every sample was scanned at once. The voltage results are shown in figure 7.27 using coil 16 and a full scan. A planar scan was used, with a $1 \times 0.1mm^2$ resolution. The probe was excited at 250khz and 5V. A linear model was used, shown in figure 7.29, and results on the full scan shown in figure 7.28. In comparison, the two calibration samples are compared in figures 7.30 and 7.31. In a later works, the corrosion will be compared with IFM and UT results.

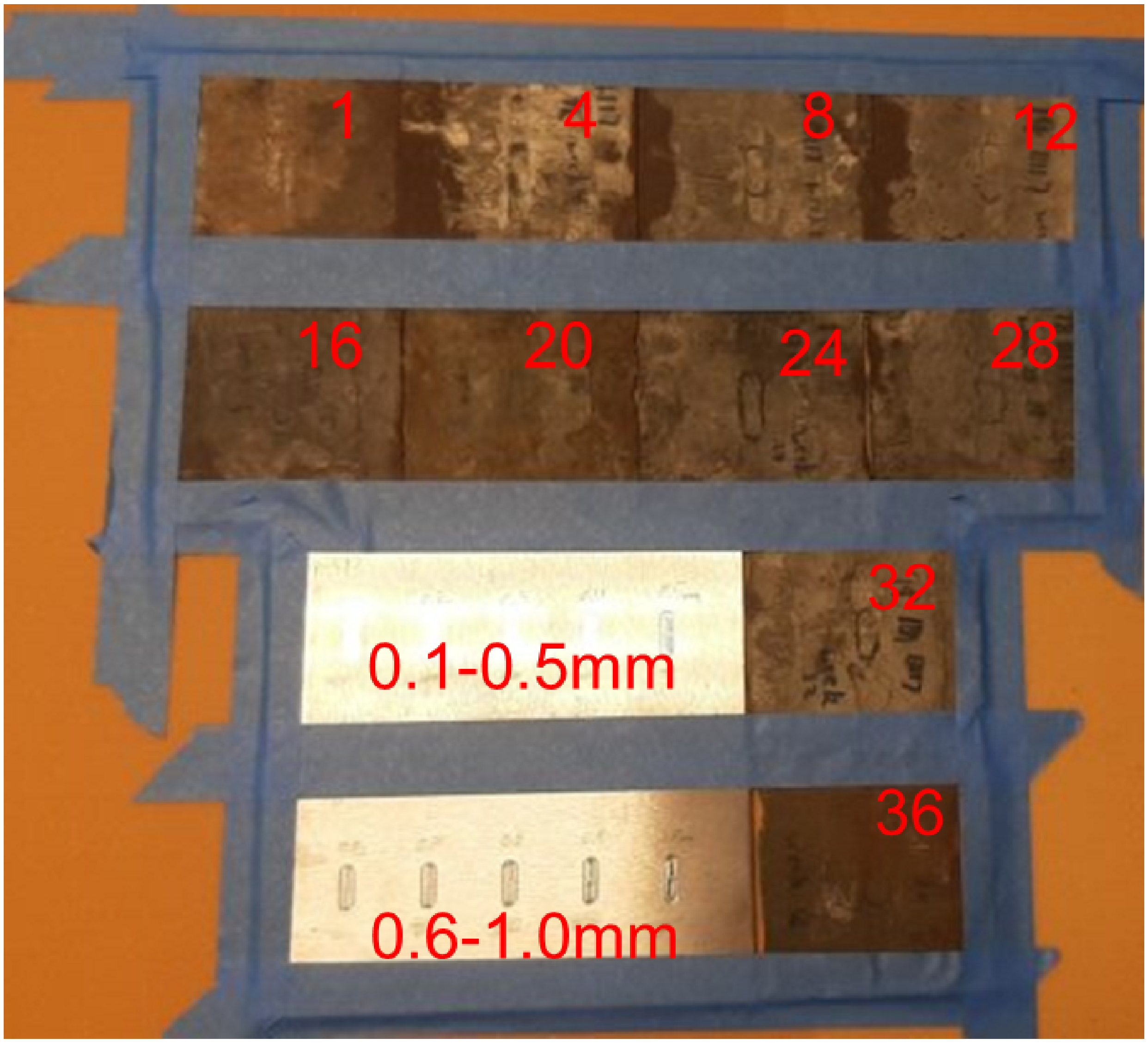


Figure 7.26 Voltage to depth samples.

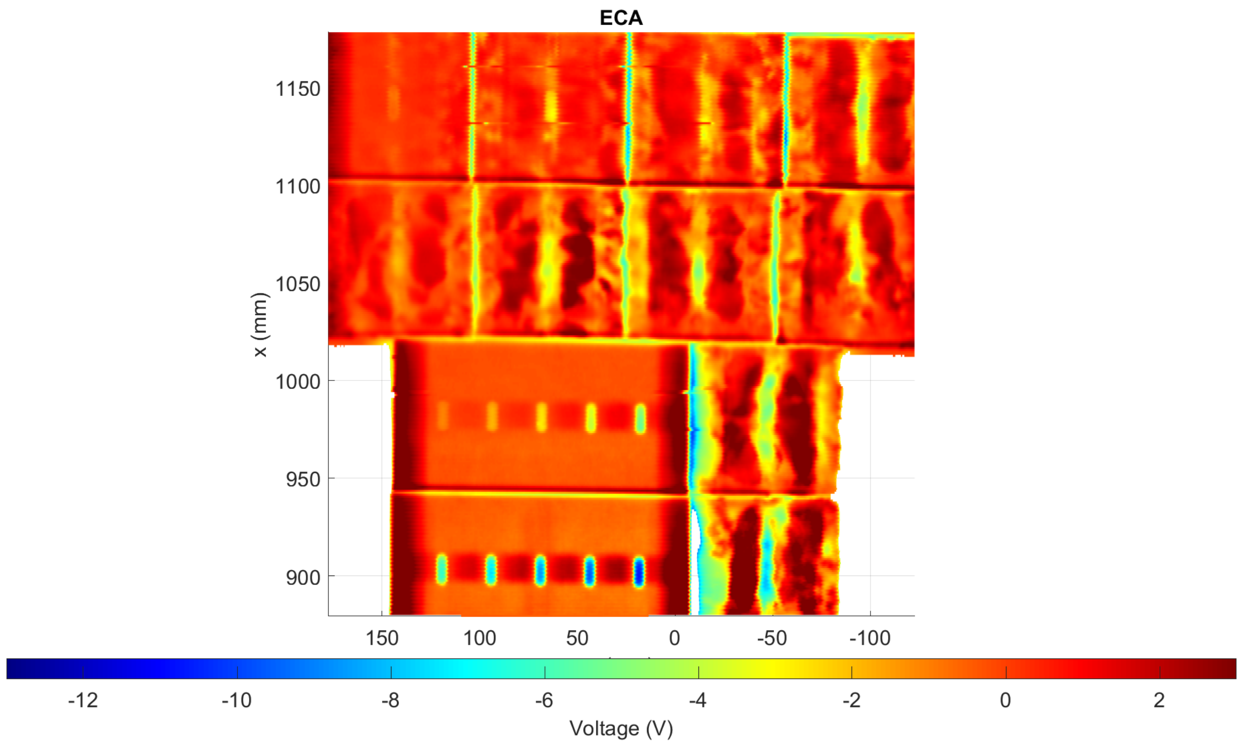


Figure 7.27 Voltage equivalent from the samples scanned.

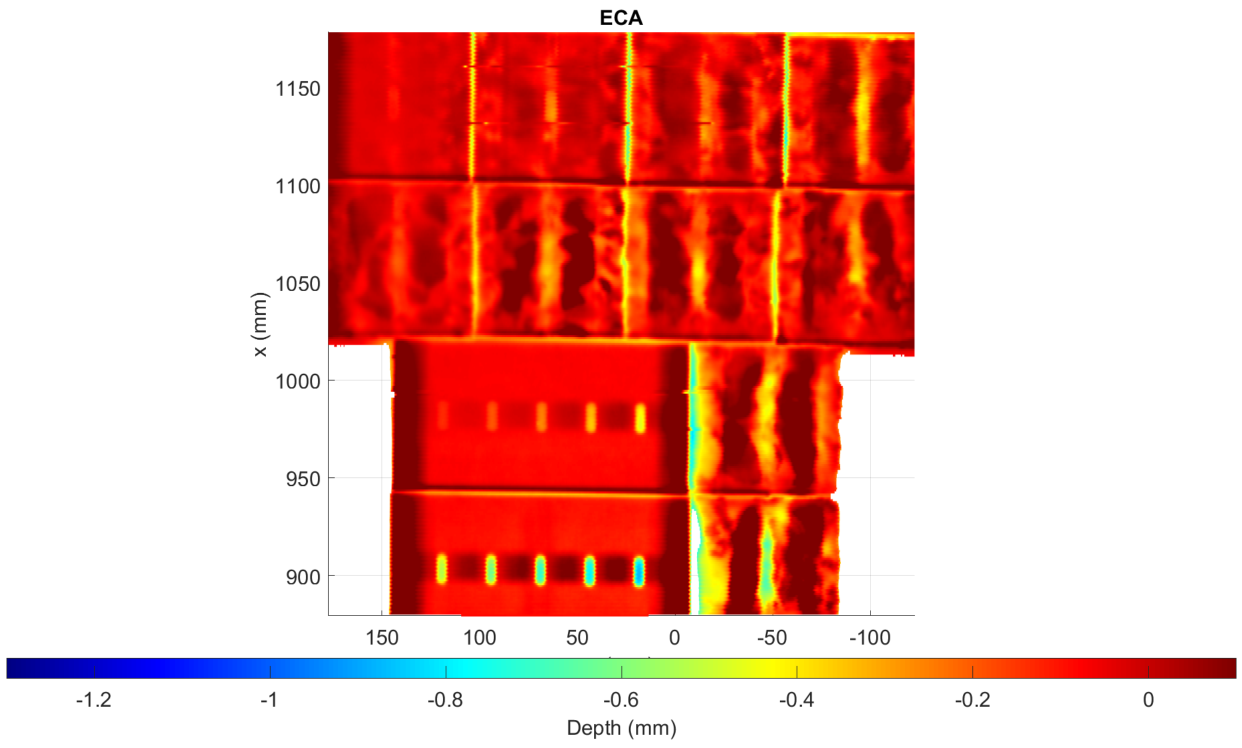


Figure 7.28 Depth equivalent from the samples scanned.

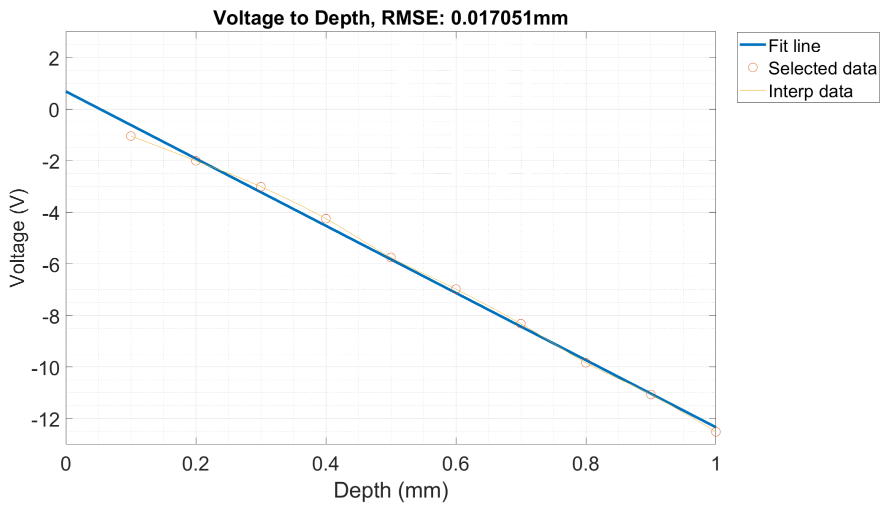


Figure 7.29 Voltage to depth response from the calibration areas, set as linear.

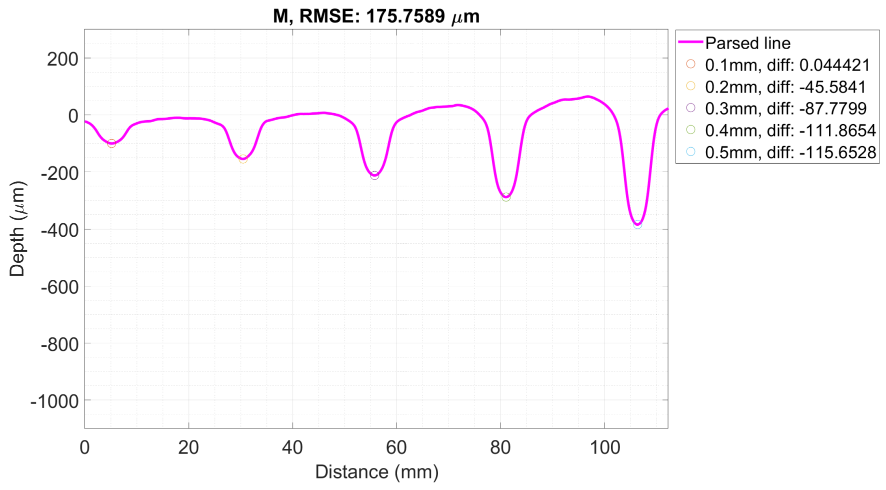


Figure 7.30 Calibration set 1 from 0.1 to 0.5mm.

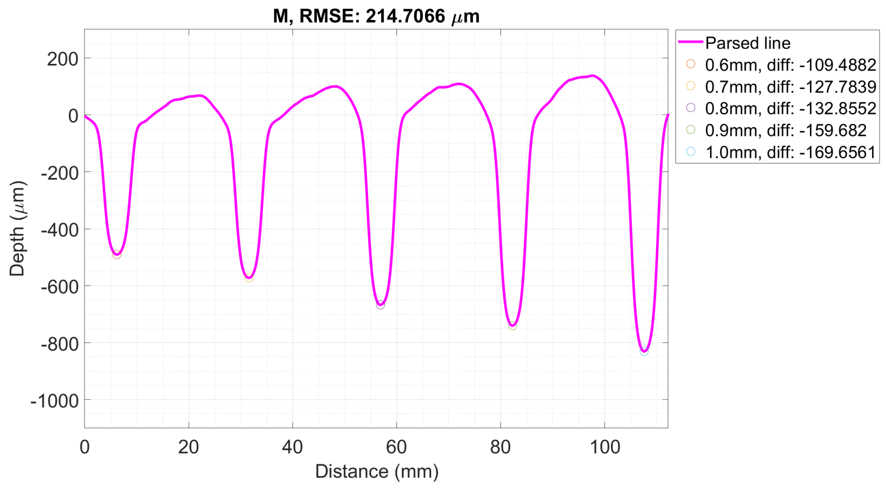


Figure 7.31 Calibration set 2 from 0.6 to 1.0mm.

7.3 Ultrasonic Testing on Carbon Fiber Reinforced Polymer

Ultrasonic testing (UT) was conducted using the same NDE 4.0 framework discussed for ECA, up until the NDE section. This is in support of the "black box" premise for surface scanning, appropriate to the NDE scenario desired. In this case, a complex-shaped carbon fiber reinforced panel (CFRP) was scanned using air-coupled UT probes, obtaining the reflected propagations that would determine the difference between a defect region and the sample's background. For more information about how this scan was conducted, see [75]. Figure 7.32 shows the robot tool setup, like the ECA setup with a laser tool for PPP. Figure 7.33 shows the laser calibration on the physical sample with figure 7.34 showing the PPP results in digital space on the reconstructed mesh. Figure 7.35 shows the cleaned mesh which the background is filtered, and the mesh is decimated and smoothed, then placed into path planning. Figure 7.36 shows the results for two different orientations after scanning, with the previous figures for the first orientation.

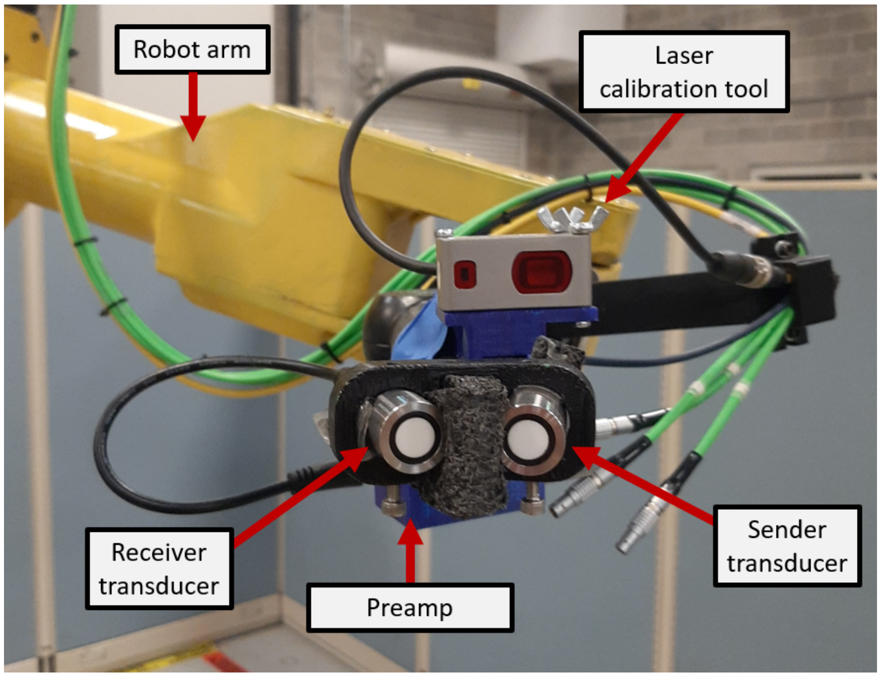


Figure 7.32 UT robotic scanner tool.

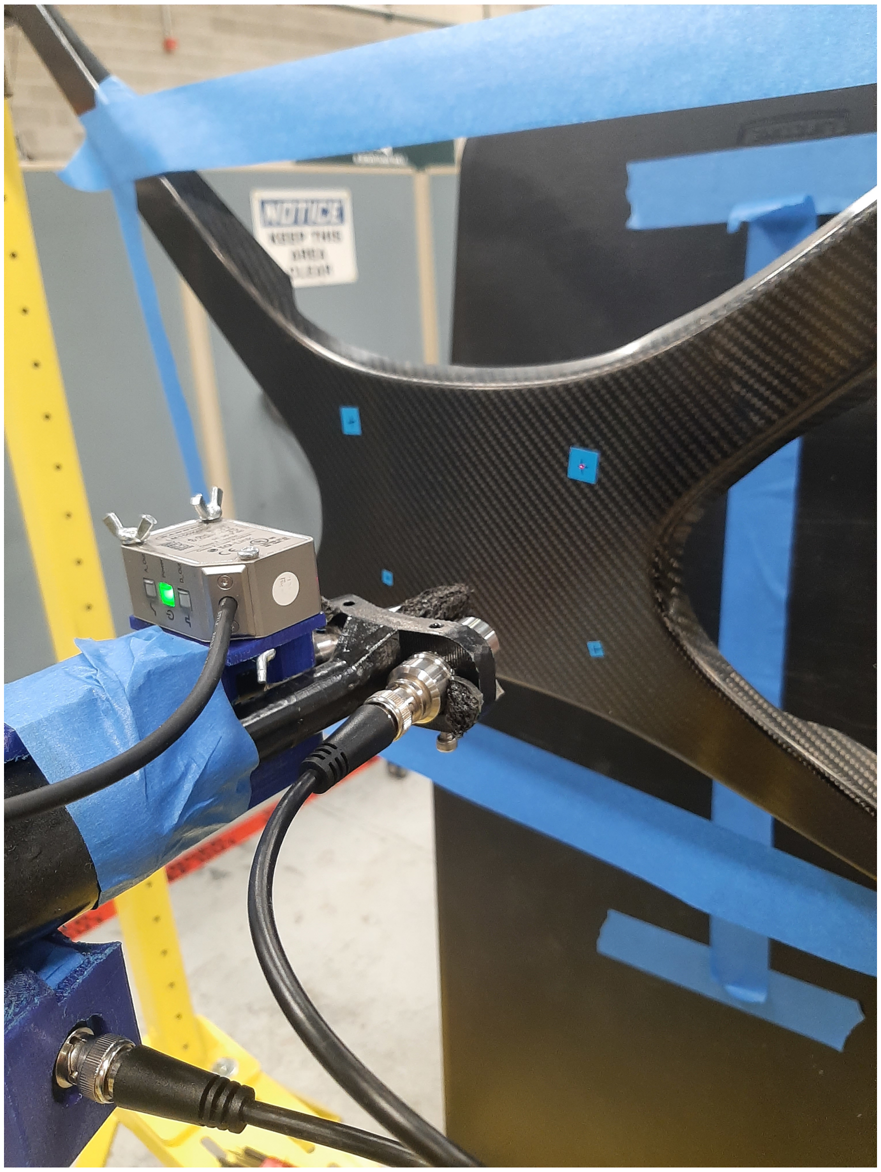


Figure 7.33 UT laser calibration on the CFRP sample.

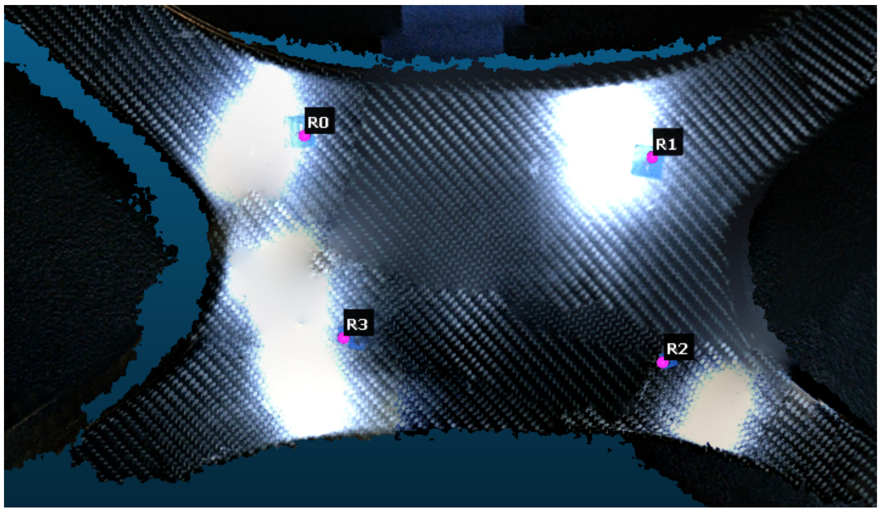


Figure 7.34 Point pairs picking on CFRP sample with under 0.1mm RMS error.

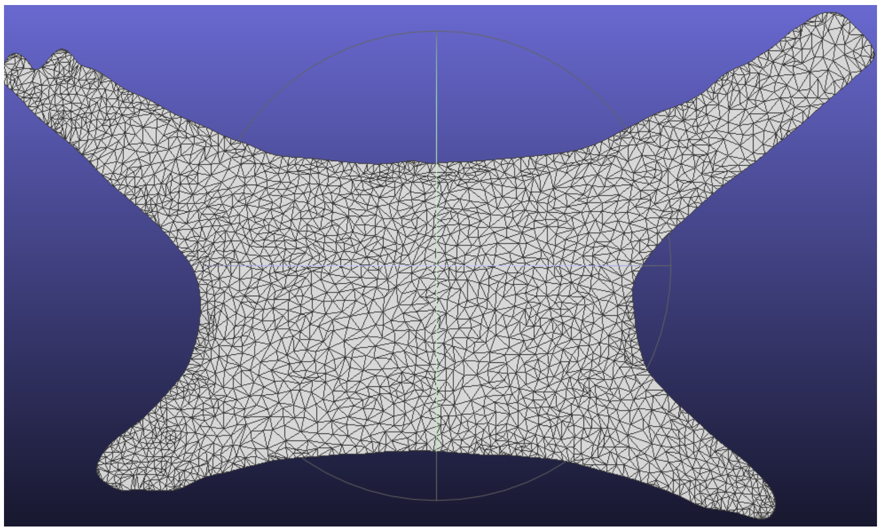
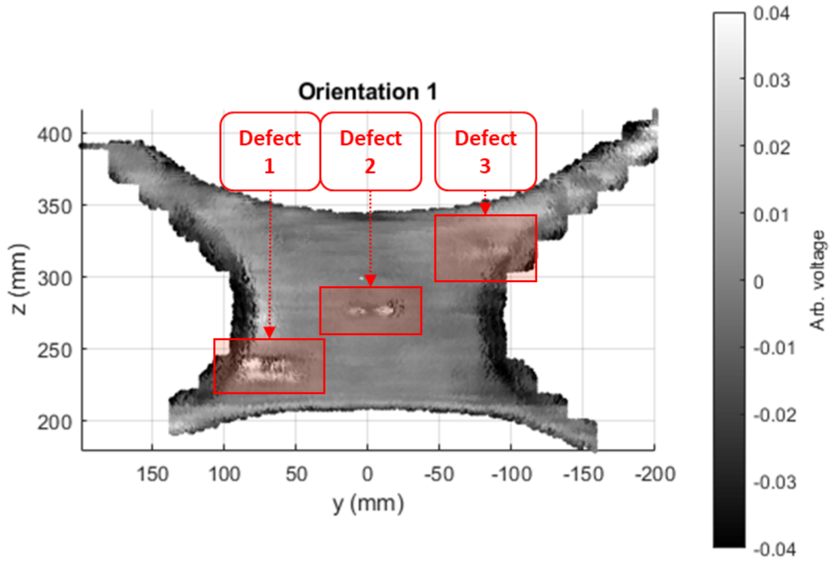
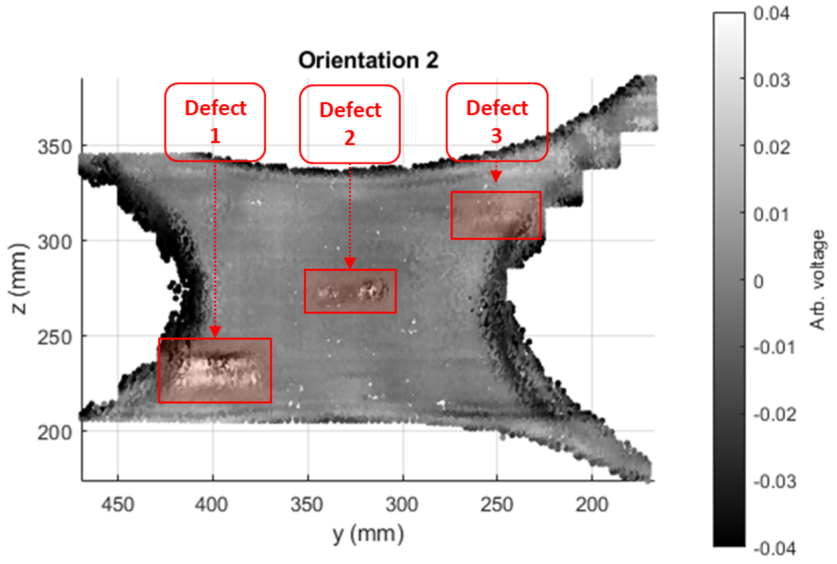


Figure 7.35 Cleaned CFRP mesh.



(a) Orientation 1



(b) Orientation 2

Figure 7.36 UT PTC results for two different orientations.

CHAPTER 8

CONCLUSION

In conclusion, the ECA freeform scanning system performed decent in terms of the complexity and complication of the scenario of detecting sub-mm depth corrosion defects. There are several considerations for improving the setup. Obviously, reducing the errors in terms of path planning is mostly desired, however other improvements such as path planning using the flexible capabilities of the coil or ways to further automate the system would be mostly desired. Covered will be the final thoughts on the objectives, limitations, and future works.

8.1 Objective Testing

This section will cover each objective mentioned in section 1.3 using the information highlighted within this chapter.

8.1.1 Objective 1

NDE complex surface scanning without a CAD model. The result from the provided framework enables surface scanning with respect to probe positioning. This is shown through the ECA and UT data provided. Despite some of the data loss from the ECA due to errors discussed in objective 2, the positioning of the probe with respect to the robot does recover information about the corrosion damages with respect to the curved steel samples. Likewise, the information of the UT probe shows the defect versus background information on the complex-shaped CFRP sample. Both of these were localized with respect to the robotic system using meshes generated from a reconstruction device (structured light), oriented to the robot's base frame using point pairs picking, pathing enabled on the mesh with a unique ray-triangle intersection array algorithm discussed in section 5.2, scanned with each surface scan, and later post-processed. Both scans were done on samples including curvature. One later examination should be ECA on more complex samples.

8.1.2 Objective 2

ECA scanning of corrosion using a freeform scanning system. The framework allows for corrosion detection using ECA scanning at a close non-contact lift-off for various depths and size defects. This is supported by objective 3 after post processing. The raw data contained issues that required post-processing, unfortunately removing any ease of voltage-to-depth conversion. The considerations were: mastering error for calibration of the robot joints creating complex errors throughout the scan, tilting error on the center of the coil causing variable lift-off effects on the edges of the coils, lift-off error for positioning error from path planning on the mesh surface in terms of reconstruction, orientation error on the body due to error in mesh registration, desynchronization between the ECA data and the robot's tool frame at the same time due to the systems running asynchronously, and other effects that occurred during scanning such as temperature fluctuations. The examinations were seen in each coil, from the full scans, and combined, from the fast scans.

Throughout later testing, it was found that there are certain effects that will cause variance and error in acquisition, though further validation on severity of these errors needs to be examined. These issues, due to non-linear nature of the scanning procedure, create certain difficulties that require more advanced post-processing techniques to solve in the future. First is that robot joint mastering error will add non-linear positional error to the eddy current array. This is non-linear due to the rotational behavior of each joint of the array, different from the expected result. This will add to Cartesian error that can be approximated via forward kinematics. However, when reading the eddy current complex voltages or impedance, this is another layer of complexity, as any error in lift-off is also non-linear and requires a model to determine this lift-off error. This sort of variance may be seen at various locations of the probe but will not suddenly jump but will create a "hill" on the image. This may also be seen in laser reconstructions, which one may assume would be useful for solving these issues, however one must consider that the kinematic solution to move the laser tool is different from the ECA probe as they are in two different tool frames, meaning the joint calculations which contain the base mastering errors are different and not entirely comparable.

The second consideration is voltage drifting. It is examined that if the robot is at a standstill, each joint locked in place, and placed on the sample, that there will be a voltage shift over time. The reasoning for this need's examination, but it is assumed that this is due to heat up of the coil. Other considerations may be fluctuations in room temperature, minor movement of the robot or otherwise probe or samples, or some other factor due to the device. Calibration of the coils or zeroing the complex voltage of each coil to the background, tends to fix this issue temporarily, but depending on changes for environmental factors, runs into voltage drift issues. The most significant issue with this voltage drift is that after a calibration curve converting the recorded importance of the probe to defect depths is this shift will inevitably add error to the expected fitting curve, shown in figure [below]. Such curves, which should be predetermined before a scan, will give different results if environmental factors change, making this process difficult without controlling environmental factors. Ideally, calibration should be done before each scan and ensuring that a stable signal is available, but since the calibration process will generate a new set of gain values, this will also change the calibration curve per coil.

8.1.3 Objective 3

ECA post-processing of corrosion with respect to robotic systems. Post-processing can be used to reduce errors that occur from robotic ECA scanning. The post processing procedures discussed recovered the precise corrosion data that is obstructed in the errors discussed in objective 2 on the raw data. The total process is shown in section 7.1.3. A significant unique post processing procedure was array subtraction, which is a relative of differential subtraction between coils. This process attempts to find common errors across each coil in order to reduce lift-off variance which was complex due to the nature of freeform robotic scanning on reconstructed and registered meshes and its associated errors. This algorithm is discussed in detail in section 6.3.1. The information from the performance metrics on the ECA coils show that there is a decent amount of detection, in terms of if a defect area is found in the ECA dataset, and the quality of detection using intersection masking between ECA and VHX datasets. There were still unrecoverable data that may require either different or improved post-processing procedures, or if the data is completely lost due to lift-off error, decreased errors in scanning. Voltage-to-depth is a considered prospect for later, as this process becomes difficult without a calibration metric and errors that were not fully controlled within the setup. Otherwise, the results of the ECA probe for freeform surface scanning on curved samples provided decent results, albeit not perfect.

8.2 Limitations

There were several limitations that affected the performance of the robotic scanning system, in terms of scan time, automation, and scan quality. First is the selection of the robot used, which if either the calibration is improved or an easier robot to work with were to be used would reduce complex lift-off errors. A Fanuc 100ib robotic arm was used, which was initially made for welding applications but repurposed for surface scan NDE. The robot ideally should contain sub-mm accuracy alone. However, the calibration process to master the robots, or set the joints as close to "zero" as possible, was not perfect. The process requires you to measure physical markings on two sections of the joints to match each other, which enables sub-degree error. The joint errors in the processes added approximately 1mm of error in positioning, in a complex pattern due to it being error on the joints causing orientation errors on the probe. The robot was also refurbished, which may enable errors in the joints as well. One other issue in terms of automation is the ability to enable real-time scanning ran directly from the simulation on the computer. This process instead required a disk to be placed in between the robot and the computer instead of direct commands being run. This capability may be possible in the current setup but was not executed due to development complications.

Another source of improvement would be the reconstruction device used and alignment of the obtained mesh. A Creality CR-01 scanner was used, which has 0.1mm accuracy on the mesh. The issue is that with structured light, there may be spots of missing points due to high reflection, either by glossiness of the material or it being bright in color. In some scans, the lights inside of the laboratory would interfere with reconstruction. On the other side, if a material absorbed light due to it being too dark, then points will also become missing. For both types of bright and dark areas that are reconstructed, there may still be errors due to these material properties. It may be better to use a reconstruction device using blue-light which should remove these issues. This device also required commercial software to operate. This was great for enabling ease of use for reconstructing a point cloud and outputting to a mesh, however due to not having certain controls limited the setup in terms of automation. Ideally, the SL would be used in a pipeline process, where essentially a "run scan" button is pressed, and the robot reconstructs the mesh, does any required processing such as segmentation of the sample vs background or smoothing, runs the path planning, runs a simulation, then runs the NDE scan, from just the user inputs for starting the scan and perhaps setting up the path planning. Instead, the software had to be run separately. Another process of automation would be to use center point transformation (CPT) as discussed in section 4.2. Ideally, with the known location of the SL probe on the robot, point clouds on the mesh can be automated as such to reconstruct and align the mesh with respect to the robot's workspace. This would not require the processing techniques of Creality for scanning larger samples, which Creality sometimes loses orientation. This would also automate the process, in which point pair picking (PPP) which was used in this work to compare physical locations within the workspace to the virtual reconstruction. CPT ideally would enable automated reconstruction. However, in practice, the center point of the mesh did not align with the expected tool location from testing. This error enabled centimeters worth of error, which would make surface scanning not viable. If control was enabled on the reconstruction sensor to enable operation within a pipeline while maintaining the expected alignment from the device, there would be a significant improvement in terms of automation and scan time, in which an extra 20 minutes per scan were dedicated just in setting up the reconstruction for path planning. Tool pathing errors from reconstruction and registrations in general should also be examined to be reduced.

Improvement in scan time and quality should also be improved upon. The robot ran at 25mm/s to prevent synchronization errors between both the ECA datasets and the expected location of the ECA from the robot's tool frame. Both sets of data were obtained asynchronously but also using the computers clock. The issue is that the timing between the two sets were variable, as the ECA used a commercial instrument that might slightly delay the timing, whereas the robots' controllers, which were old, added 50ms of delay that was variable. If this timing was exact, then the robots could move much faster. This, paired with fast scanning which paints, would enable extremely fast inspections, important for larger scanning specimens. One other improvement would be to use better curved pathing, such as with splines. This would require an adjustment to the ray-triangle intersection algorithm which instead uses planes for intersection that would run along the scan, then generate a spline based on that path. This would allow the robot to run more smoothly rather than relying on several points on the path. This potentially would reduce tilt; however, this requires testing. As mentioned before, the scan should be more automated to decrease setup times as reduce human interaction within the system so scans may be run in the background.

Finally, for discussed limitations is how to increase the quality of scanning. First, there should be an examination of which type of ECA scanning should be used. Absolute scanning mode was used, which simply records the impedance information of returned signals per coil, returning both the real and imaginary voltage components per data point. The issue with this is that absolute mode at high frequencies is susceptible to temperature fluctuations, which may be due to heating of the coils or the scanning environment. The scanning environment itself was not set up to counter these errors, so a more insulated setup would be preferred. However, in a real-world application, it may not be ideal to have such a setup, especially for extended robotic setups including mobility as mentioned soon. Reducing errors in tilt would be preferred, as just mentioned with improving reconstruction and robotic setups. If these errors were accounted for, this may enable voltage-to-depth inspection, which would help qualify the results of the voltage in terms of depth.

8.3 Future Works

Despite the limitations and shortcomings mentioned, the ECA freeform scanning system developed for this work can detect corrosion on curved samples to a decent extent. There is room for improvement and extension of this system. As the final words, here are some future works that may be considered. On top of improvements in the limitations, there are several extensions of this work that may be viable. Improvements or further development of data fusion may be considered. This may enable fusion between the structured light or other reconstruction device with the ECA data or may even have several different methods in tangent. For quality metrics, VHX was used, but that is far from a portable device that should be placed onto a robot inspection system. AI may also be a valuable tool. This can be used in tangent with data fusion or enable a comparison with simulated results to attempt to understand each ECA signal in terms of signal detected and maybe even lift-off detection. Finally, expansion into mobile platforms with different surface scanning systems considered. Such may be for ground-based vehicles, crawler robots, underwater vehicles, or aerial vehicles. Each system may have a static placed sensor, or even an arm attached. Such systems would enable an appropriate scanning environment for larger systems. Otherwise, the freeform ECA robotic inspection system currently designed may be used for applications either within a pipeline for inspection of fresh materials, or a "car wash" system to inspect larger components such as airplanes or mobile vehicles which require inspection.

BIBLIOGRAPHY

[1] The meaning of r.u.r. *Saturday Review July 21*, 136(79), 1923.

[2] Automated control of welding penetration based on audio sensing technology. *Journal of Materials Processing Technology*, 250:81–98, 2017.

[3] Haider A Abdulkarim and Ibrahim F Alshammari. Comparison of algorithms for solving traveling salesman problem. *International Journal of Engineering and Advanced Technology*, 4(6):76–79, 2015.

[4] Guillem Alenyà, Sergi Foix, and Carme Torras. Tof cameras for active vision in robotics. *Sensors and Actuators A: Physical*, 218:10–22, 2014.

[5] Mohand Alzuhiri. *Multi-Modality Nondestructive Evaluation Techniques for Inspection of Plastic and Composite Pipeline Networks*. PhD thesis, Michigan State University, 2022.

[6] Dave Tong Anders Grunnet-Jepsen. Depth post-processing for intel realsense depth camera d400 seriess.

[7] Oleg V Angelsky, Aleksandr Y Bekshaev, Steen G Hanson, Claudia Yu Zenkova, Igor I Mokhun, and Zheng Jun. Structured light: ideas and concepts. *Frontiers in Physics*, 8:114, 2020.

[8] Ghassan Aouad, Song Wu, Angela Lee, and Timothy Onyenobi. *Computer aided design guide for architecture, engineering and construction*. Routledge, 2013.

[9] David L Applegate, Robert E Bixby, Vašek Chvátal, and William J Cook. The traveling salesman problem. In *The Traveling Salesman Problem*. Princeton university press, 2011.

[10] Isaac Asimov. Three laws of robotics. *Asimov, I. Runaround*, 1941.

[11] ASTM B117-03. Standard practice for operating salt spray apparatus. 2003.

[12] Saeed Asadi Bagloee, Madjid Tavana, Mohsen Asadi, and Tracey Oliver. Autonomous vehicles: challenges, opportunities, and future implications for transportation policies. *Journal of modern transportation*, 24(4):284–303, 2016.

[13] John W Bandler. Optimization methods for computer-aided design. *IEEE Transactions on Microwave Theory and Techniques*, 17(8):533–552, 1969.

[14] Firoozi F Capello S Kolios E Perrotti M Belarmino J, Moran ME. Tesla's robot and the dawn of the current era. *Society of urology and engineering*, 19(7), 2005.

[15] Ralf B Bergmann, Florian T Bessler, and Walter Bauer. Non-destructive testing in the

automotive supply industry. *Research and Advance Engineering*, 2007.

[16] Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean geometry*, pages 47–123. World Scientific, 1995.

[17] Evandro Bernardes and Stéphane Viollet. Quaternion to euler angles conversion: A direct, general and computationally efficient method. *Plos one*, 17(11):e0276302, 2022.

[18] Abdenacer Berradja. Electrochemical techniques for corrosion and tribocorrosion monitoring: fundamentals of electrolytic corrosion. *Corrosion Inhibitors*, 2019.

[19] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[20] JV Birnie. Practical implications of programmable manipulators. *Industrial Robot: An International Journal*, 1974.

[21] José Luis Blanco-Claraco. A tutorial on mathbf(se)(3) transformation parameterizations and on-manifold optimization. *arXiv preprint arXiv:2103.15980*, 2021.

[22] Robert Bogue. The role of robotics in non-destructive testing. *Industrial Robot: An International Journal*, 2010.

[23] Robert Bogue. The role of robotics in non-destructive testing. *Industrial Robot: An International Journal*, 37(5):421–426, 2010.

[24] Lucia Botti, Emilio Ferrari, and Cristina Mora. Automated entry technologies for confined space work activities: A survey. *Journal of Occupational and Environmental Hygiene*, 14(4):271–284, 2017. PMID: 27754794.

[25] Denys Breysse. Nondestructive evaluation of concrete strength: An historical review and a new perspective by combining ndt methods. *Construction and Building Materials*, 33:139–163, 2012.

[26] Nick Brierley, Trevor Tippetts, and Peter Cawley. Improving the reliability of automated non-destructive inspection. In *AIP Conference Proceedings*, volume 1581, pages 1912–1919. American Institute of Physics, 2014.

[27] Sergio Cantero-Chinchilla, Paul D Wilcox, and Anthony J Croxford. Deep learning in automated ultrasonic nde–developments, axioms and opportunities. *NDT & E International*, page 102703, 2022.

[28] Josef Capek. *Lelio a Pro delfína.* Aventinum, Prague, 1925.

[29] Karel Capek. *RUR (Rossum's universal robots).* Penguin, 2004.

[30] Laura A Carlson. Selecting a reference frame. *Spatial cognition and computation*, 1(4):365–379, 1999.

[31] Jasmine Cashbaugh and Christopher Kitts. Automatic calculation of a transformation matrix between two frames. *IEEE Access*, 6:9614–9622, 2018.

[32] Long Chen and Jin-chao Xu. Optimal delaunay triangulations. *Journal of Computational Mathematics*, pages 299–308, 2004.

[33] Xi-Zhang Chen, Yu-Ming Huang, and Shan-ben Chen. Model analysis and experimental technique on computing accuracy of seam spatial position information based on stereo vision for welding robot. *Industrial Robot: An International Journal*, 2012.

[34] Mingmei Cheng, Le Hui, Jin Xie, and Jian Yang. Sspc-net: Semi-supervised semantic 3d point cloud segmentation network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1140–1147, 2021.

[35] E James Chern, HP Chu, and JN Yang. Assessment of probability of detection of delaminations in fiber-reinforced composites. Technical report, 1991.

[36] Ian Chivers and Jane Sleightholme. An introduction to algorithms and the big o notation. In *Introduction to programming with Fortran*, pages 359–364. Springer, 2015.

[37] CloudCompare. Alignment and registration.

[38] Ian Cornwell and Alistair McNab. Towards automated interpretation of ultrasonic ndt data. *NDT & E International*, 32(2):101–107, 1999.

[39] Intel Corporation. Post-processing filters.

[40] Bart Custers. Drones here, there and everywhere introduction and overview. In *The future of drone use*, pages 3–20. Springer, 2016.

[41] Magdalena Czaban. Aircraft corrosion – review of corrosion processes and its effects in selected cases. *Fatigue of Aircraft Structures*, 2018, 05 2019.

[42] Robert L Dahlstrom. The emergence of contact based nondestructive testing ndt at height utilizing aerial robotic drone systems. In *Offshore technology conference*. OnePetro, 2020.

[43] Kadir Alpaslan Demir and H Cicibas. Industry 5.0 and a critique of industry 4.0. In *Proceedings of the 4th international management information systems conference, Istanbul, Turkey*, pages 17–20, 2017.

[44] Kadir Alpaslan Demir and Halil Cicibaş. The next industrial revolution: industry 5.0 and discussions on industry 4.0, industry 4.0 from the management information systems

perspectives, 2018.

[45] Kadir Alpaslan Demir, Gözde Döven, and Bülent Sezen. Industry 5.0 and human-robot co-working. *Procedia computer science*, 158:688–695, 2019.

[46] Jacques Denavit and Richard S Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. 1955.

[47] L Derecichei, C Lucaci, G Cheregi, L Lustun, et al. Simulation of sculptural surface processing in wood in 5 axis cnc with sprutcam program. *Analele Universității din Oradea, Fascicula: Protecția Mediului*, 29:165–172, 2017.

[48] Jr George C Devol. Programmed article transfer, June 13 1961. US Patent 2,988,237.

[49] Carmelo Di Franco and Giorgio Buttazzo. Coverage path planning for uavs photogrammetry with energy and resolution constraints. *Journal of Intelligent & Robotic Systems*, 83(3):445–462, 2016.

[50] Henry Winram Dickinson. *A short history of the steam engine*. Cambridge University Press, 2011.

[51] Living Dolls. A magical history of the quest for mechanical life, 2002.

[52] Ugyen Dorji and Reza Ghomashch. Hydro turbine failure mechanisms: An overview. *Engineering Failure Analysis*, 44:136–147, 2014.

[53] Rainer Drath and Alexander Horch. Industrie 4.0: Hit or hype?[industry forum]. *IEEE industrial electronics magazine*, 8(2):56–58, 2014.

[54] David Eberly. Quaternion algebra and calculus. *Magic Software Inc*, 26, 2002.

[55] David Eberly. *3D game engine design: a practical approach to real-time computer graphics*. CRC Press, 2006.

[56] Joseph Engelberger. Robots in service. *Biddler, Ltd. Great Britain*, 1989.

[57] EuclideanSpace. Maths - angle between vectors.

[58] EuclideanSpace. Maths - axisangle to quaternion.

[59] Abbas Fahr. *Aeronautical applications of non-destructive testing*. DEStech Publications, Inc, 2013.

[60] Andrew Forbes, Michael de Oliveira, and Mark R Dennis. Structured light. *Nature Photonics*, 15(4):253–262, 2021.

[61] James Foreman-Peck. The american challenge of the twenties: Multinationals and the european motor industry. *The Journal of Economic History*, 42(4):865–881, 1982.

[62] Euan Alexander Foster, Gary Bolton, Robert Bernard, Martin McInnes, Shaun McKnight, Ewan Nicolson, Charalampos Loukas, Momchil Vasilev, Dave Lines, Ehsan Mohseni, et al. Automated real-time eddy current array inspection of nuclear assets. *Sensors*, 22(16):6036, 2022.

[63] C Galleguillos, A Zorrilla, A Jimenez, L Diaz, ÁL Montiano, M Barroso, A Viguria, and F Lasagni. Thermographic non-destructive inspection of wind turbine blades using unmanned aerial systems. *Plastics, Rubber and Composites*, 44(3):98–103, 2015.

[64] Zhen Gao, Tom Wanyama, Ishwar Singh, Anoop Gadhrri, and Reiner Schmidt. From industry 4.0 to robotics 4.0-a conceptual framework for collaborative and intelligent robotic systems. *Procedia manufacturing*, 46:591–599, 2020.

[65] Javier García-Martín, Jaime Gómez-Gil, and Ernesto Vázquez-Sánchez. Non-destructive techniques based on eddy current testing. *Sensors*, 11(3):2525–2565, 2011.

[66] Michael R Garey, David S. Johnson, and R Endre Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

[67] Alessandro Gasparetto and Lorenzo Scalera. A brief history of industrial robotics in the 20th century. *Advances in Historical Studies*, 8(1):24–35, 2019.

[68] Eduardo SL Gastal and Manuel M Oliveira. Domain transform for edge-aware image and video processing. In *ACM SIGGRAPH 2011 papers*, pages 1–12. 2011.

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

[70] Lian Genkuan. Analysis of causes of boiler accidents in power plant and accident handling based on mathematical statistics. *2018 International Conference on Engineering Simulation and Intelligent Control*, 2018, 2019.

[71] Sanchit Goyal. A survey on travelling salesman problem. In *Midwest instruction and computing symposium*, pages 1–9, 2010.

[72] S Gripp. A twin robot approach for ut inspection and porosity evaluation of complex shaped helicopter components. *ECNDT 2006*, 2006.

[73] Canzhi Guo, Chunguang Xu, Juan Hao, Dingguo Xiao, and Wanxin Yang. Ultrasonic non-destructive testing system of semi-enclosed workpiece with dual-robot testing system. *Sensors*, 19(15):3359, 2019.

[74] Mridul Gupta, Muhsin Ahmad Khan, Ravi Butola, and Ranganath M Singari. Advances in applications of non-destructive testing (ndt): A review. *Advances in Materials and Processing Technologies*, pages 1–22, 2021.

[75] Ciaron Hamilton, Oleksii Karpenko, Lalita Udpa, Mahmoodul Haq, and Yiming Deng. Efficient data acquisition and reconstruction for air-coupled ultrasonic robotic nde. *Scientific Reports*, 14(1):14021, 2024.

[76] Ciaron Nathan Hamilton. *5d Nondestructive Evaluation: Object Reconstruction to Toolpath Generation*. PhD thesis, Michigan State University, 2021.

[77] J Hansen. The eddy current inspection method. *Insight*, 46(5):279–281, 2004.

[78] S Harsimran, K Santosh, and K Rakesh. Overview of corrosion and its control: A critical review. *Proc. Eng. Sci*, 3(1):13–24, 2021.

[79] Charles Hellier. *Handbook of nondestructive evaluation*. McGraw-Hill, 2001.

[80] Neil G Hockstein, CG Gourin, RA Faust, and David J Terris. A history of robots: from science fiction to surgical robotics. *Journal of robotic surgery*, 1(2):113–118, 2007.

[81] Stephen D Holland and Adarsh Krishnamurthy. Registration of nde data to cad. In *Handbook of Nondestructive Evaluation 4.0*, pages 1–35. Springer, 2021.

[82] Berthold KP Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. 1970.

[83] Ruiting Huang, Mengyao Jia, and Simeng Guo. Track detection system of uav based on line structure light. In *Journal of Physics: Conference Series*, volume 1453, page 012087. IOP Publishing, 2020.

[84] Yanbo Huang, LAN Yubin, WC Hoffmann, and RE Lacey. A pixel-level method for multiple imaging sensor data fusion through artificial neural networks. *Advances in Natural Science*, 4(1):01–13, 2011.

[85] Thomas Parke Hughes. *American genesis: a century of invention and technological enthusiasm, 1870-1970*. University of Chicago Press, 2004.

[86] ME Ibrahim. Nondestructive testing and structural health monitoring of marine composite structures. In *Marine applications of advanced fibre-reinforced composites*, pages 147–183. Elsevier, 2016.

[87] Nursen Işık, Fatma Meral Halifeoğlu, and Süleyman Ipek. Nondestructive testing techniques to evaluate the structural damage of historical city walls. *Construction and Building Materials*, 253:119228, 2020.

[88] David Jeske. Simplescene: 3d scene manager in c-sharp and opentk, 2014.

[89] David S Johnson and Lyle A McGeoch. The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 1(1):215–310, 1997.

[90] Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. The traveling salesman problem. *Handbooks in operations research and management science*, 7:225–330, 1995.

[91] Henning Kagermann, Wolf-Dieter Lukas, and Wolfgang Wahlster. Industrie 4.0: Mit dem internet der dinge auf dem weg zur 4. industriellen revolution. *VDI nachrichten*, 13(1):2–3, 2011.

[92] Niharika Karnik, Urvi Bora, Karan Bhadri, Prasanna Kadambi, and Pankaj Dhatrak. A comprehensive study on current and future trends towards the characteristics and enablers of industry 4.0. *Journal of Industrial Information Integration*, 27:100294, 2022.

[93] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.

[94] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstructionn. *ACM Transactions on Graphics*, 2013.

[95] Prasarn Kiddee, Zaojun Fang, and Min Tan. An automated weld seam tracking system for thick plate using cross mark structured light. *The International Journal of Advanced Manufacturing Technology*, 87(9):3589–3603, 2016.

[96] Dennis Kolberg, Christoph Berger, B-C Pirvu, Marco Franke, and Joachim Michniewicz. Cyprof–insights from a framework for designing cyber-physical systems in production environments. *procedia Cirp*, 57:32–37, 2016.

[97] Vladislav Kolchinskiy, Cheng-Hung Shih, Ikai Lo, and Roman Romashko. Refractive index measurement using the laser profiler. *Physics Procedia*, 86:176–180, 2017.

[98] Akshi Kumar and Divya Gupta. Challenges within the industry 4.0 setup. In *A Roadmap to Industry 4.0: Smart Production, Sharp Business and Sustainable Development*, pages 187–205. Springer, 2020.

[99] Naomi R Lamoreaux, Margaret Levenstein, and Kenneth L Sokoloff. Financing invention during the second industrial revolution: Cleveland, ohio, 1870-1920, 2004.

[100] HJ Landau. Sampling, data transmission, and the nyquist rate. *Proceedings of the IEEE*, 55(10):1701–1706, 1967.

[101] David Lattanzi and Gregory Miller. Review of robotic infrastructure inspection systems. *Journal of Infrastructure Systems*, 23(3):04017004, 2017.

[102] Nalpantidis Lazaros, Georgios Christou Sirakoulis, and Antonios Gasteratos. Review of stereo vision algorithms: from software to hardware. *International Journal of Optomechatronics*, 2(4):435–462, 2008.

[103] Jiewu Leng, Weinan Sha, Baicun Wang, Pai Zheng, Cunbo Zhuang, Qiang Liu, Thorsten Wuest, Dimitris Mourtzis, and Lihui Wang. Industry 5.0: Prospect and retrospect. *Journal of Manufacturing Systems*, 65:279–295, 2022.

[104] Marc Levoy. History of computer graphics. 2004.

[105] Jingjing Li, Guanghui Zhou, and Chao Zhang. A twin data and knowledge-driven intelligent process planning framework of aviation parts. *International Journal of Production Research*, pages 1–18, 2021.

[106] Zi Li, Jianliang Li, Adithya Rao, Yiming Deng, et al. Synchronous and concurrent multi-dimensional structured light sensing with efficient automatic calibration.

[107] John DC Little, Katta G Murty, Dura W Sweeney, and Caroline Karel. An algorithm for the traveling salesman problem. *Operations research*, 11(6):972–989, 1963.

[108] Songping Liu, Feifei Liu, Yusen Yang, Legang Li, and Zhiying Li. Nondestructive evaluation 4.0: ultrasonic intelligent nondestructive testing and evaluation for composites. *Research in Nondestructive Evaluation*, 31(5-6):370–388, 2020.

[109] Zheng Liu, Norbert Meyendorf, and Nezih Mrad. The role of data fusion in predictive maintenance using digital twin. In *AIP conference proceedings*, volume 1949, page 020023. AIP Publishing LLC, 2018.

[110] Zongxing Lu, Chunguang Xu, Qinxue Pan, Fanwu Meng, and Xinliang Li. Automatic method for synchronizing workpiece frames in twin-robot nondestructive testing system. *Chinese Journal of Mechanical Engineering*, 28(4):860–868, 2015.

[111] Zongxing Lu, Chunguang Xu, Qinxue Pan, Xinyu Zhao, and Xinliang Li. Inverse kinematic analysis and evaluation of a robot for nondestructive testing application. *Journal of Robotics*, 2015, 2015.

[112] Albrecht Maurer, WD Odorico, Roland Huber, and Thierry Laffont. Aerospace composite testing solutions using industrial robots. In *18th World Conference on Nondestructive Testing, Durban, South Africa*, pages 1–7, 2012.

[113] Ross McMillan, Seyed Morteza Tabatabaeipour, Konstantinos Tzaferis, William Jackson, Rachel S Edwards, Oksana Trushkevych, Charles Macleod, Gordon Dobie, and Anthony Gachagan. Crawler-based automated non-contact ultrasonic inspection of large structural assets. *49th Annual Review of Progress in Quantitative Nondestructive Evaluation*, 2022.

[114] N Meyendorf, B Frankenstein, and L Schubert. Structural health monitoring for aircraft, ground transportation vehicles, wind turbines and pipes—prognosis, 2012.

[115] Norbert G Meyendorf, Leonard J Bond, J Curtis-Beard, S Heilmann, Saveri Pal, R Schallert, H Scholz, and C Wunderlich. Nde 4.0–nde for the 21st century–the internet of things and cyber physical systems will revolutionize nde. In *15th Asia Pacific conference for non-destructive testing (APCNDT 2017), Singapore*, 2017.

[116] Matthew Bolitho Michael Kazhdan and Hugues Hoppe. Poisson surface reconstruction. *Eurographics Symposium on Geometry Processing*, 2006.

[117] Microsoft. C-sharp guide.

[118] Edward M Mikhail, James S Bethel, and J Chris McGlone. *Introduction to modern photogrammetry*. John Wiley & Sons, 2001.

[119] C Mineo, SG Pierce, B Wright, I Cooper, and PI Nicholson. Paut inspection of complex-shaped composite materials through six dofs robotic manipulators. *Insight-Non-Destructive Testing and Condition Monitoring*, 57(3):161–166, 2015.

[120] Carmelo Mineo, Donatella Cerniglia, and Alastair Poole. Autonomous robotic sensing for simultaneous geometric and volumetric inspection of free-form parts. *Journal of Intelligent & Robotic Systems*, 105(3):54, 2022.

[121] Carmelo Mineo and Yashar Javadi. Robotic non-destructive testing, 2022.

[122] Carmelo Mineo, Nicola Montinaro, Mario Fustaino, Antonio Pantano, and Donatella Cerniglia. Fine alignment of thermographic images for robotic inspection of parts with complex geometries. *Sensors*, 22(16):6267, 2022.

[123] Carmelo Mineo, Stephen Gareth Pierce, Pascual Ian Nicholson, and Ian Cooper. Robotic path planning for non-destructive testing–a custom matlab toolbox approach. *Robotics and Computer-Integrated Manufacturing*, 37:1–12, 2016.

[124] Tomas Möller and Ben Trumbore. Fast, minimum storage ray/triangle intersection. In *ACM SIGGRAPH 2005 Courses*, pages 7–es. 2005.

[125] Jérôme Monnot and Sophie Toulouse. The traveling salesman problem and its variations. *Paradigms of combinatorial optimization: problems and new approaches*, pages 173–214, 2014.

[126] Michael E Moran. Evolution of robotic arms. *Journal of robotic surgery*, 1(2):103–111, 2007.

[127] M Morozov, SG Pierce, Ch N MacLeod, C Mineo, and R Summan. Off-line scan path

planning for robotic ndt. *Measurement*, 122:284–290, 2018.

[128] Nawfel Muhammed Baqer Muhsin. Review on engineering methods in treatment of chemical rust. *International Journal of Chemical and Molecular Engineering*, 6(2):49–53, 2020.

[129] Subrata Mukherjee, Renrui Zhang, Mohand Alzuhiri, Varun Venkat Rao, Lalita Udpa, and Yiming Deng. Inline pipeline inspection using hybrid deep learning aided endoscopic laser profiling. *Journal of Nondestructive Evaluation*, 41(3):1–13, 2022.

[130] Robert W Neuschaefer and James B Beal. Assessment of and standardization for quantitative nondestructive test. Technical report, 1972.

[131] Anh Nguyen and Bac Le. 3d point cloud segmentation: A survey. In *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*, pages 225–230. IEEE, 2013.

[132] A Normandy. Xliii. on the spheroidal state of water in steam-boilers. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 7(45):283–286, 1854.

[133] Cristina Nuzzi, Simone Pasinetti, Roberto Pagani, Stefano Ghidini, Manuel Beschi, Gabriele Coffetti, and Giovanna Sansoni. Meguru: a gesture-based robot program builder for meta-collaborative workstations. *Robotics and Computer-Integrated Manufacturing*, 68:102085, 2021.

[134] International Federation of Robotics. World robotics 2012, executive summary.

[135] International Federation of Robotics. World robotics 2022, executive summary.

[136] P Olivieri, L Birglen, X Maldague, and I Mantegh. Coverage path planning for eddy current inspection on complex aeronautical parts. *Robotics and Computer-Integrated Manufacturing*, 30(3):305–314, 2014.

[137] Reinhold Oster. Non-destructive testing methodologies on helicopter fiber composite components challenges today and in the future. In *18th World Conference on Nondestructive Testing*, pages 16–20, 2012.

[138] Ercan Oztemel and Samet Gursev. Literature review of industry 4.0 and related technologies. *Journal of Intelligent Manufacturing*, 31(1):127–182, 2020.

[139] Bradley Evan Paden. Kinematics and control of robot manipulators. *Ph. D. Thesis*, 1985.

[140] Geethamani Palanisamy. Corrosion inhibitors. *Corrosion inhibitors*, 2019:1–24, 2019.

[141] Soo-Jin Park. History and structure of carbon fibers. In *Carbon Fibers*, pages 1–30. Springer, 2018.

[142] Sun-Oh Park, Min Cheol Lee, and Jaehyung Kim. Trajectory planning with collision avoidance for redundant robots using jacobian and artificial potential field-based real-time inverse kinematics. *International Journal of Control, Automation and Systems*, 18(8):2095–2107, 2020.

[143] Olivier Patrouix, Sébastien Bottecchia, and Joseph Canou. Improving robotized non destructive testing for large parts with local surface approximation and force control scheme. In *THE 19TH INTERNATIONAL CONFERENCE ON COMPOSITE MATERIALS*, pages 7913–7921, 2013.

[144] Flávia Pires, Ana Cachada, José Barbosa, António Paulo Moreira, and Paulo Leitão. Digital twin in industry 4.0: Technologies, applications and challenges. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 721–726. IEEE, 2019.

[145] Zhi-jie Qiao, Jia Liu, Sheng-qiang Yang, Jing-jing Zhang, and Jing-zheng Li. Robot-assisted blade polishing and process control based on sprutcam offline programming software. In *2021 33rd Chinese Control and Decision Conference (CCDC)*, pages 345–349. IEEE, 2021.

[146] Marco Ricci, Giuseppe Silipigni, Luigi Ferrigno, Marco Laracca, Ibukun D Adewale, and Gui Yun Tian. Analysis of the influence of lift-off variation on eddy-current images. In *2015 IEEE Metrology for Aerospace (MetroAeroSpace)*, pages 182–187. IEEE, 2015.

[147] D Roach and P Walkington. Development and validation of nondestructive inspection techniques for composite doubler repairs on commercial aircraft. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 1998.

[148] ARTHUR ROGERS. Some causes of derailment on railway-curves.(including appendixes). In *Minutes of the Proceedings of the Institution of Civil Engineers*, volume 185, pages 263–276. Thomas Telford-ICE Virtual Library, 1911.

[149] Mark Rosheim. *Leonardo's Lost Robots*. Springer Science & Business Media, 2006.

[150] Sajjad Sayyar Roudari, Theophilus Okore-Hanson, Sameer Hamoush, Sun Yi, and Ahmed Megri. Robotic nondestructive evaluation of rc structures using 3d vision camera, ie, and gpr. In *American Society for Nondestructive Testing*, 2019.

[151] Ward D Rummel. Nondestructive inspection reliability history, status and future path. In *Proceedings of the 18th World Conference on Nondestructive, Durban, South Africa*, pages 16–20, 2010.

[152] Amanur Rahman Saiyed. The traveling salesman problem. *Indiana State University*, 2:1–15, 2012.

[153] Gene Salecker. *Disaster on the Mississippi: The Sultana Explosion, April 27, 1865*. Naval

Institute Press, 2015.

[154] Joaquim Salvi, Sergio Fernandez, Tomislav Pribanic, and Xavier Llado. A state of the art in structured light patterns for surface profilometry. *Pattern recognition*, 43(8):2666–2680, 2010.

[155] Rohan Sawhney and Keenan Crane. Boundary first flattening. *ACM Transactions on Graphics (ToG)*, 37(1):1–14, 2017.

[156] Xiaodong Shi, Anthony Olvera, Ciaron Hamilton, Erzhuo Gao, Jiaoyang Li, Lucas Utke, Andrew Petruska, Zhenzhen Yu, Lalita Udpa, Yiming Deng, et al. Ai-enabled robotic nde for structural damage assessment and repair. *Materials Evaluation*, 79(7), 2021.

[157] Ripi Singh. The next revolution in nondestructive testing and evaluation: what and how? *Materials Evaluation*, 77(1):45–50, 2019.

[158] Gregory G Slabaugh. Computing euler angles from a rotation matrix. *Retrieved on August*, 6(2000):39–63, 1999.

[159] Ali Sophian. Non-destructive testing (ndt) in industry 4.0: A brief review. In *Proceeding International Conference on Science (ICST)*, volume 2, pages 1–9, 2021.

[160] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, 2004.

[161] Mark W Spong and Mathukumalli Vidyasagar. *Robot dynamics and control*. John Wiley & Sons, 2008.

[162] W Swiderski. Military applications of infrared thermography nondestructive testing in poland. In *16th World Conference on Nondestructive Testing, Montreal-Canada*, 2004.

[163] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[164] Ninad Thakoor, Venkat Devarajan, and Jean Gao. Computation complexity of branch-and-bound model selection. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1895–1900. IEEE, 2009.

[165] DJ Titman. Applications of thermography in non-destructive testing of structures. *NDT & e International*, 34(2):149–154, 2001.

[166] Hamid A Toliyat, Karim Abbaszadeh, Mina M Rahimian, and Leslie E Olson. Rail defect diagnosis using wavelet packet decomposition. *IEEE Transactions on Industry Applications*, 39(5):1454–1461, 2003.

[167] VA Udod, Ya Van, SP Osipov, SV Chakhlov, E Yu Usachev, MB Lebedev, and AK Temnik. State-of-the art and development prospects of digital radiography systems for nondestructive testing, evaluation, and inspection of objects: a review. *Russian Journal of Nondestructive Testing*, 52(9):492–503, 2016.

[168] Alex Vary. Nondestructive evaluation technique guide. Technical report, 1973.

[169] Momchil Vasilev, Charles N MacLeod, Charalampos Loukas, Yashar Javadi, Randika KW Vithanage, David Lines, Ehsan Mohseni, Stephen Gareth Pierce, and Anthony Gachagan. Sensor-enabled multi-robot system for automated welding and in-process ultrasonic nde. *Sensors*, 21(15):5077, 2021.

[170] Roney A Villaviencio et al. Generation of trajectories-robot by off-line programming to obtain high quality mag welding beads. In *2020 IEEE ANDESCON*, pages 1–6. IEEE, 2020.

[171] Johannes Vrana, Norbert Meyendorf, Nathan Ida, and Ripudaman Singh. Introduction to nde 4.0. *Handbook of Nondestructive Evaluation 4.0*, pages 1–28, 2021.

[172] Johanna Wallén. *The history of the industrial robot*. Linköping University Electronic Press, 2008.

[173] Brandon Z Wang, Christopher PL Barkan, and M Rapik Saat. Quantitative analysis of changes in freight train derailment causes and rates. *Journal of transportation engineering, Part A: Systems*, 146(11):04020127, 2020.

[174] Jiexin Wang, Stefan Elfwing, and Eiji Uchibe. Modular deep reinforcement learning from reward and punishment for robot navigation. *Neural Networks*, 135:115–126, 2021.

[175] Lihui Wang and Wei Ji. Cloud enabled cps and big data in manufacturing. In *International Conference on the Industry 4.0 model for Advanced Manufacturing*, pages 265–292. Springer, 2018.

[176] Zheng Ju Wang and Ming Rui Chen. Implementation of shading techniques based on opengl. In *Applied Mechanics and Materials*, volume 577, pages 1038–1042. Trans Tech Publ, 2014.

[177] Office of Energy Efficiency Water Power Technologies Office and Renewable Energy. Types of hydropower turbines.

[178] Eric W Weisstein. Unit vector. *https://mathworld. wolfram. com/*.

[179] Eric W Weisstein. Circular segment. *https://mathworld. wolfram. com/*, 2002.

[180] Eric W Weisstein. Normal vector. *https://mathworld. wolfram. com/*, 2002.

[181] Eric W Weisstein. Rotation matrix. *https://mathworld. wolfram. com/*, 2003.

[182] Eric W Weisstein. Matrix inverse. *https://mathworld. wolfram. com/*, 2014.

[183] L Xiang, S Dixon, CB Thring, Z Li, and RS Edwards. Lift-off performance of electromagnetic acoustic transducers (emats) for surface acoustic wave generation. *NDT & E International*, 126:102576, 2022.

[184] Zhe Xu and Yuanjiang Liu. Abb robotic arm offline programming system. In *Journal of Physics: Conference Series*, volume 1267, page 012064. IOP Publishing, 2019.

[185] Yijun Yan, Jinchang Ren, Huan Zhao, James FC Windmill, Winifred Ijomah, Jesper De Wit, and Justus Von Freeden. Non-destructive testing of composite fiber materials with hyperspectral imaging—evaluative studies in the eu h2020 fibreeuse project. *IEEE Transactions on Instrumentation and Measurement*, 71:1–13, 2022.

[186] Siril Yella, MS Dougherty, and NK Gupta. Artificial intelligence techniques for the automatic interpretation of data from non-destructive testing. *Insight-Non-Destructive Testing and Condition Monitoring*, 48(1):10–20, 2006.

[187] Leijian Yu, Erfu Yang, Peng Ren, Cai Luo, Gordon Dobie, Dongbing Gu, and Xiutian Yan. Inspection robots in oil and gas industry: a review of current solutions and future trends. In *2019 25th International Conference on Automation and Computing (ICAC)*, pages 1–6. IEEE, 2019.

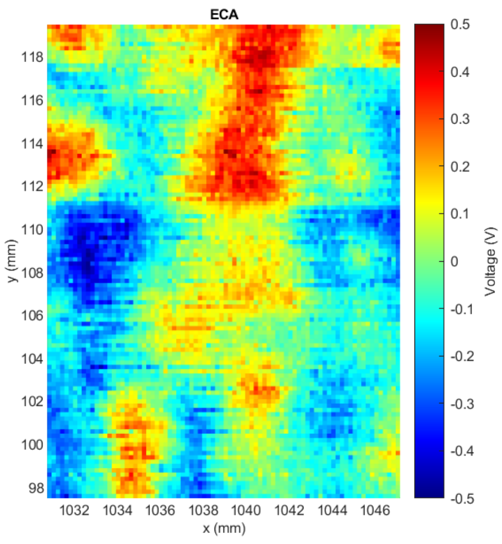
[188] Weipeng Zhang, Chenglong Wang, Fengqin Xie, and Huayu Zhang. Defect imaging curved surface based on flexible eddy current array sensor. *Measurement*, 151:107280, 2020.

[189] Ji Zhou, Yanhong Zhou, Baicun Wang, and Jiyuan Zang. Human–cyber–physical systems (hcpss) in the context of new-generation intelligent manufacturing. *Engineering*, 5(4):624–636, 2019.

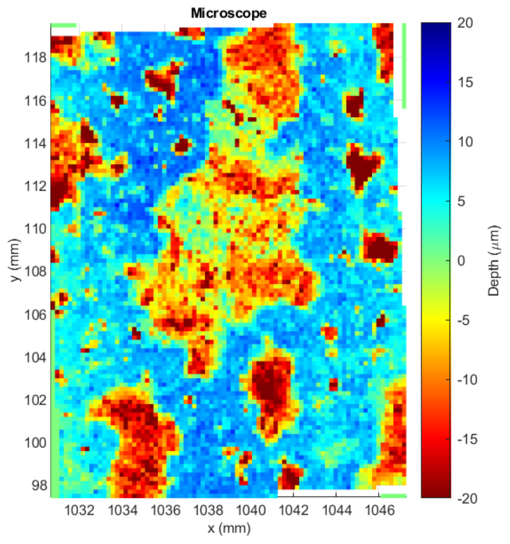
[190] Olivier Zunz. *Making America Corporate, 1870-1920.* University of Chicago Press, 1992.

APPENDIX

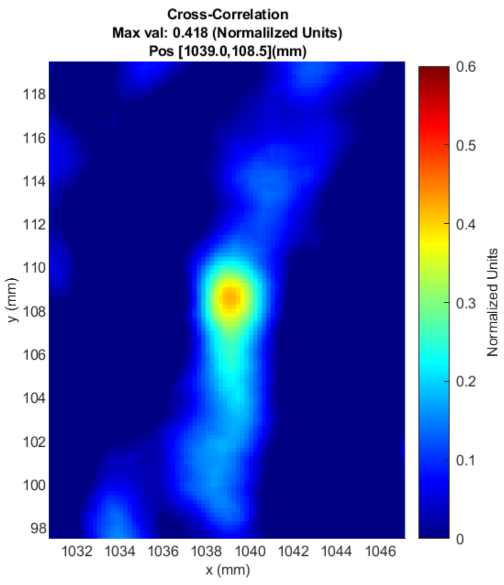
Below covers the bulk defect information, showing results for ECA, VHX, xcor, and intersections for each defect, coil 16 only, for full and fast scans.



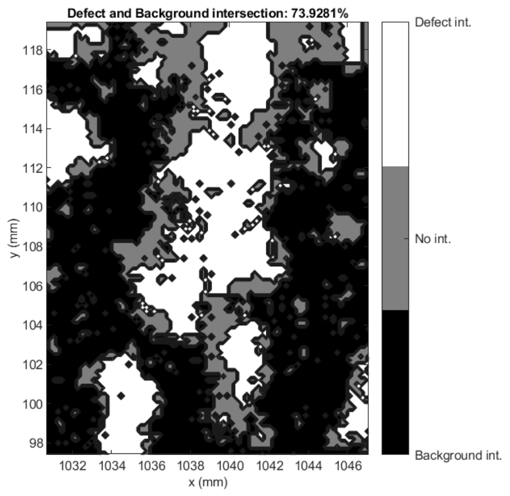
(a) ECA



(b) VHX

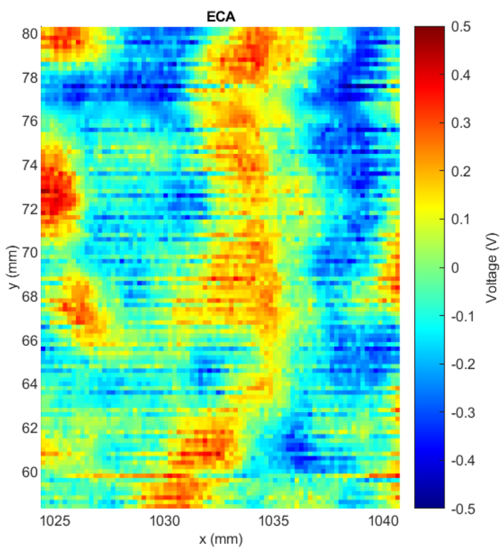


(c) Xcor

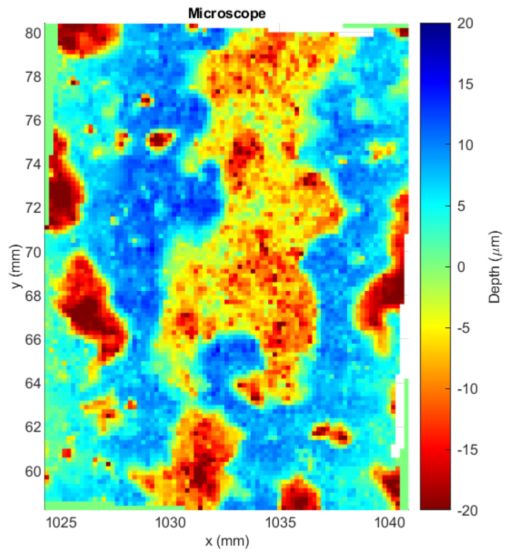


(d) Intersection

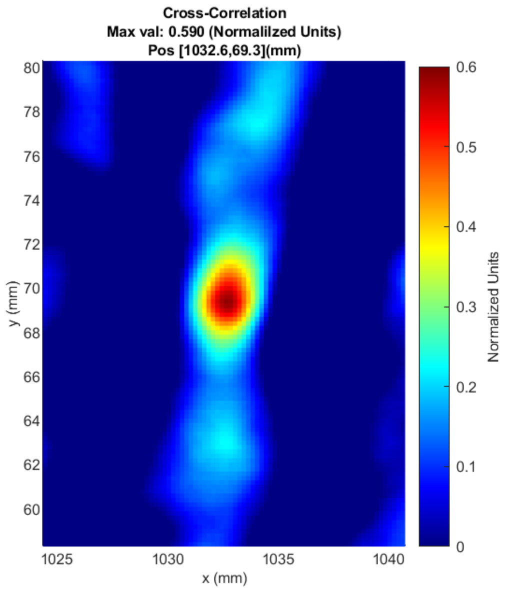
Figure 1 Flat sample a, defect D1, full scan.



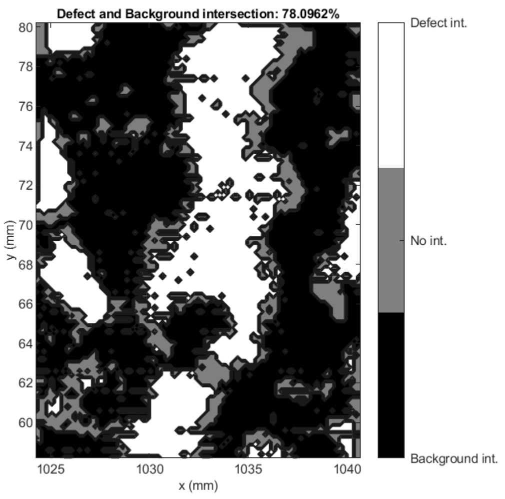
(a) ECA



(b) VHX

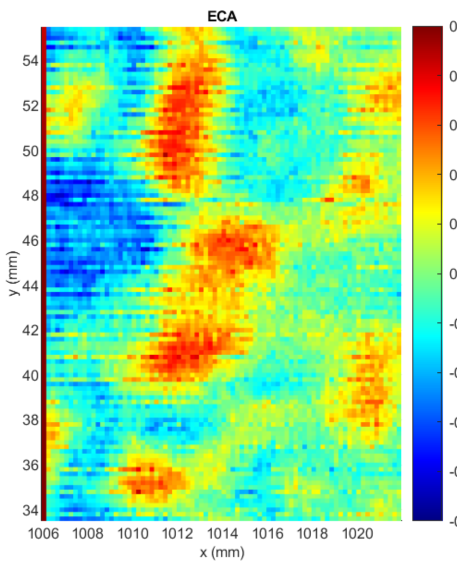


(c) Xcor

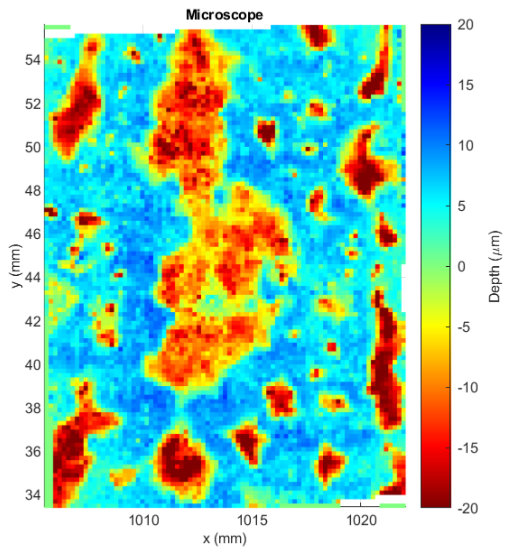


(d) Intersection

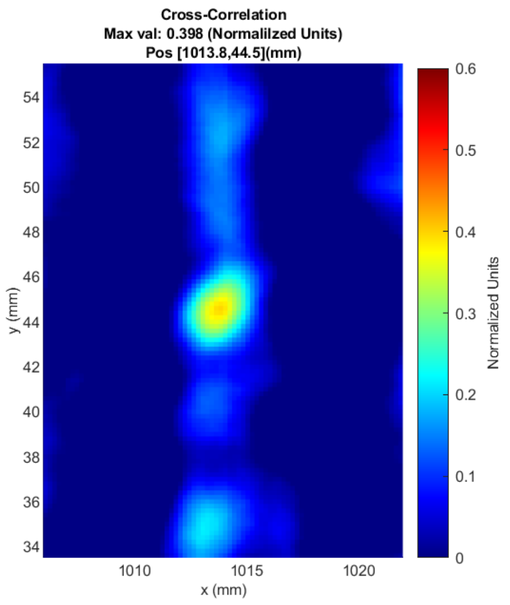
Figure 2 Flat sample a, defect D2, full scan.



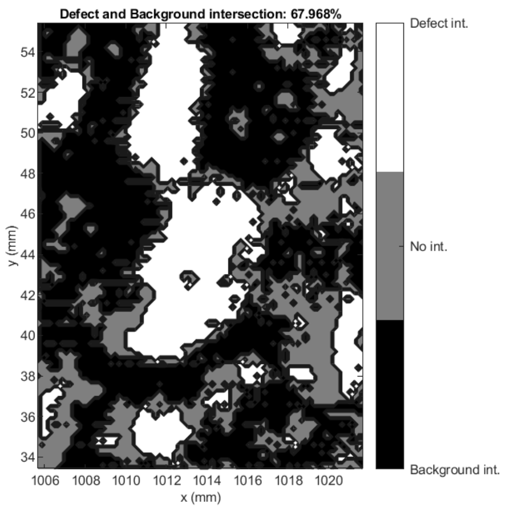
(a) ECA



(b) VHX

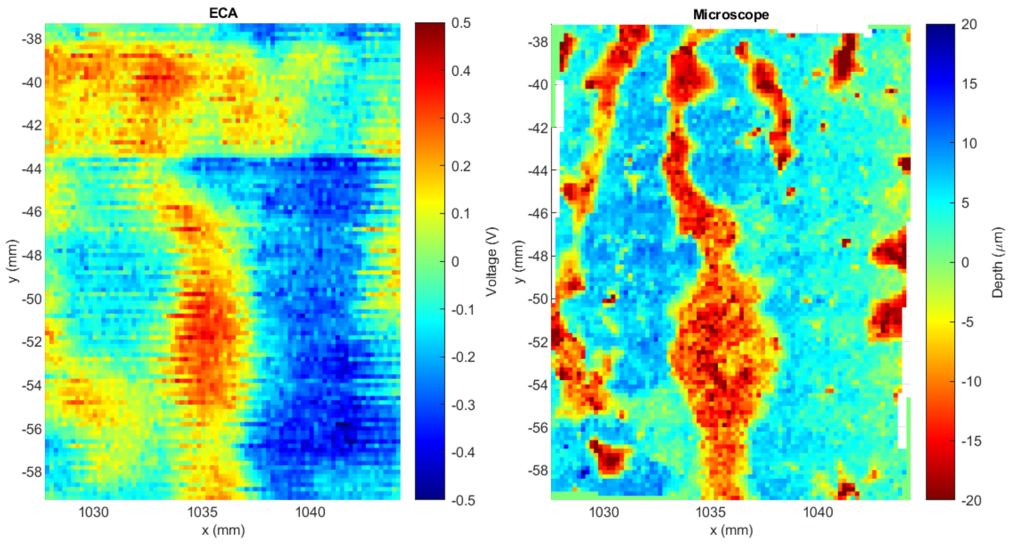


(c) Xcor

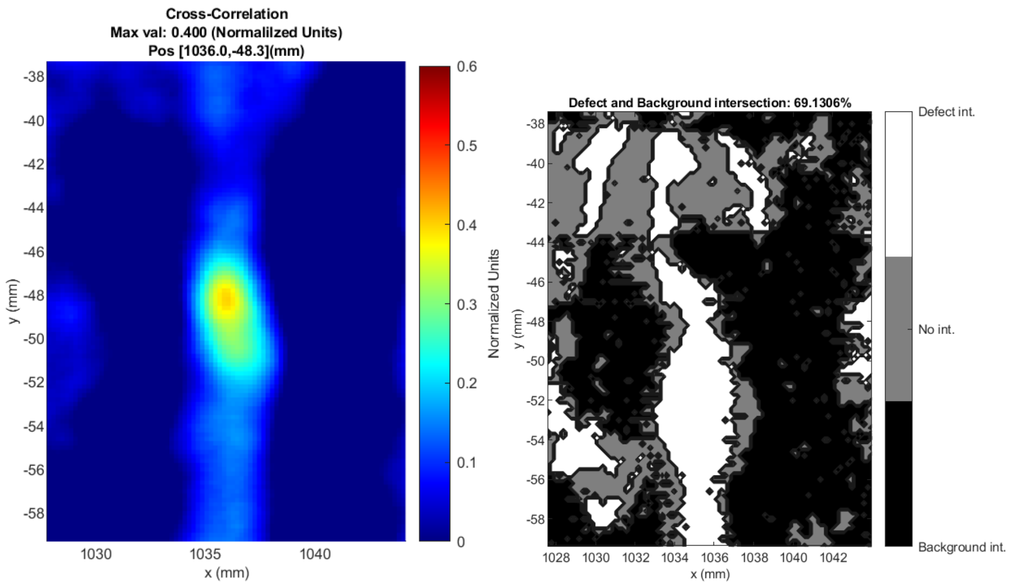


(d) Intersection

Figure 3 Flat sample a, defect D3, full scan.

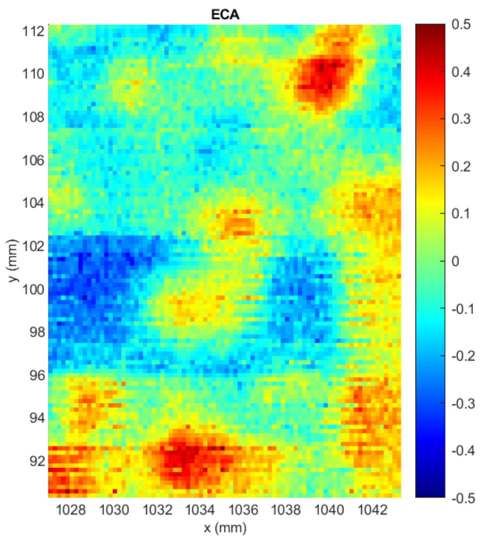


(a) ECA (b) VHX

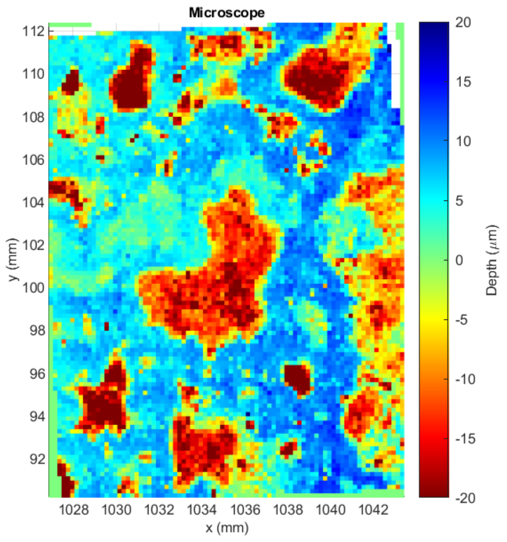


(c) Xcor (d) Intersection

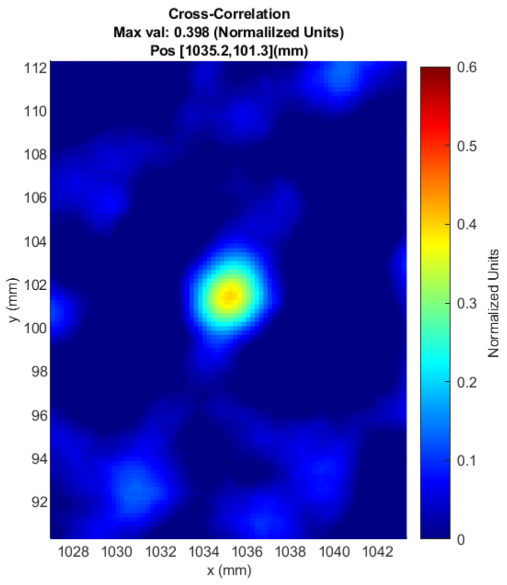
Figure 4 Flat sample a, defect D4, full scan.



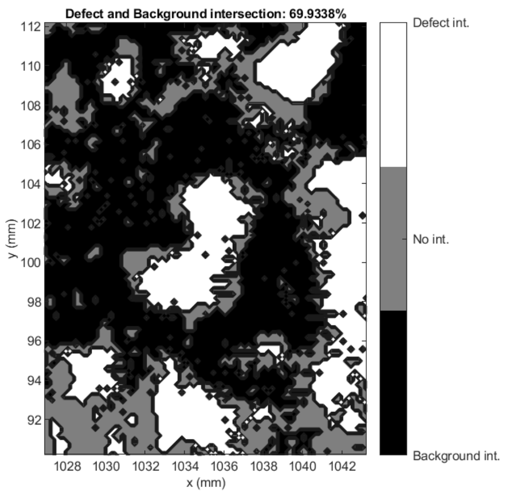
(a) ECA



(b) VHX

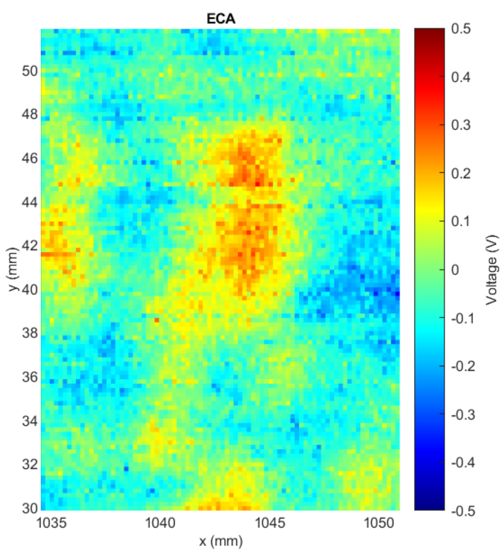


(c) Xcor

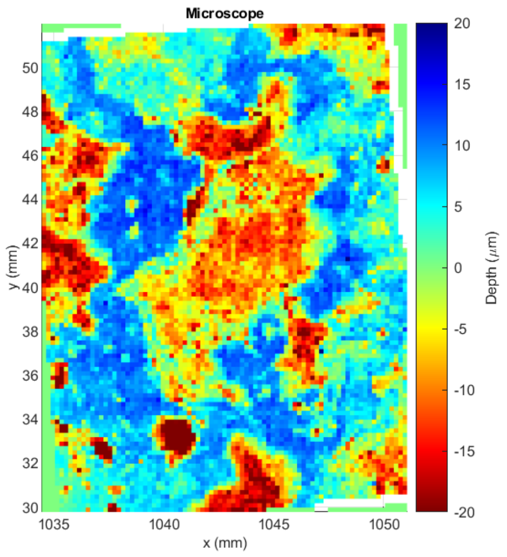


(d) Intersection

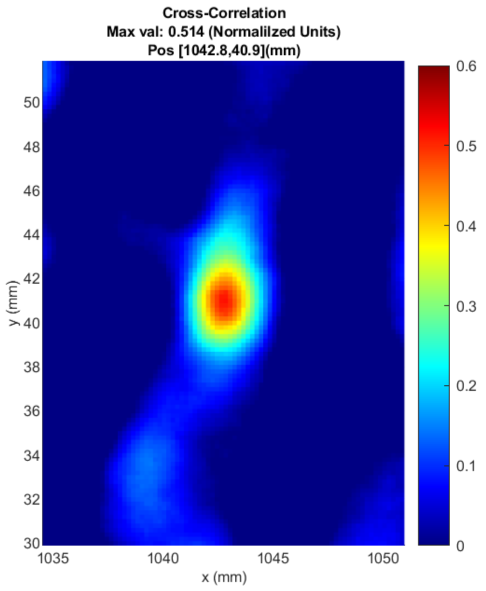
Figure 5 Concave sample b1, defect D1, full scan.



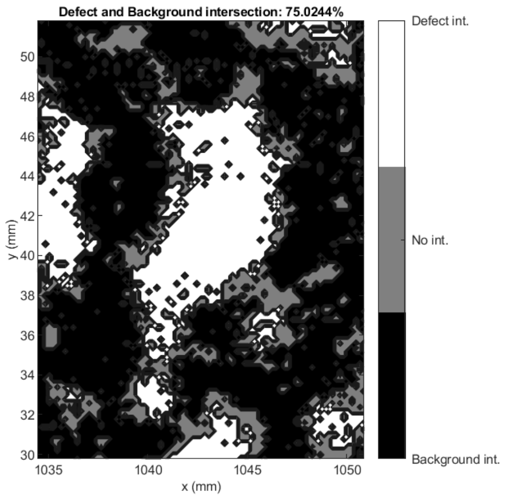
(a) ECA



(b) VHX

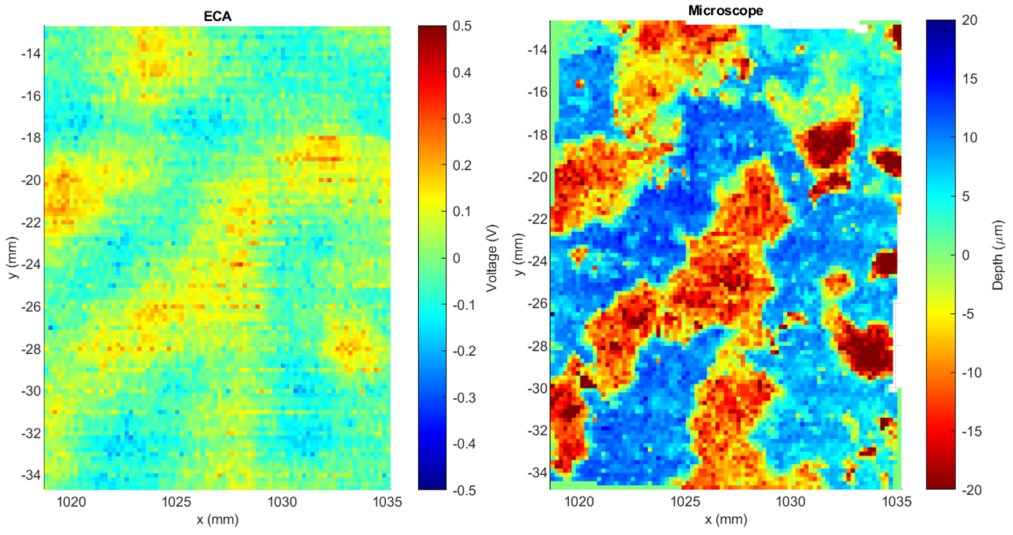


(c) Xcor

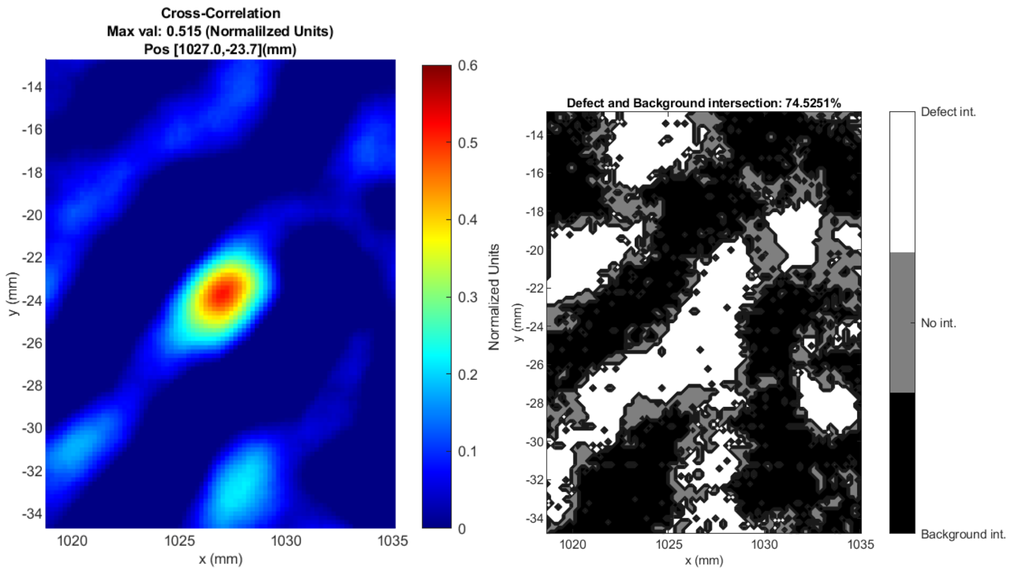


(d) Intersection

Figure 6 Concave sample b1, defect D2, full scan.

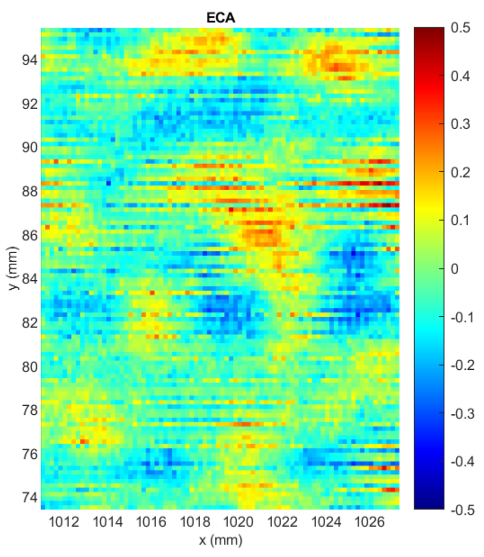


(a) ECA (b) VHX

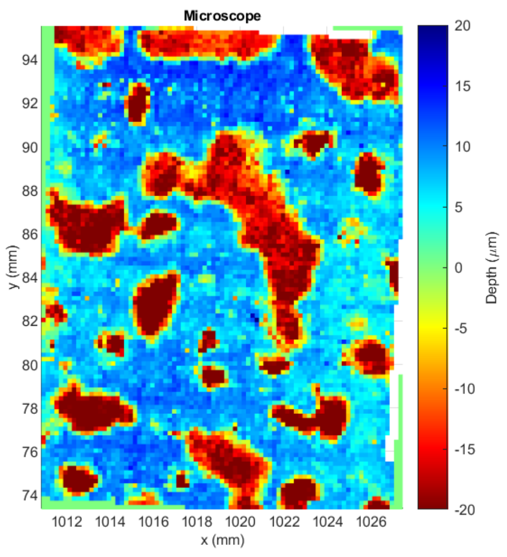


(c) Xcor (d) Intersection

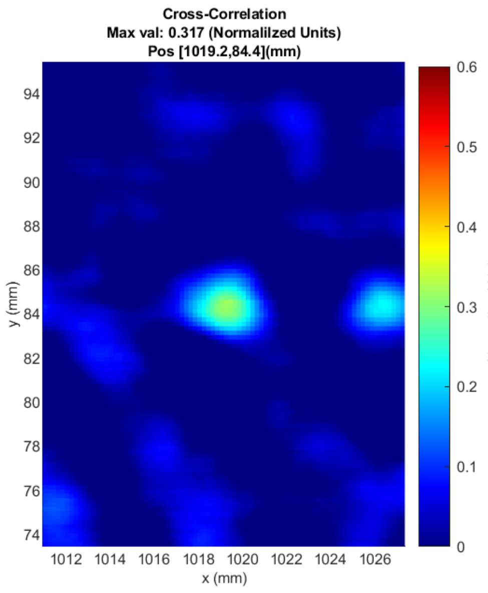
Figure 7 Concave sample b1, defect D3, full scan.



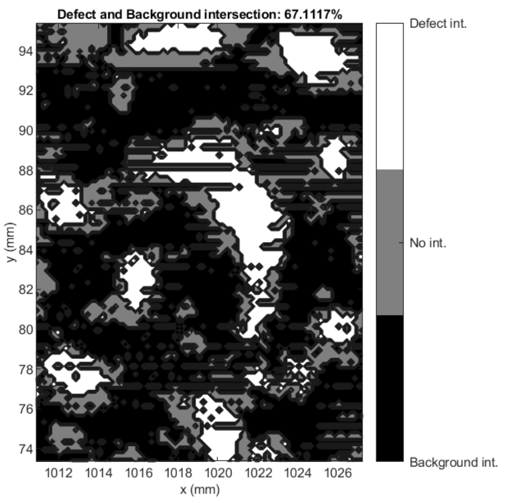
(a) ECA



(b) VHX

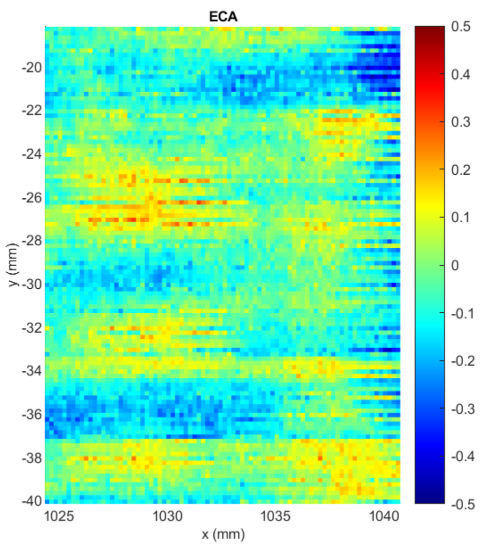


(c) Xcor

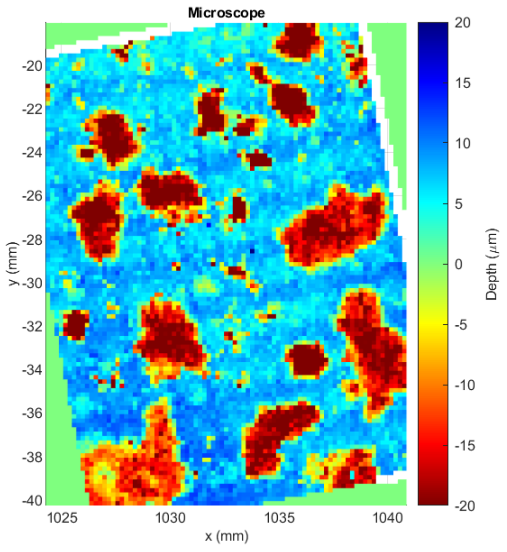


(d) Intersection

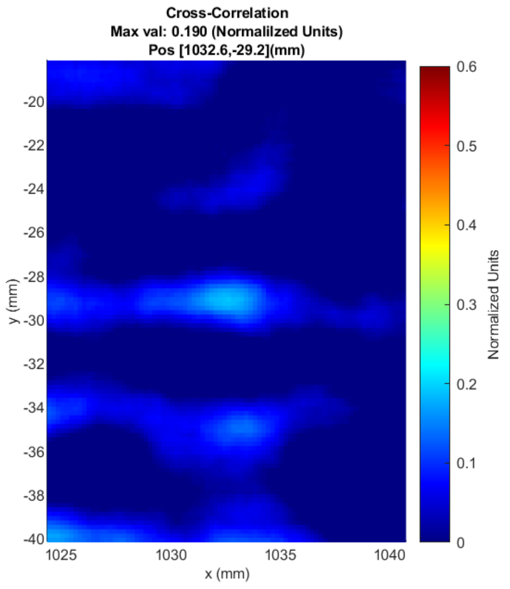
Figure 8 Concave sample b2, defect D1, full scan.



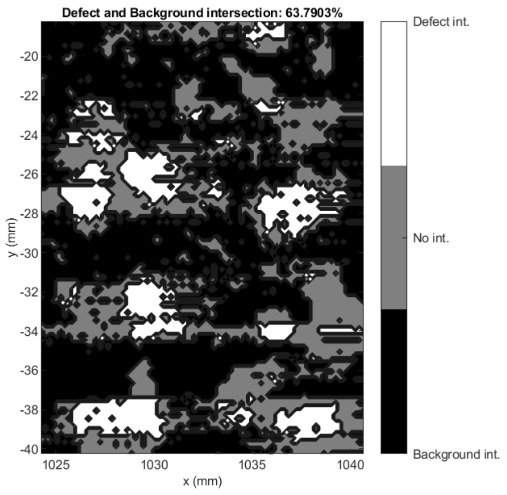
(a) ECA



(b) VHX

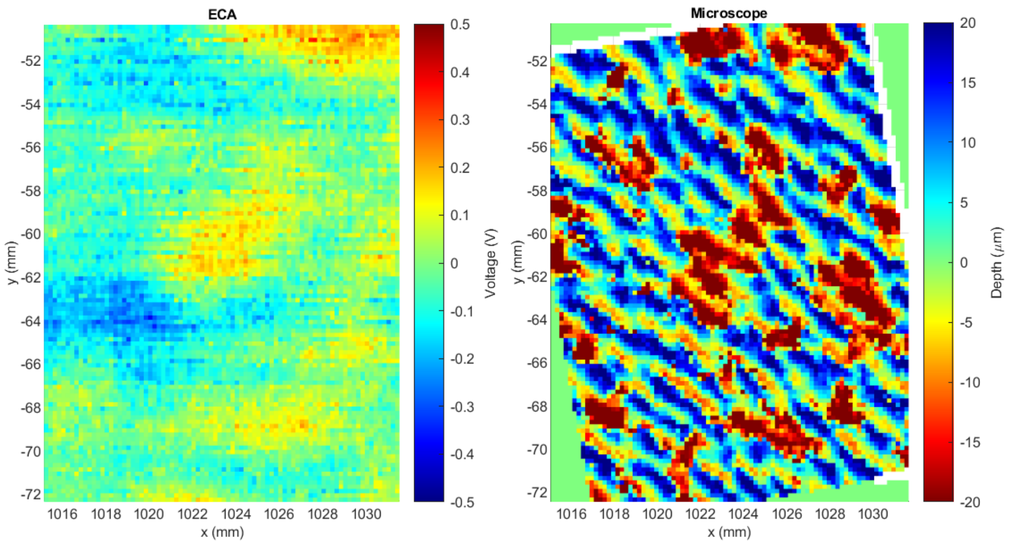


(c) Xcor



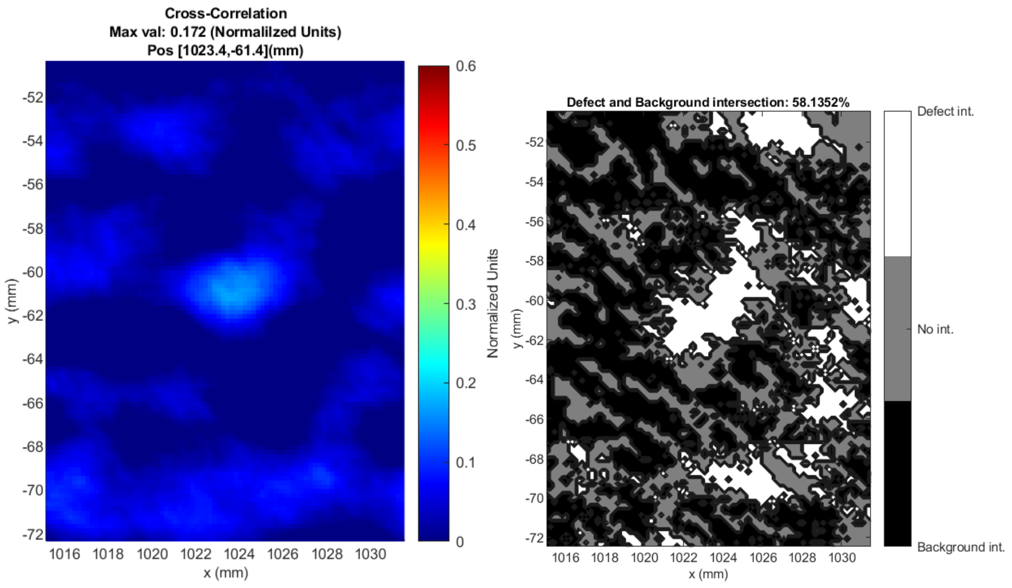
(d) Intersection

Figure 9 Concave sample b2, defect D2, full scan.



(a) ECA

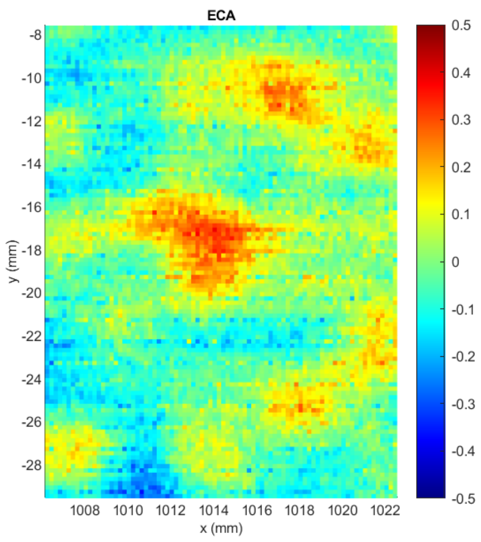
(b) VHX



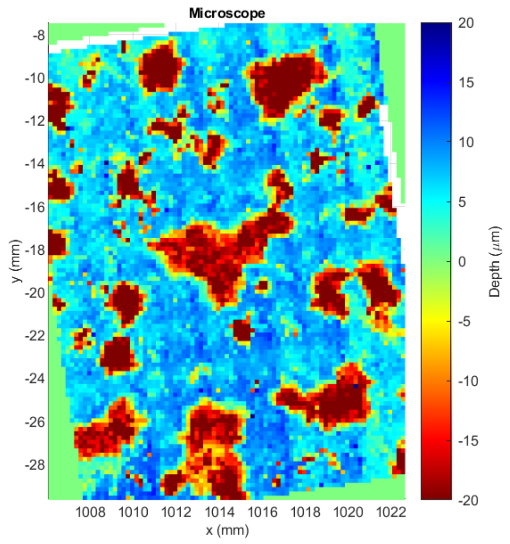
(c) Xcor

(d) Intersection

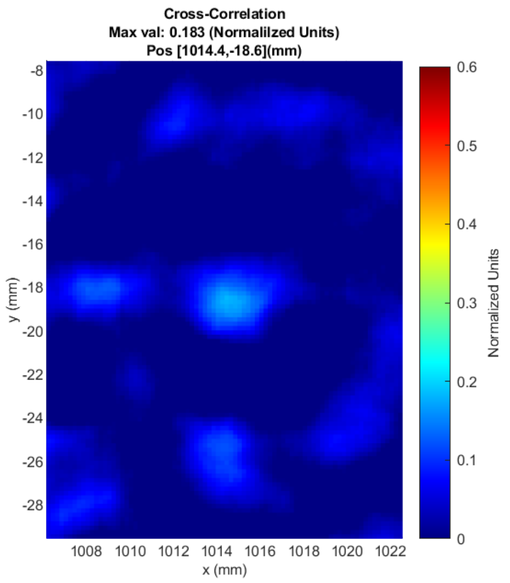
Figure 10 Concave sample b2, defect D3, full scan.



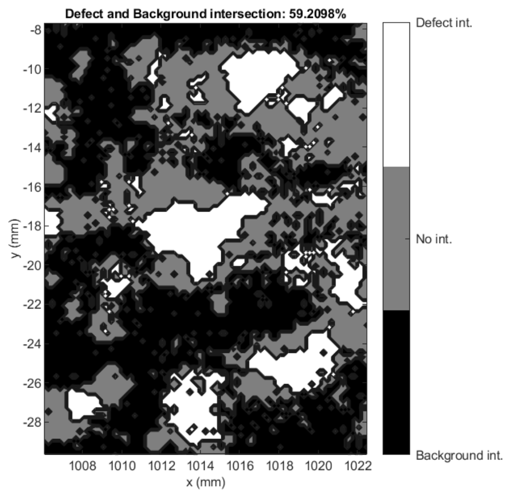
(a) ECA



(b) VHX

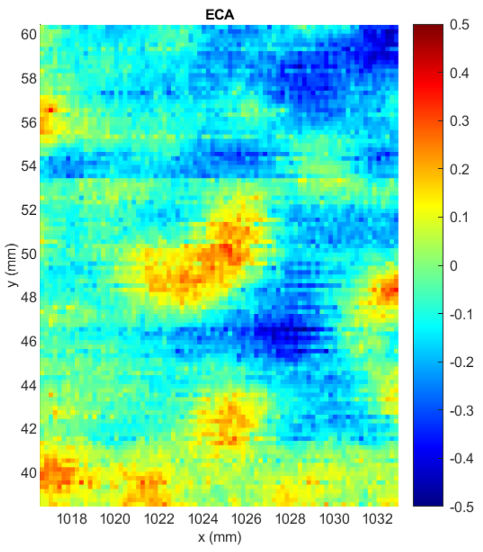


(c) Xcor

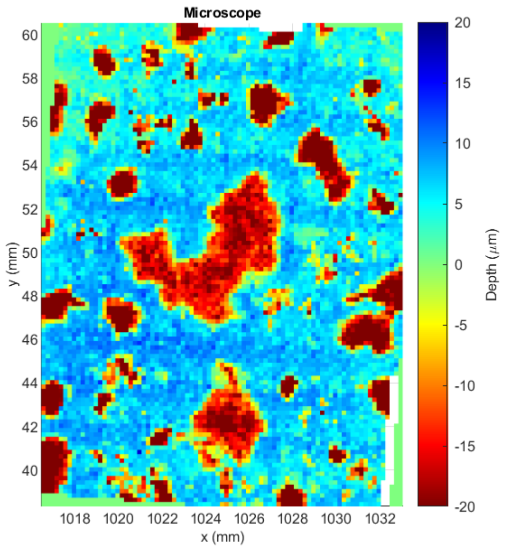


(d) Intersection

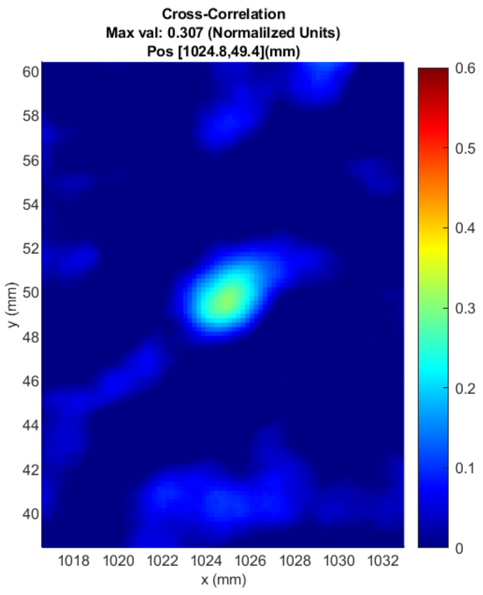
Figure 11 Concave sample b3, defect D1, full scan.



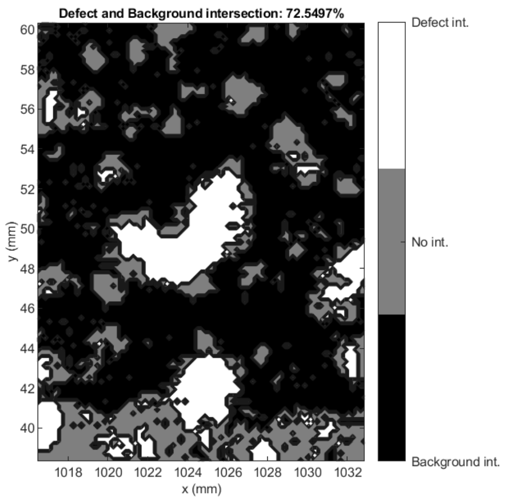
(a) ECA



(b) VHX

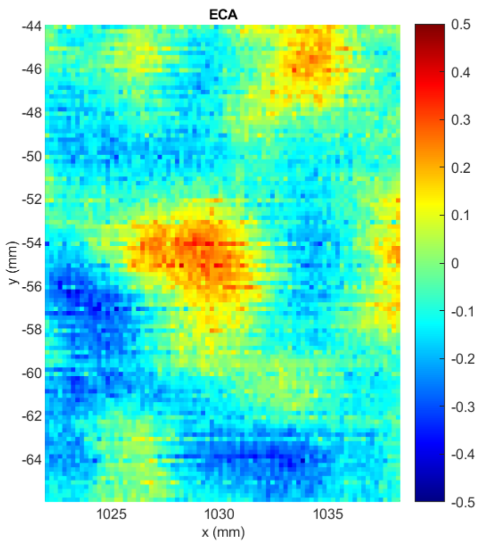


(c) Xcor

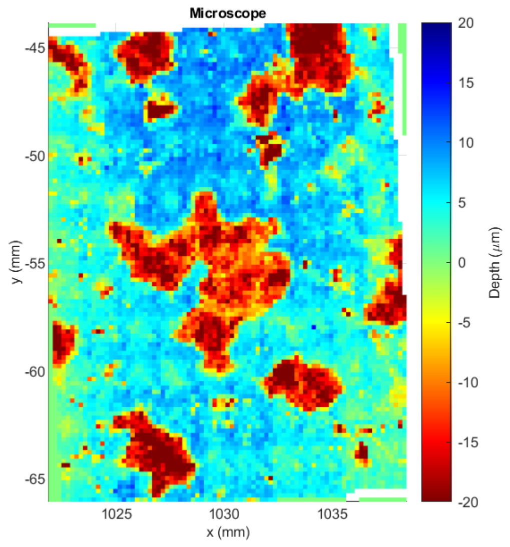


(d) Intersection

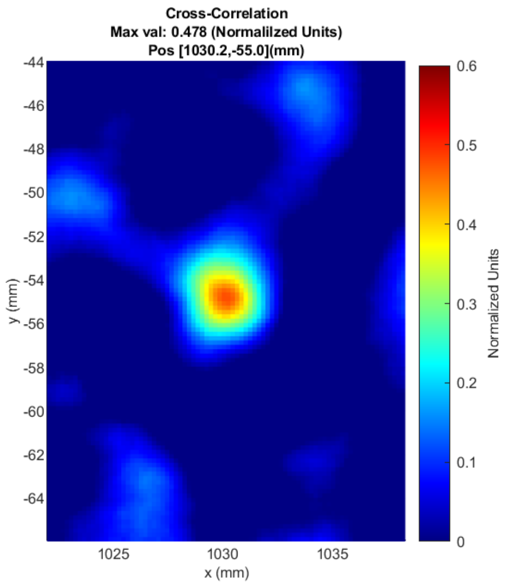
Figure 12 Concave sample b3, defect D2, full scan.



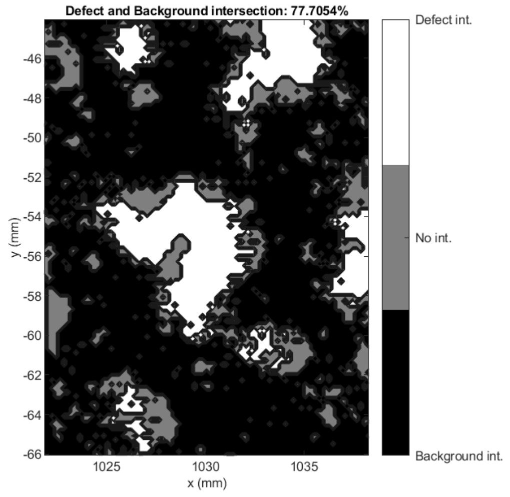
(a) ECA



(b) VHX

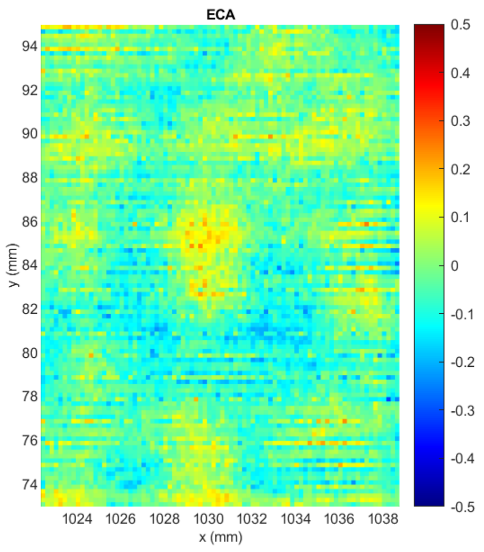


(c) Xcor

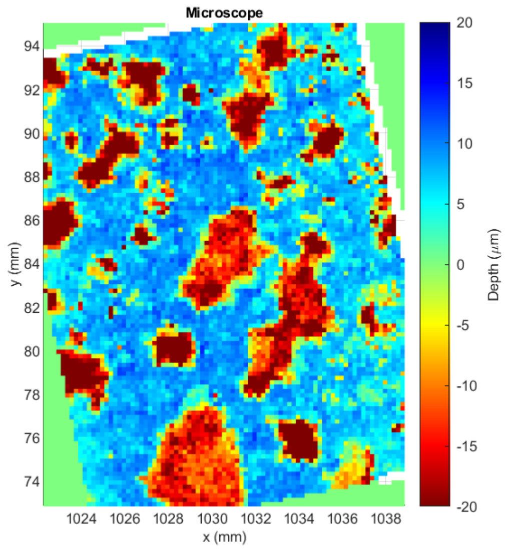


(d) Intersection

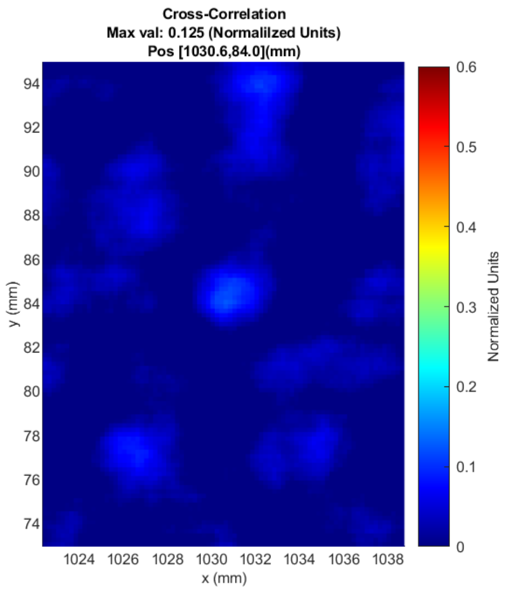
Figure 13 Concave sample b3, defect D3, full scan.



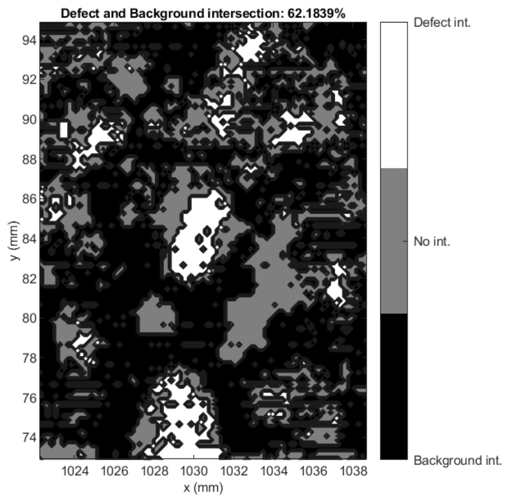
(a) ECA



(b) VHX

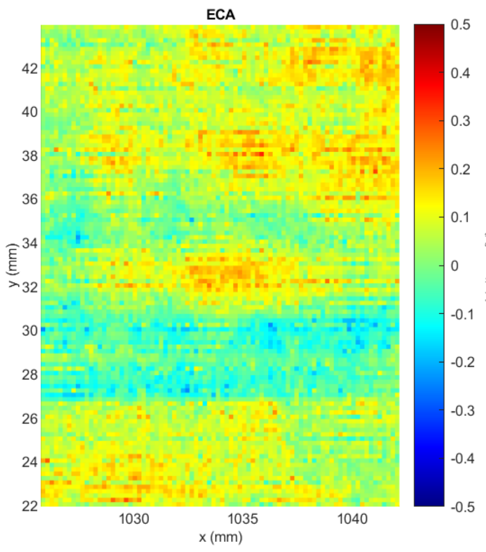


(c) Xcor

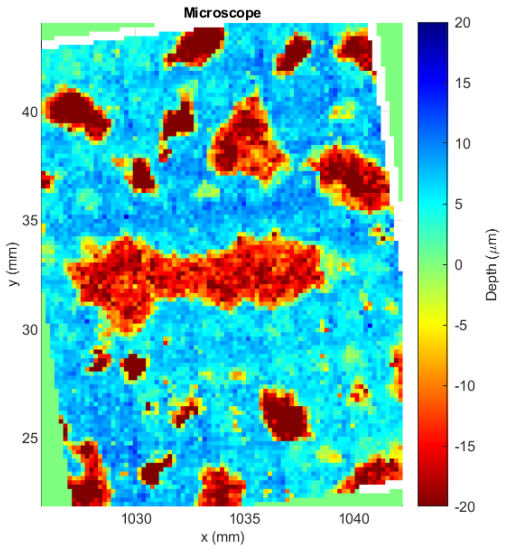


(d) Intersection

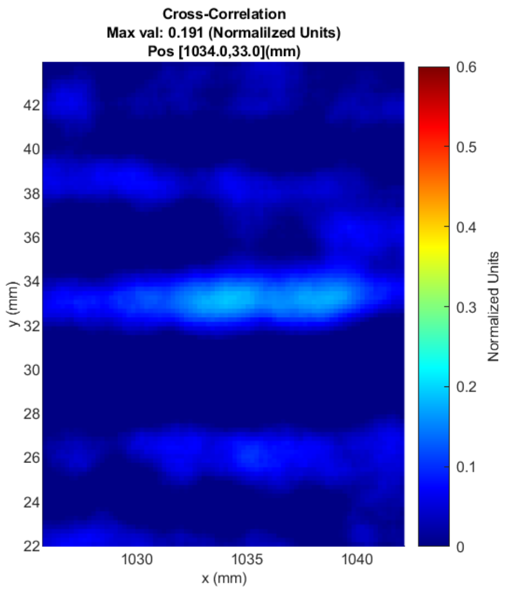
Figure 14 Concave sample b4, defect D1, full scan.



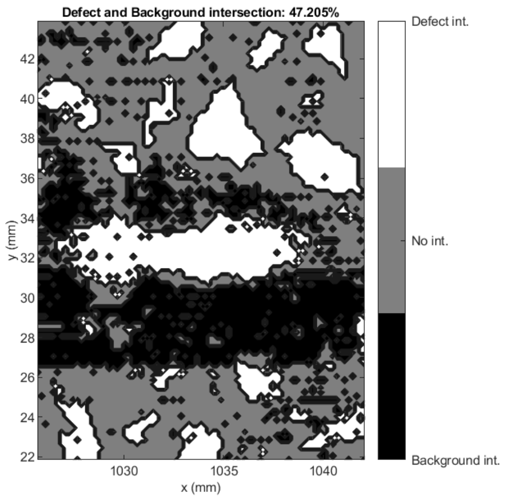
(a) ECA



(b) VHX

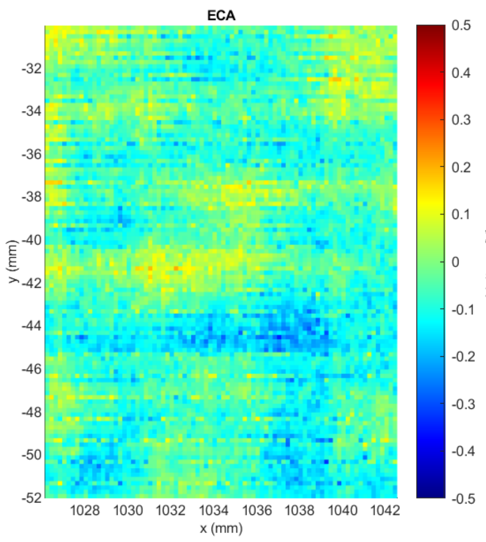


(c) Xcor

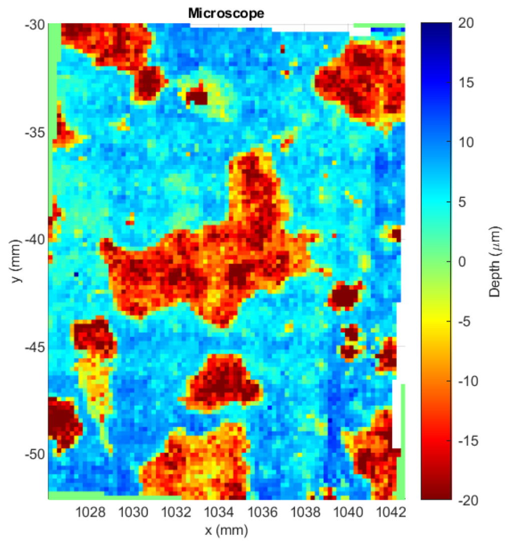


(d) Intersection

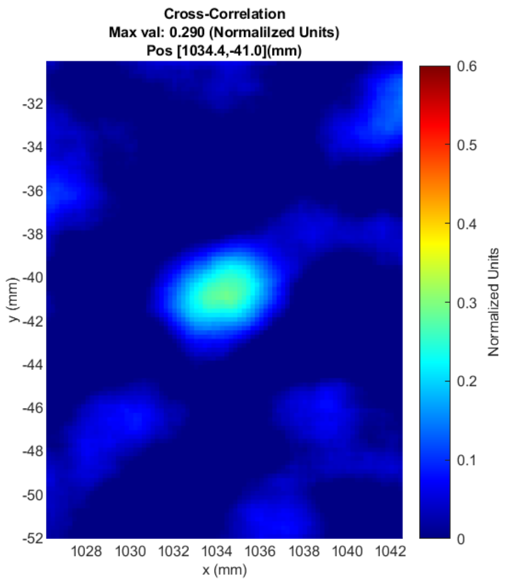
Figure 15 Concave sample b4, defect D2, full scan.



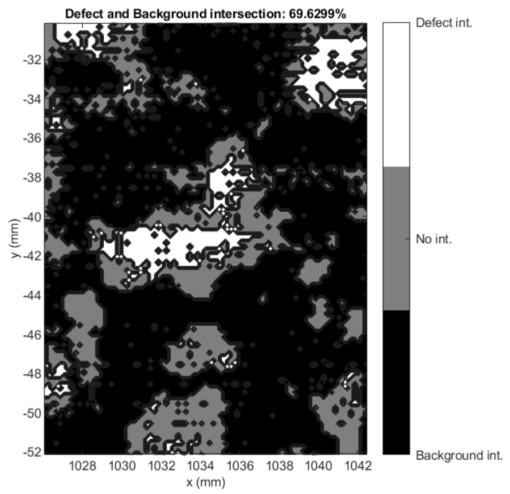
(a) ECA



(b) VHX

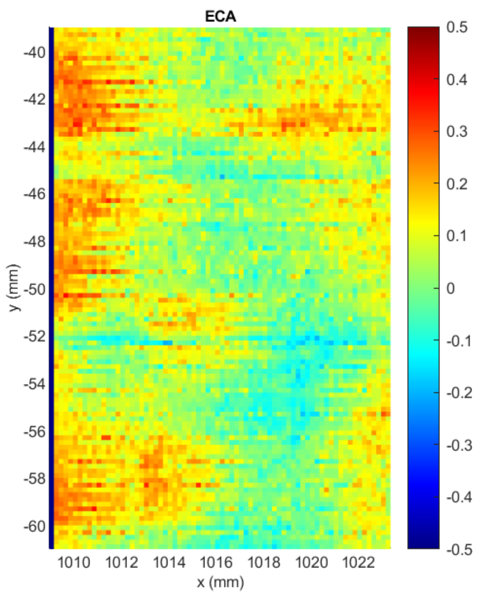


(c) Xcor

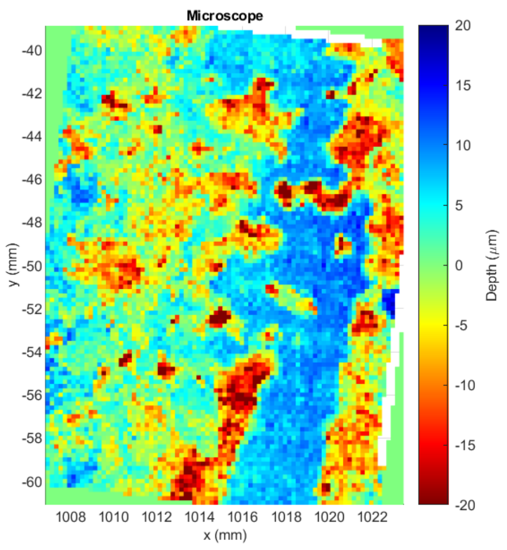


(d) Intersection

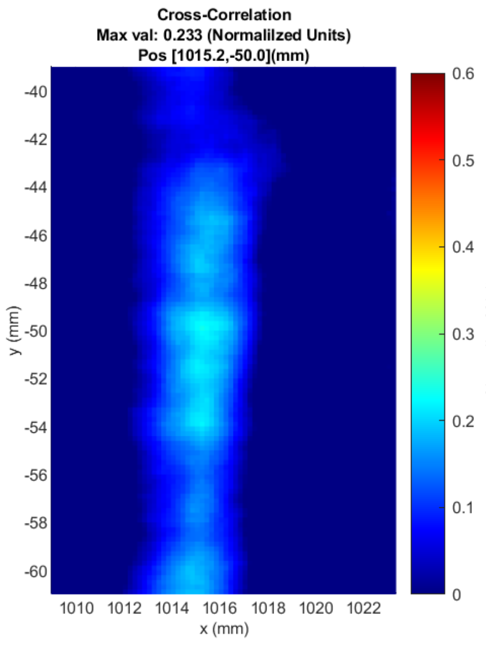
Figure 16 Concave sample b4, defect D3, full scan.



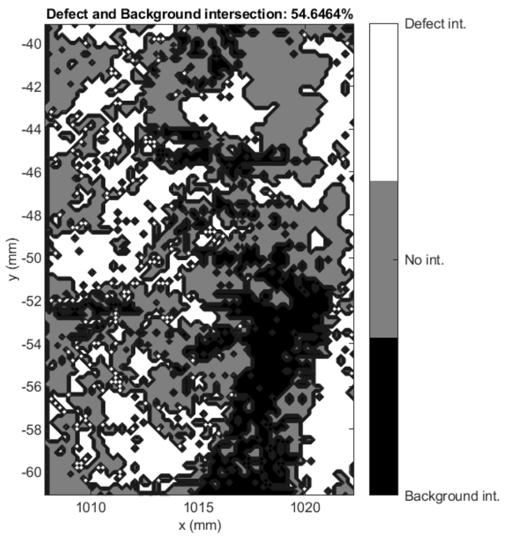
(a) ECA



(b) VHX

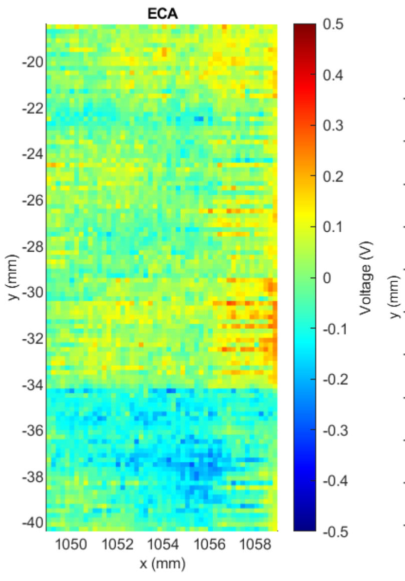


(c) Xcor

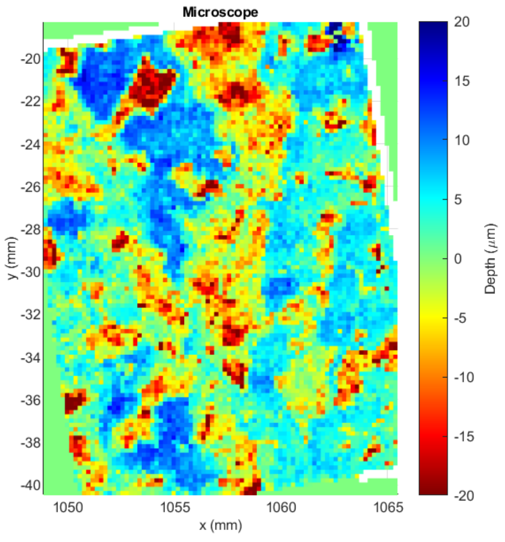


(d) Intersection

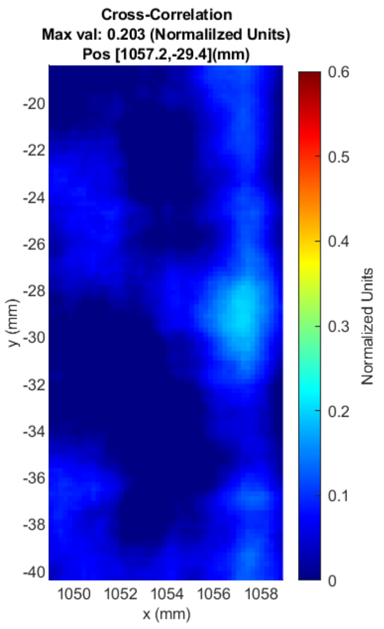
Figure 17 Convex sample c1, defect D1, full scan.



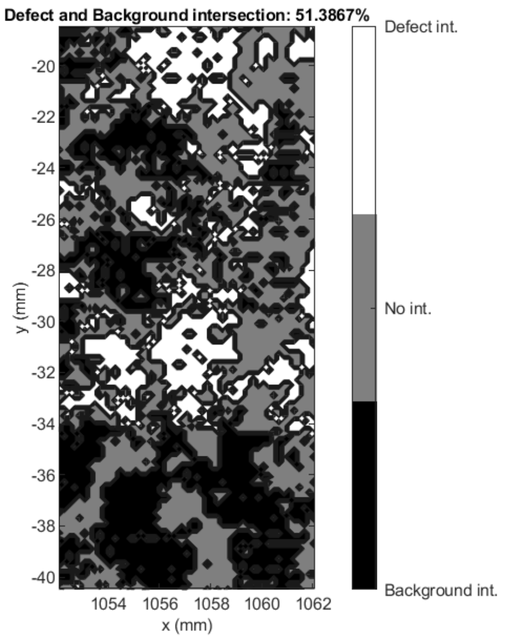
(a) ECA



(b) VHX

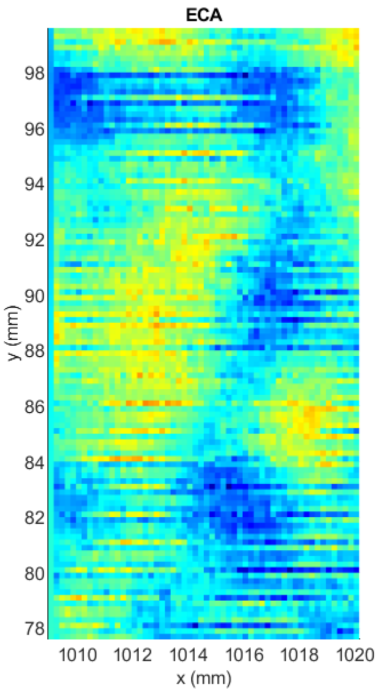


(c) Xcor

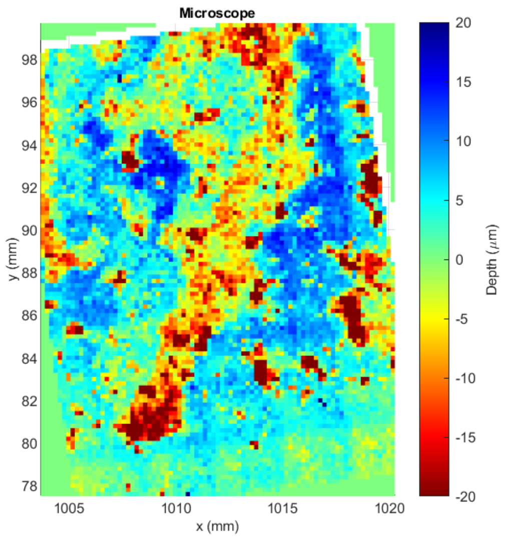


(d) Intersection

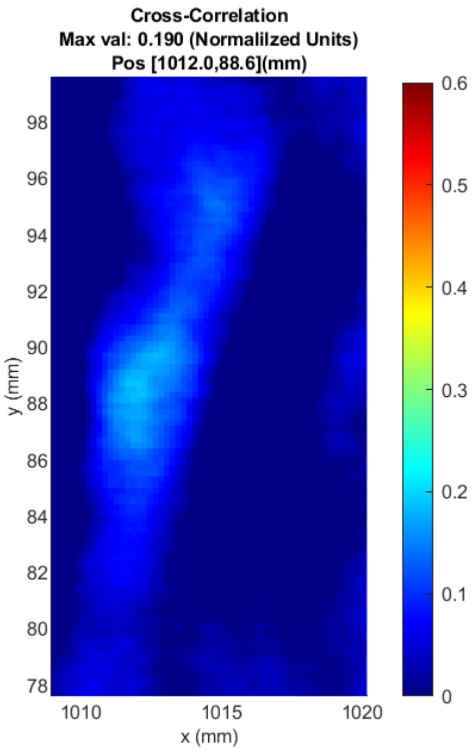
Figure 18 Convex sample c1, defect D2, full scan.



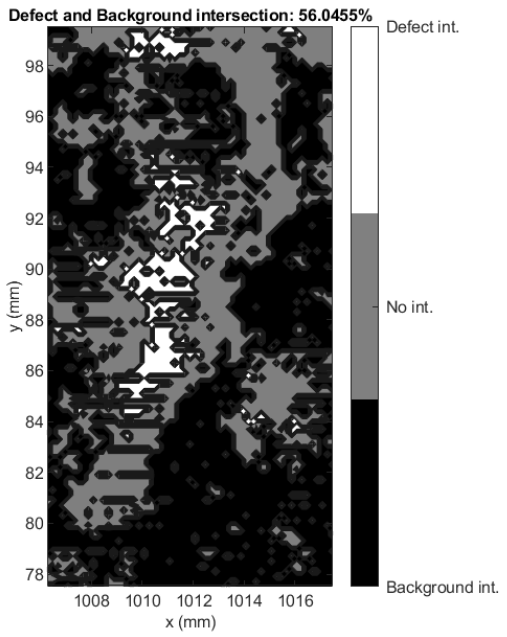
(a) ECA



(b) VHX

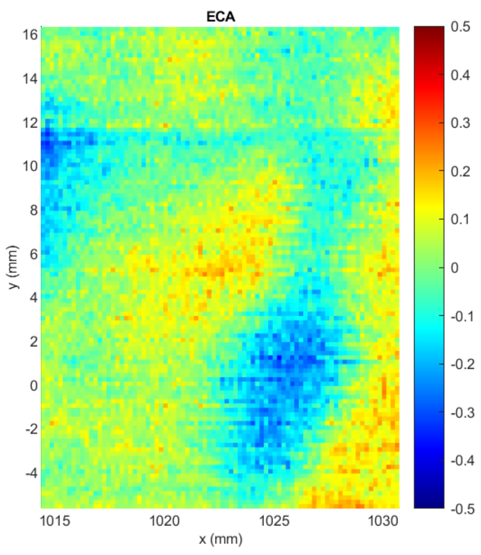


(c) Xcor

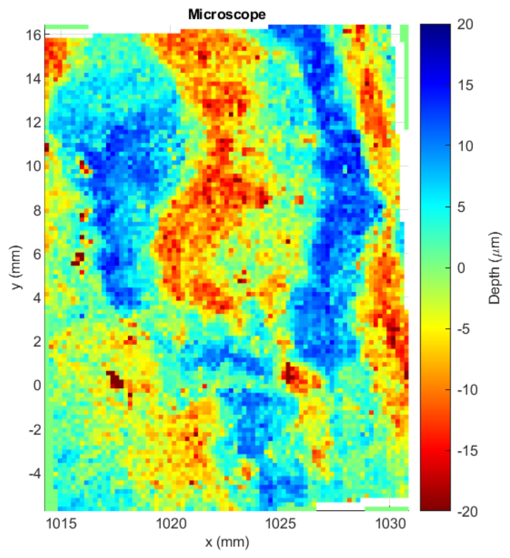


(d) Intersection

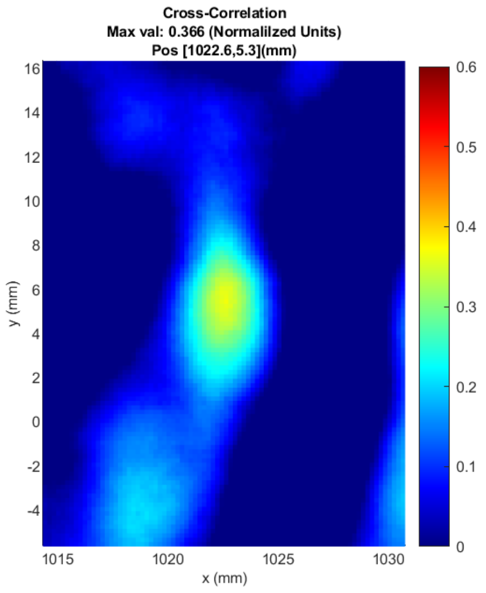
Figure 19 Convex sample c1, defect D3, full scan.



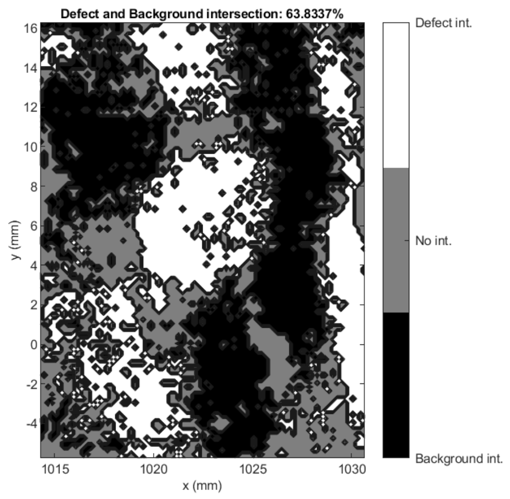
(a) ECA



(b) VHX

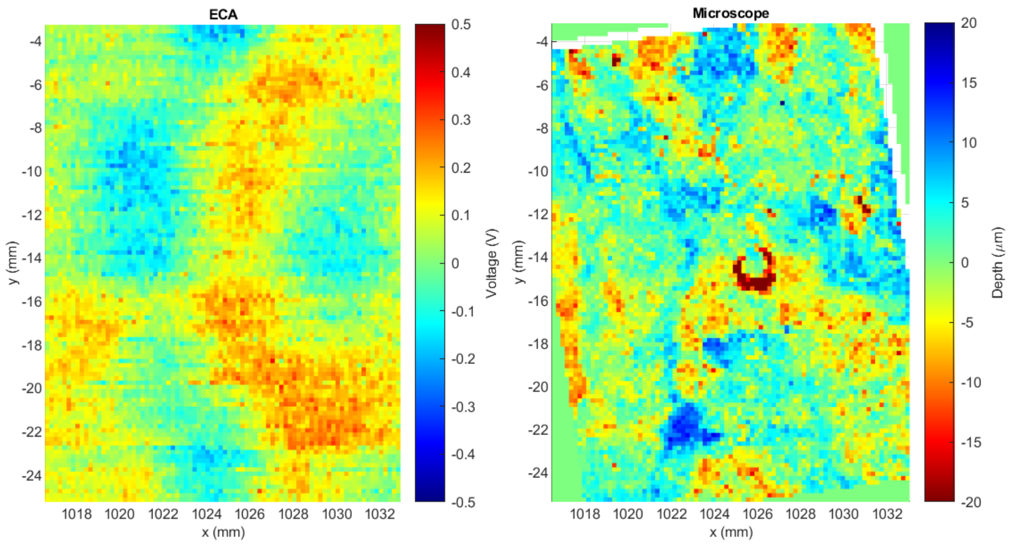


(c) Xcor

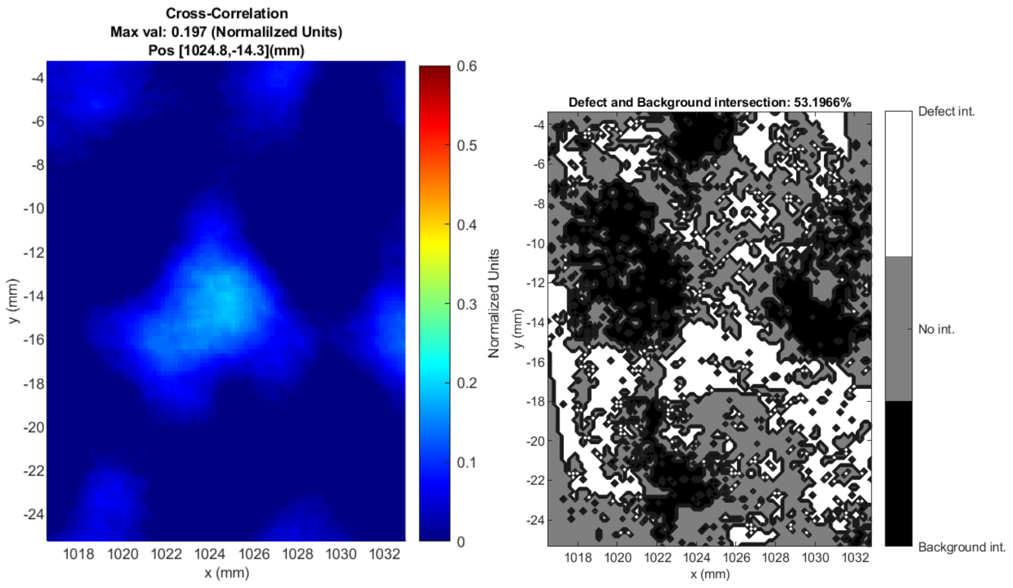


(d) Intersection

Figure 20 Convex sample c2, defect D1, full scan.

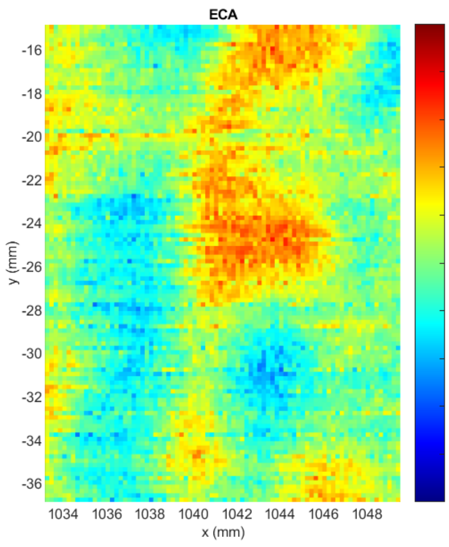


(a) ECA (b) VHX

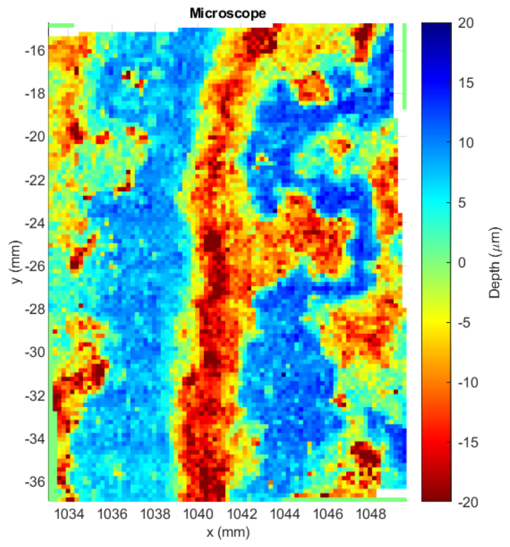


(c) Xcor (d) Intersection

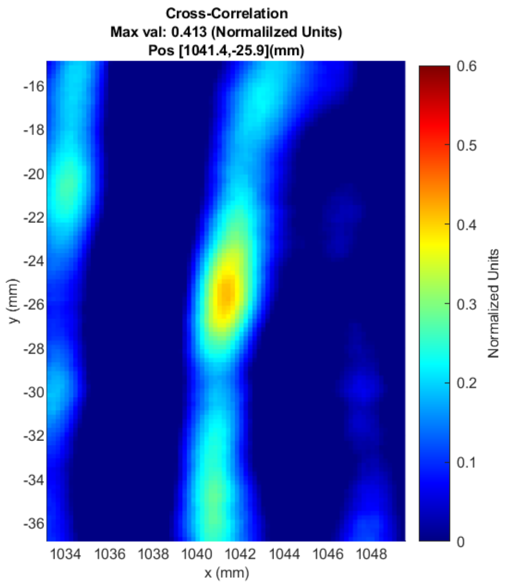
Figure 21 Convex sample c2, defect D2, full scan.



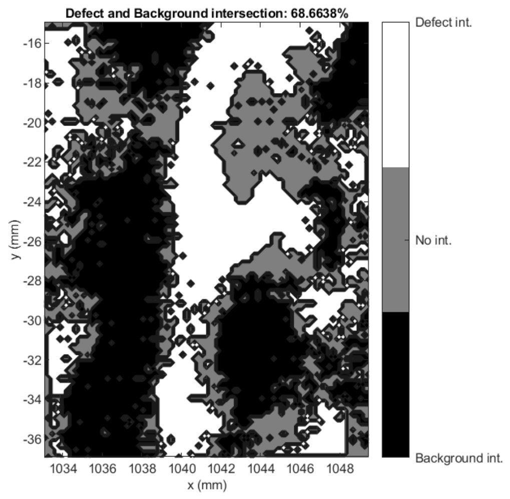
(a) ECA



(b) VHX

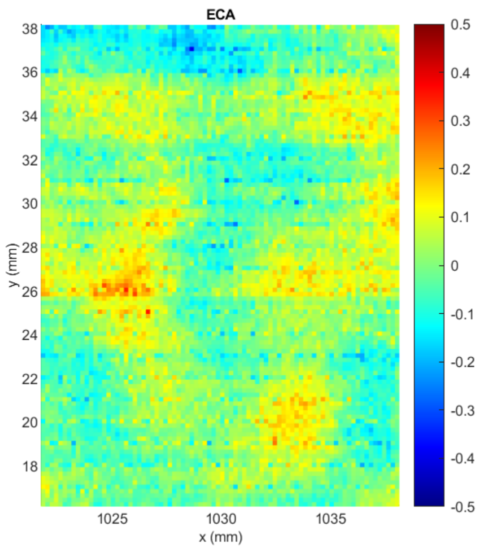


(c) Xcor

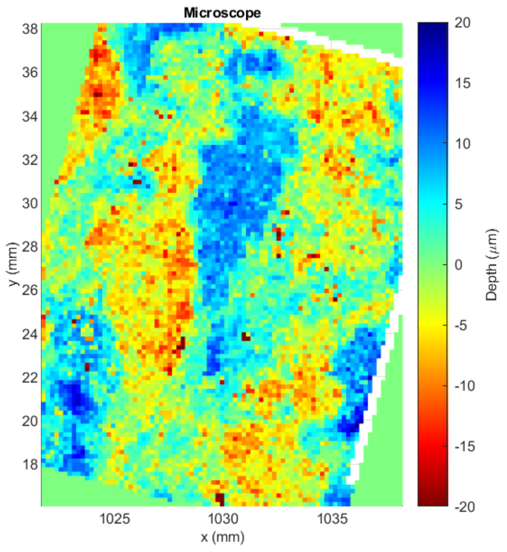


(d) Intersection

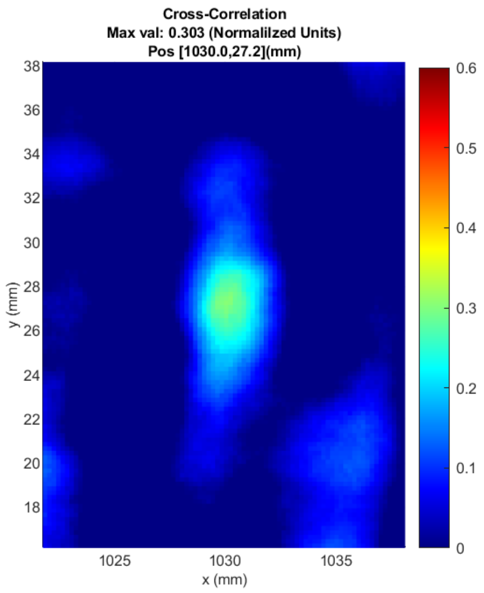
Figure 22 Convex sample c2, defect D3, full scan.



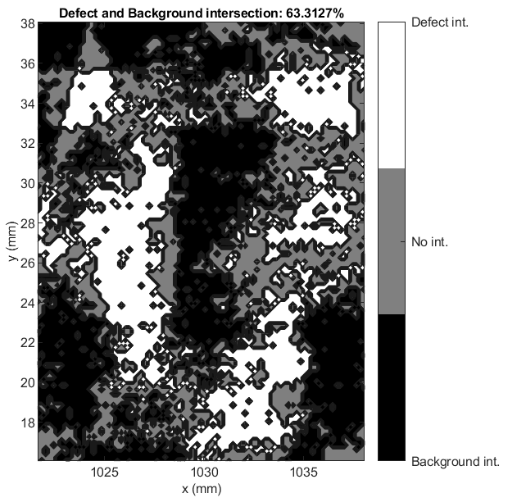
(a) ECA



(b) VHX

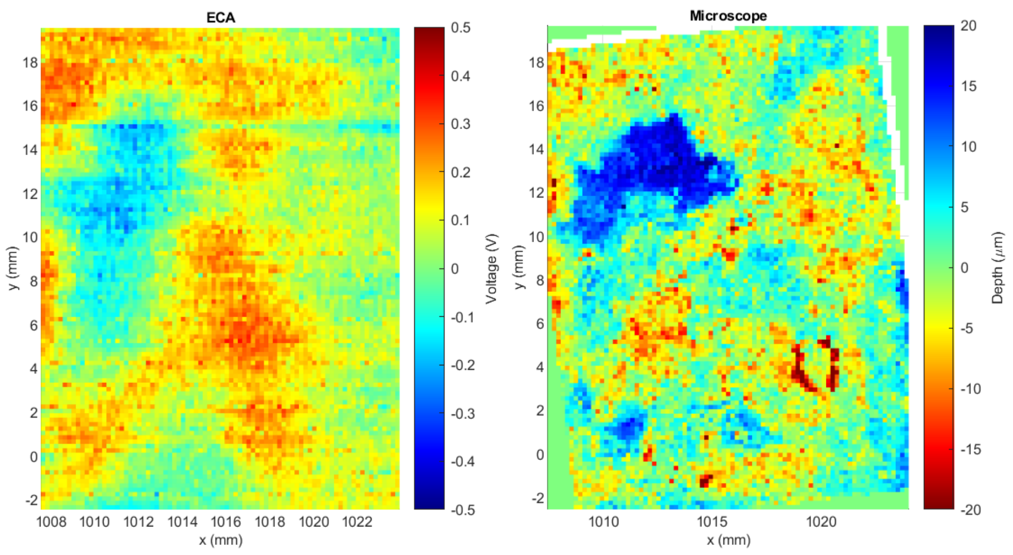


(c) Xcor

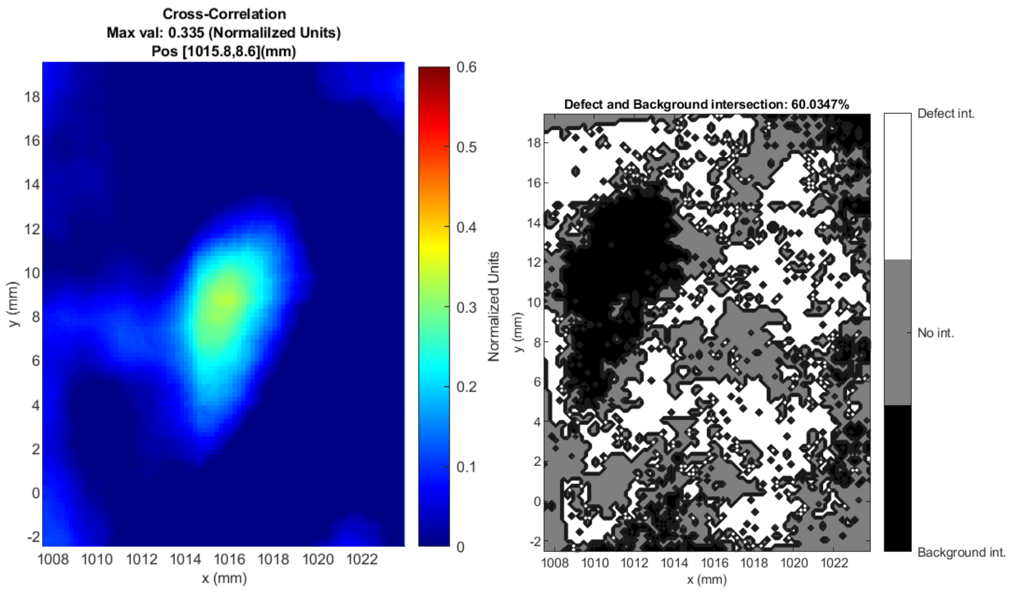


(d) Intersection

Figure 23 Convex sample c3, defect D1, full scan.

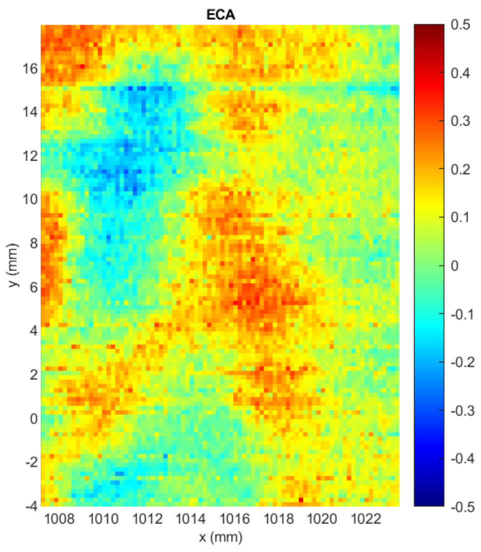


(a) ECA (b) VHX

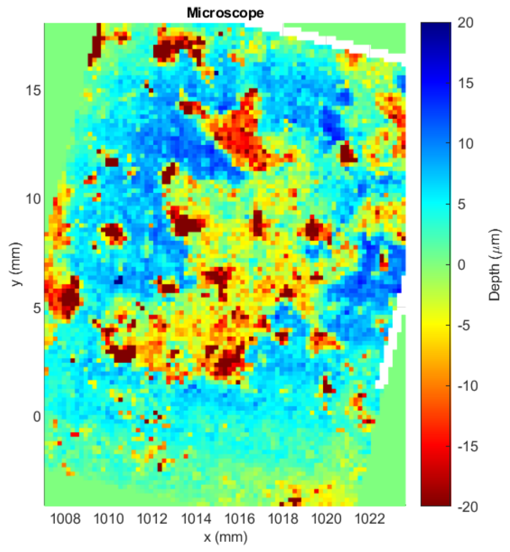


(c) Xcor (d) Intersection

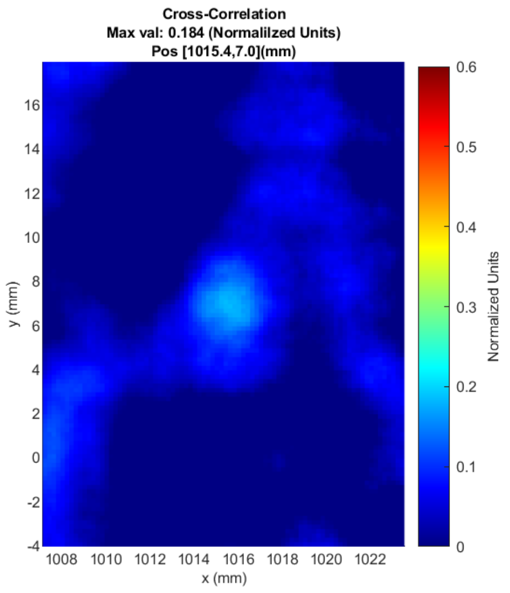
Figure 24 Convex sample c3, defect D2, full scan.



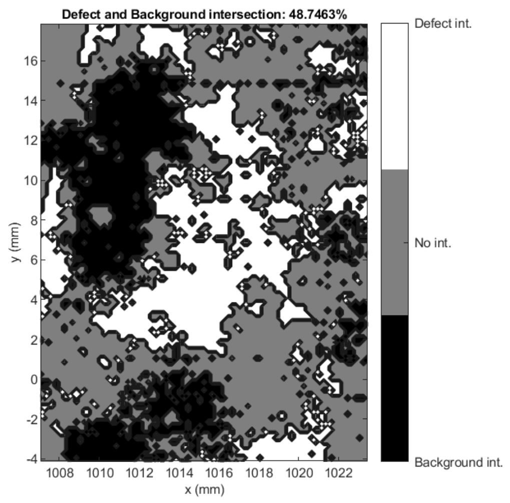
(a) ECA



(b) VHX

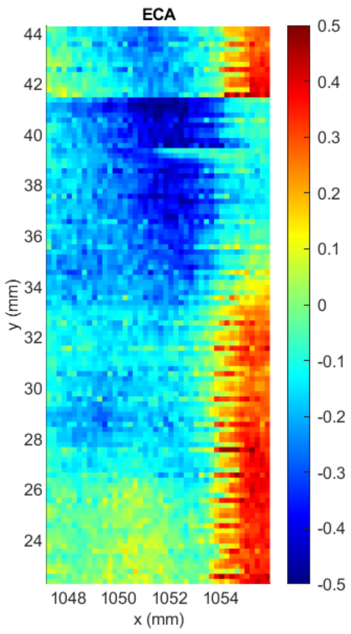


(c) Xcor

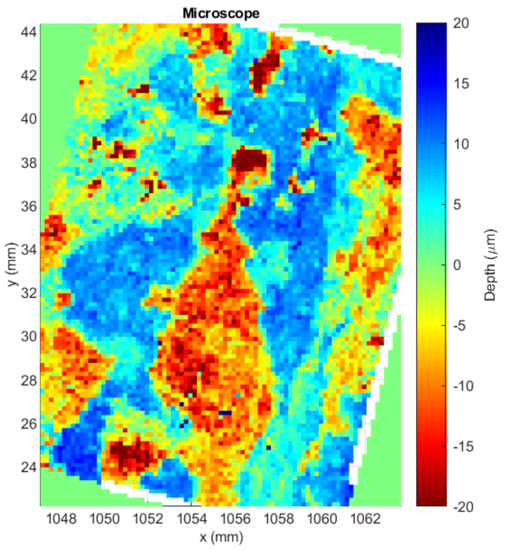


(d) Intersection

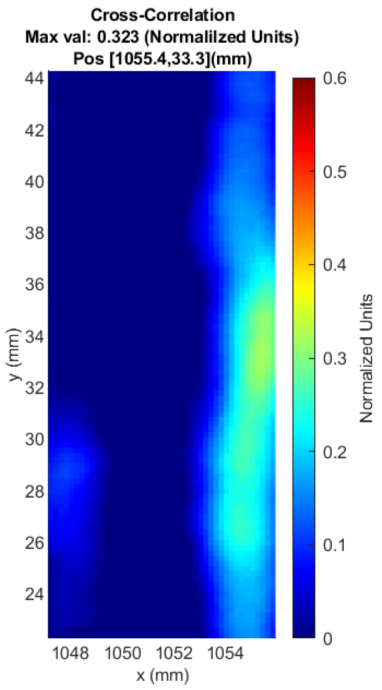
Figure 25 Convex sample c3, defect D3, full scan.



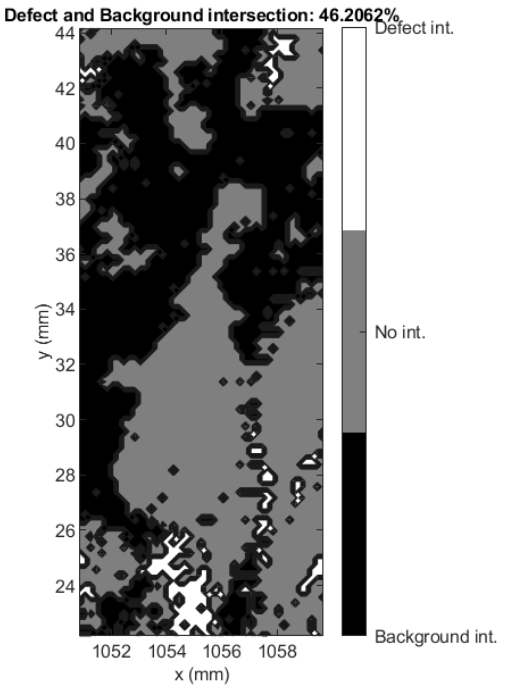
(a) ECA



(b) VHX

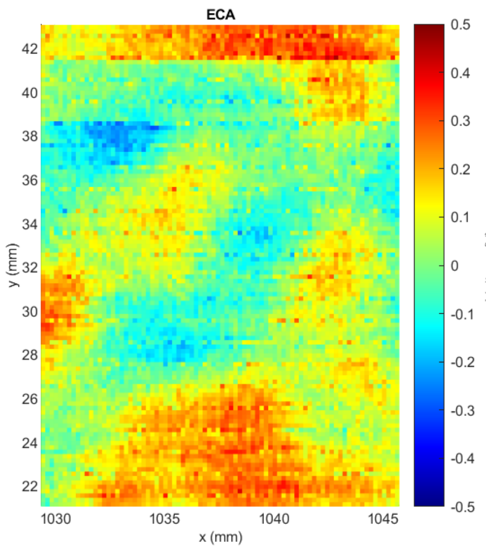


(c) Xcor

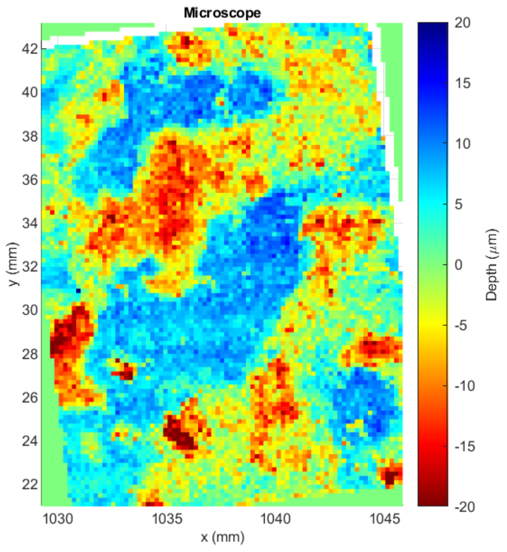


(d) Intersection

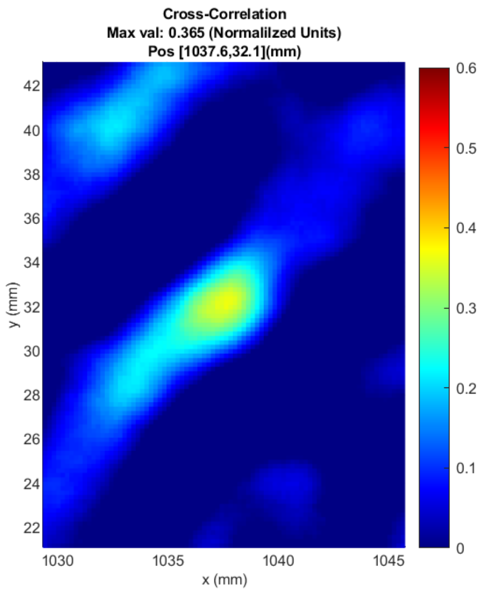
Figure 26 Convex sample c4, defect D1, full scan.



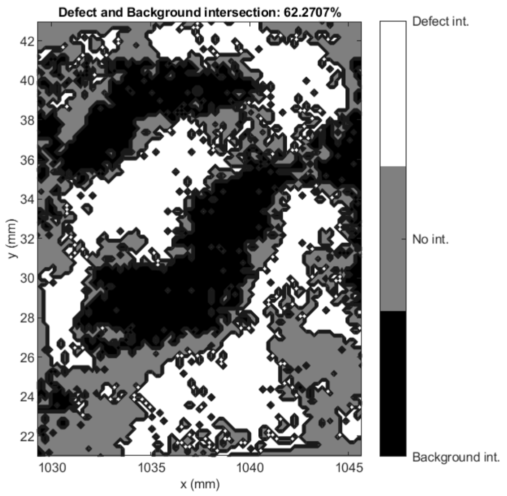
(a) ECA



(b) VHX

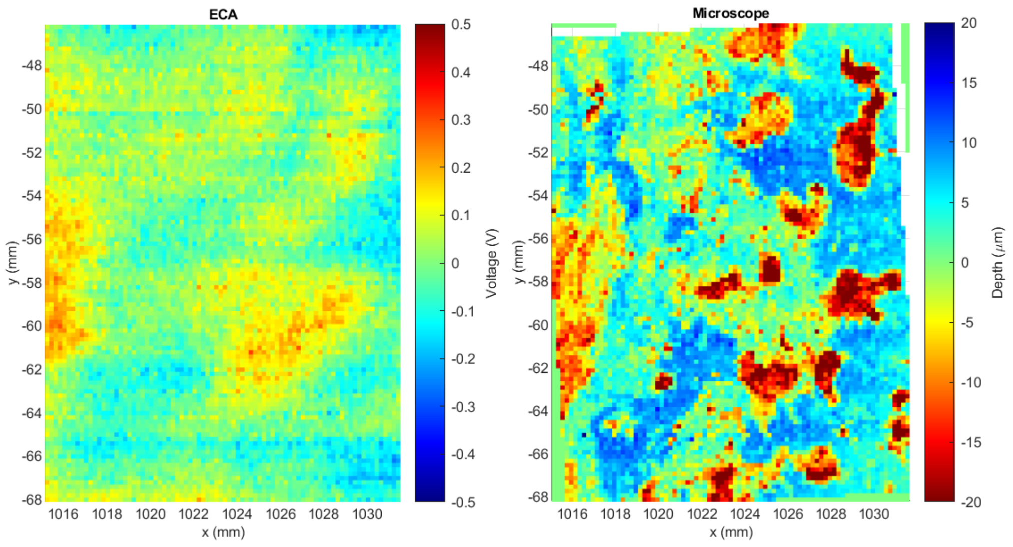


(c) Xcor

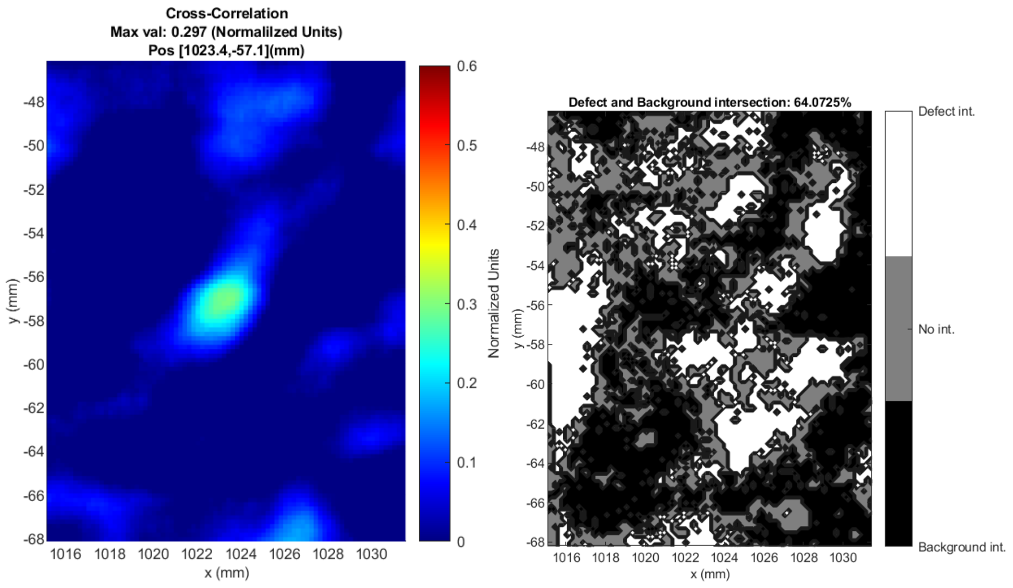


(d) Intersection

Figure 27 Convex sample c4, defect D2, full scan.

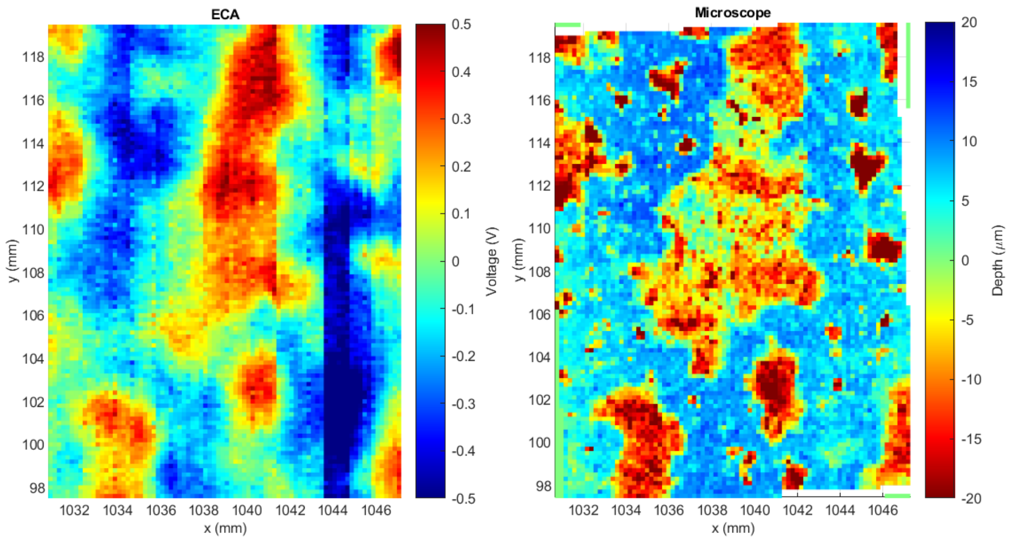


(a) ECA (b) VHX

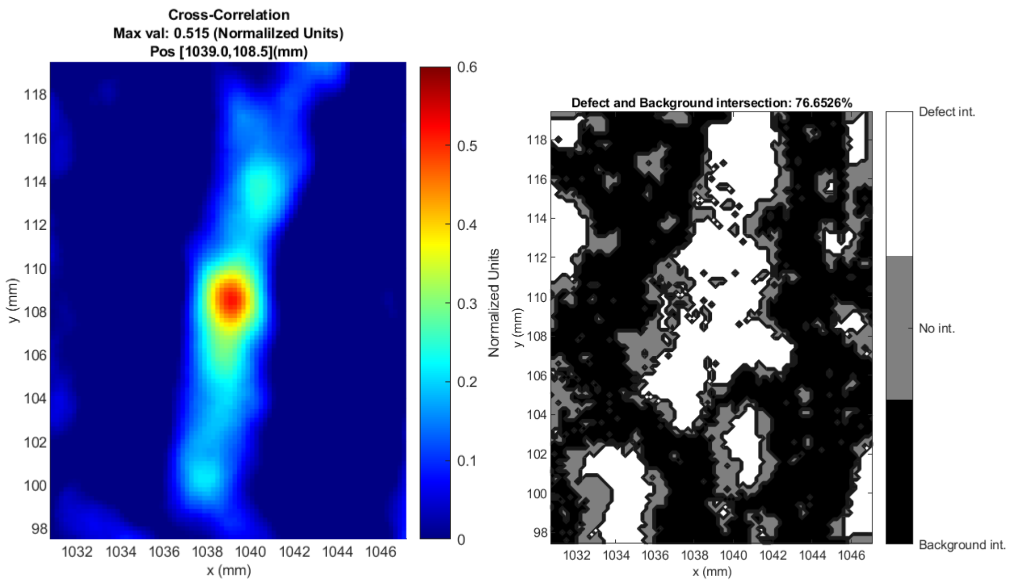


(c) Xcor (d) Intersection

Figure 28 Convex sample c4, defect D3, full scan.

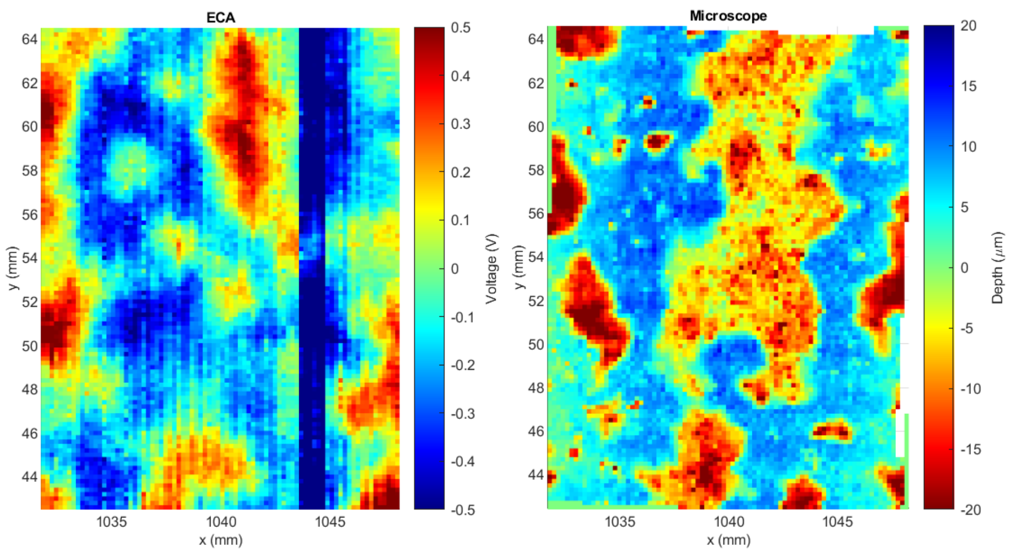


(a) ECA (b) VHX

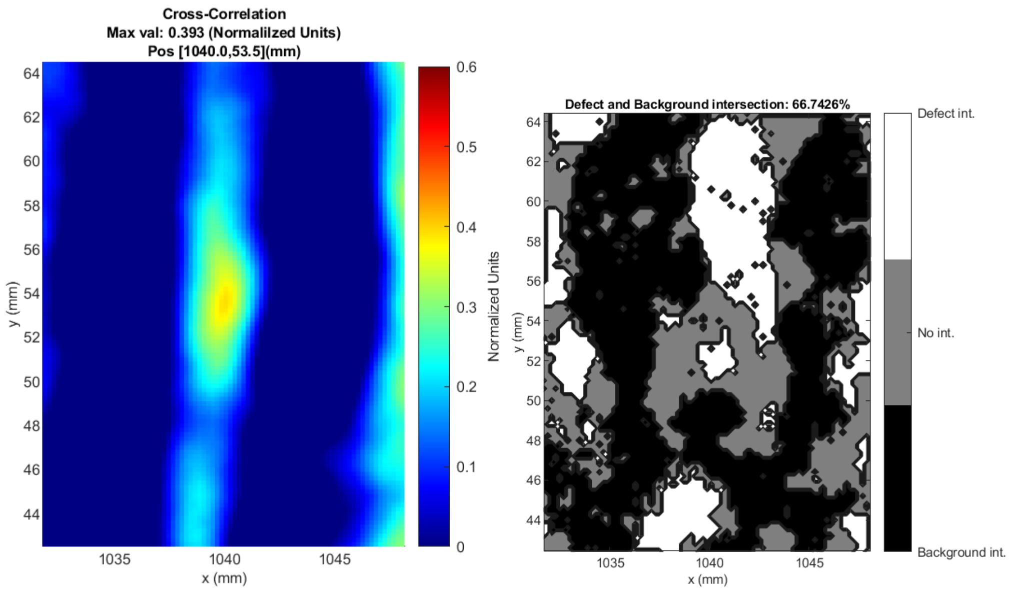


(c) Xcor (d) Intersection

Figure 29 Flat sample a, defect D1, fast scan.

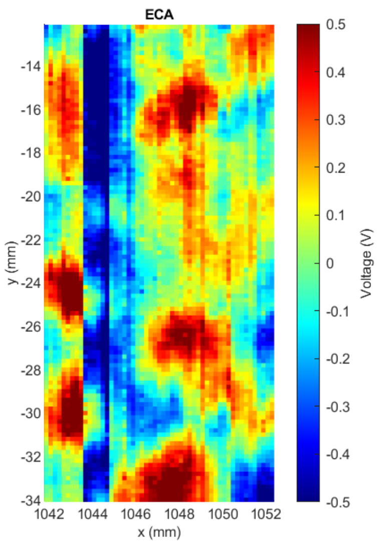


(a) ECA (b) VHX

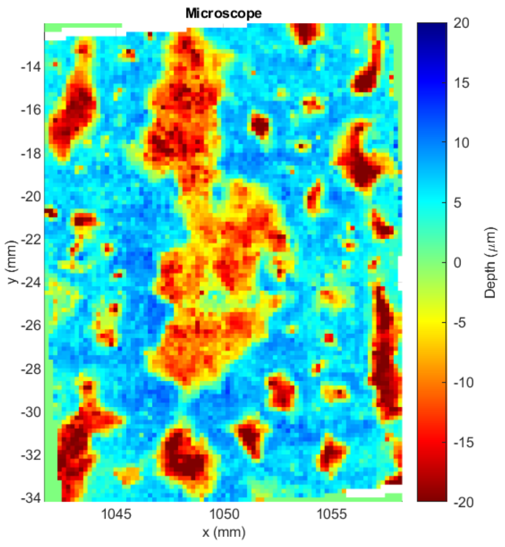


(c) Xcor (d) Intersection

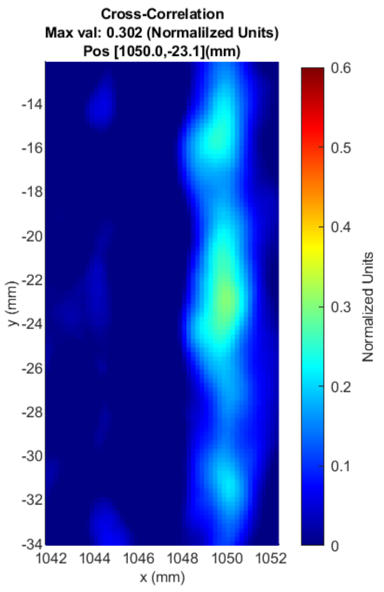
Figure 30 Flat sample a, defect D2, fast scan.



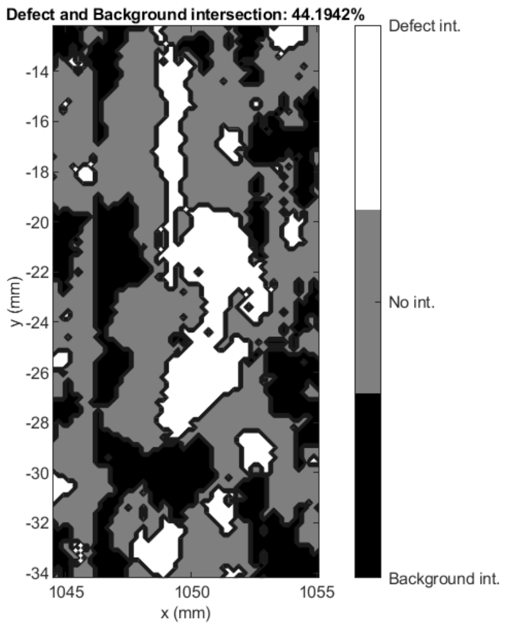
(a) ECA



(b) VHX

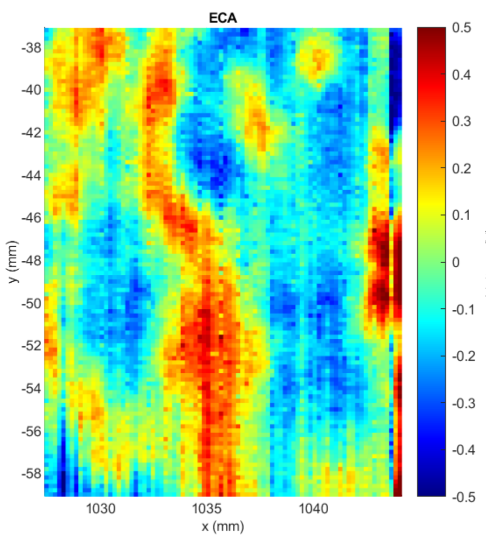


(c) Xcor

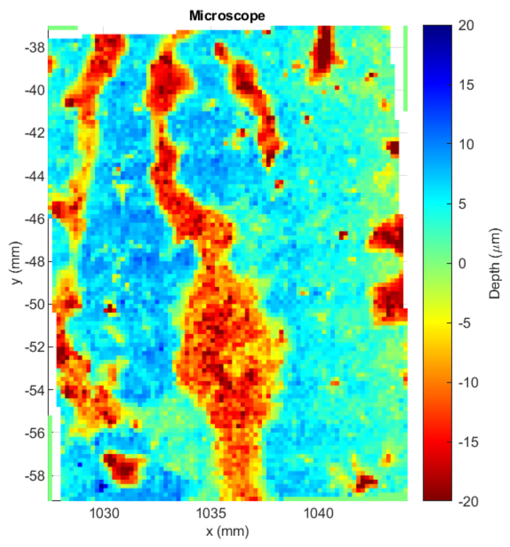


(d) Intersection

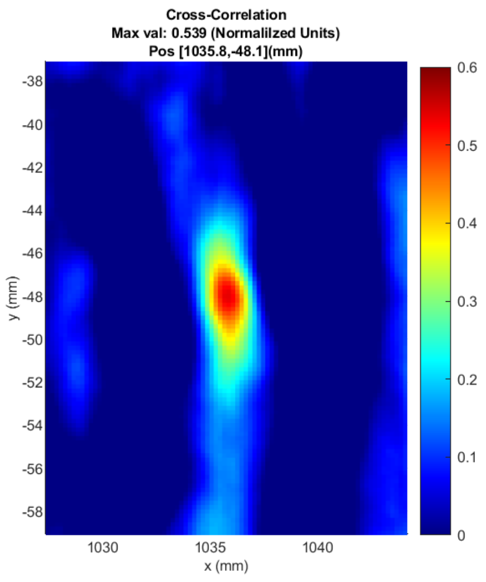
Figure 31 Flat sample a, defect D3, fast scan.



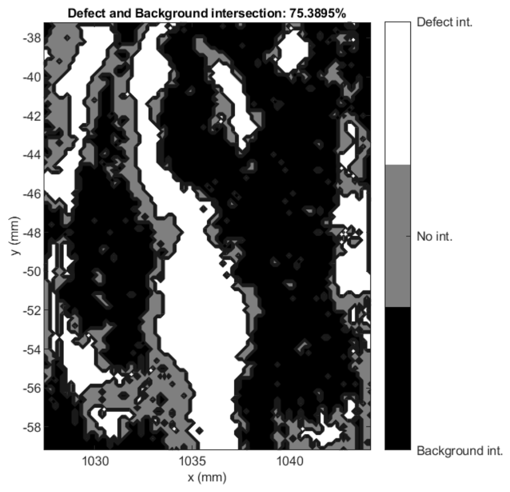
(a) ECA



(b) VHX

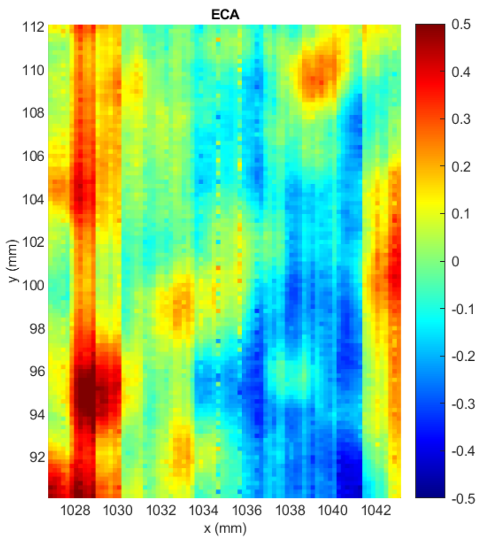


(c) Xcor

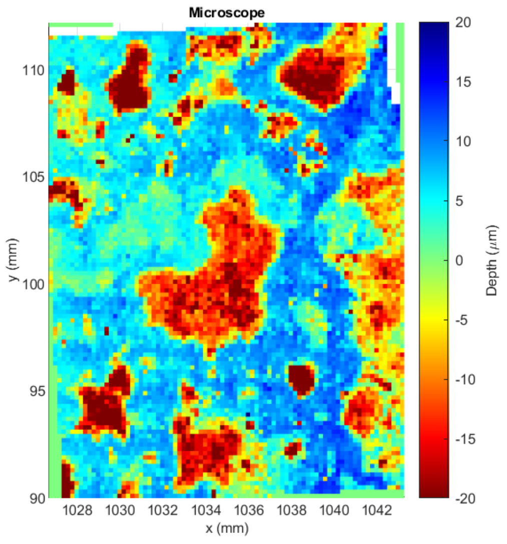


(d) Intersection

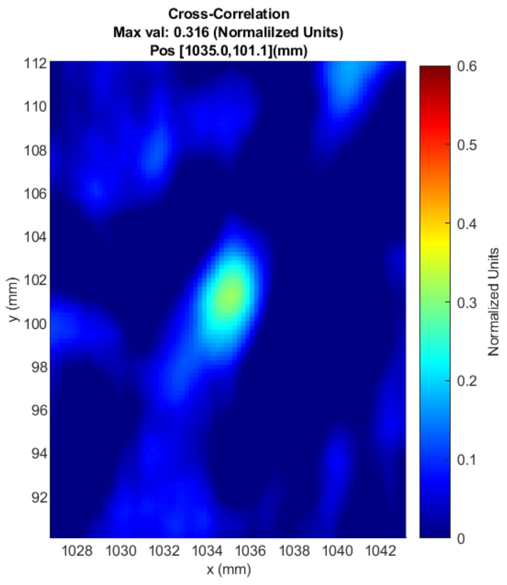
Figure 32 Flat sample a, defect D4, fast scan.



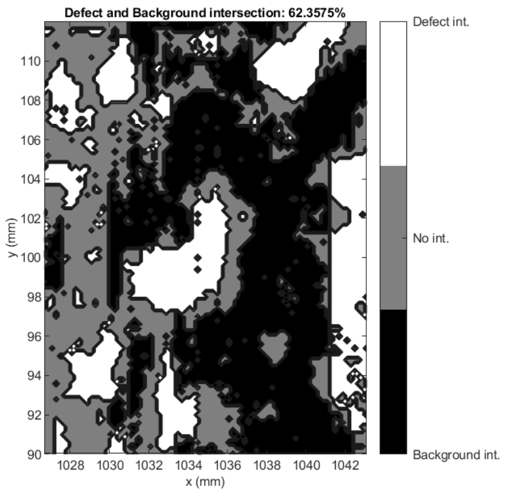
(a) ECA



(b) VHX

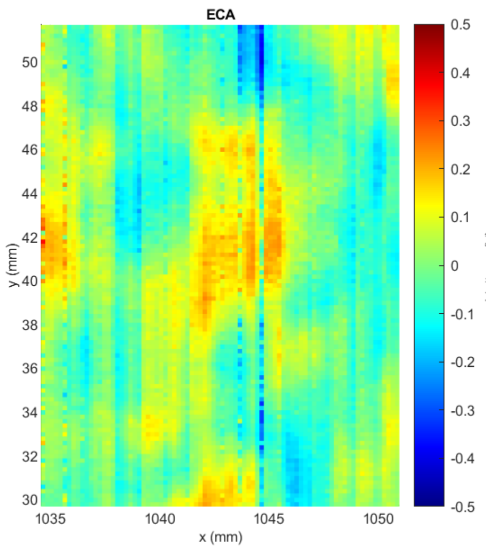


(c) Xcor

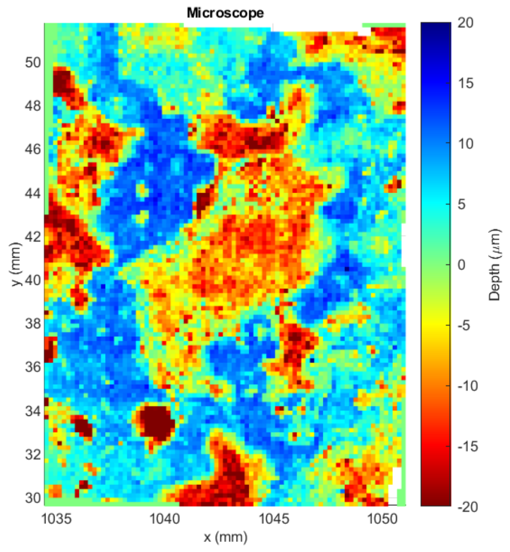


(d) Intersection

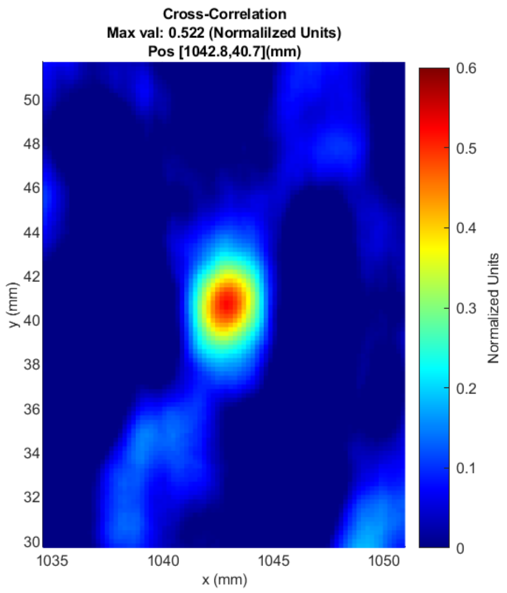
Figure 33 Concave sample b1, defect D1, fast scan.



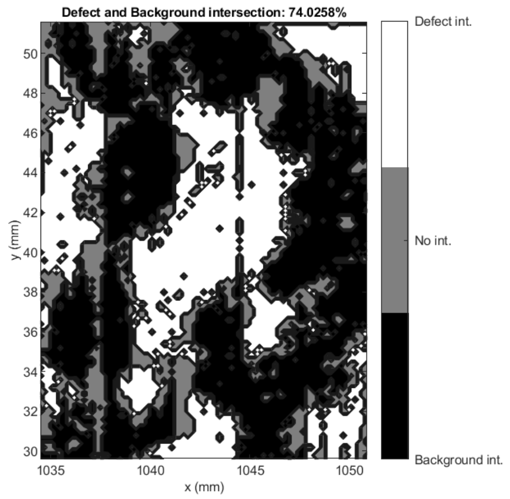
(a) ECA



(b) VHX

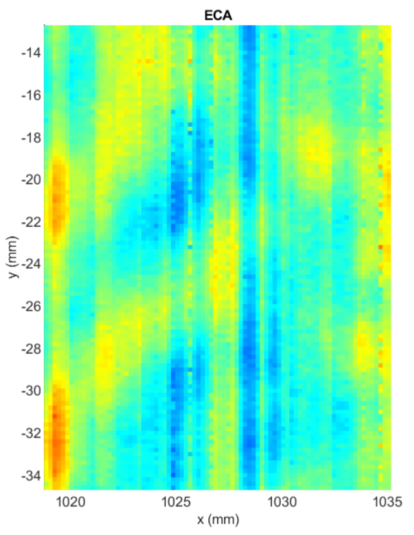


(c) Xcor

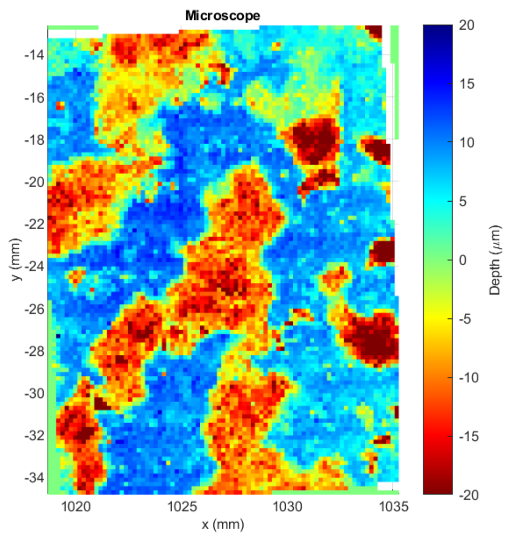


(d) Intersection

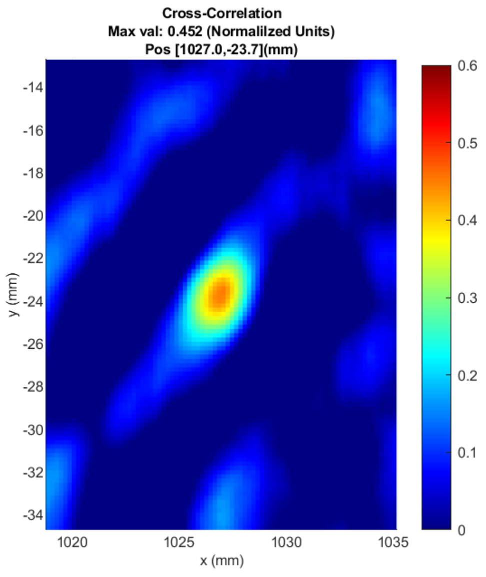
Figure 34 Concave sample b1, defect D2, fast scan.



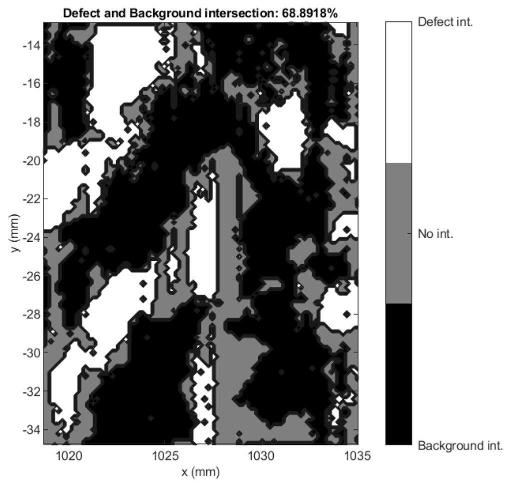
(a) ECA



(b) VHX

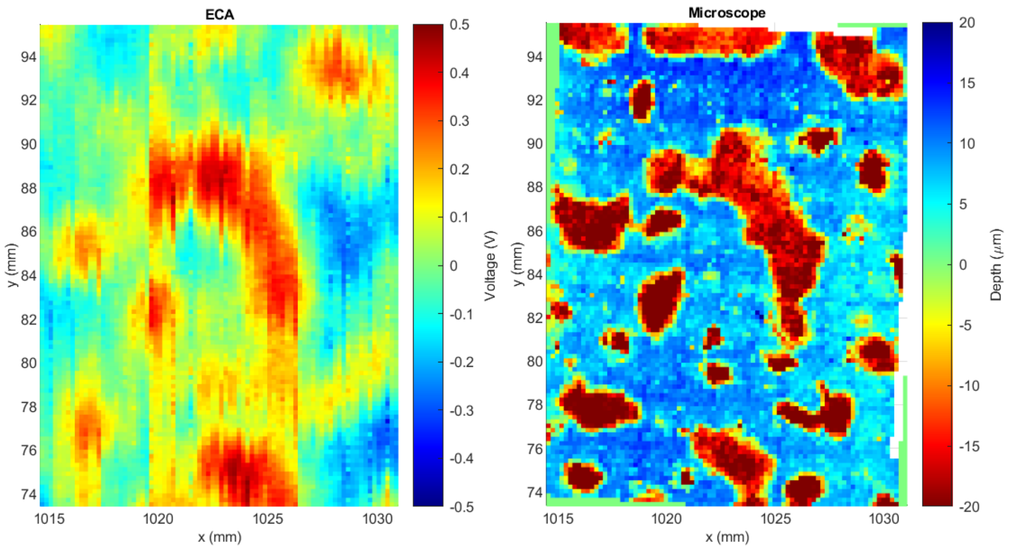


(c) Xcor

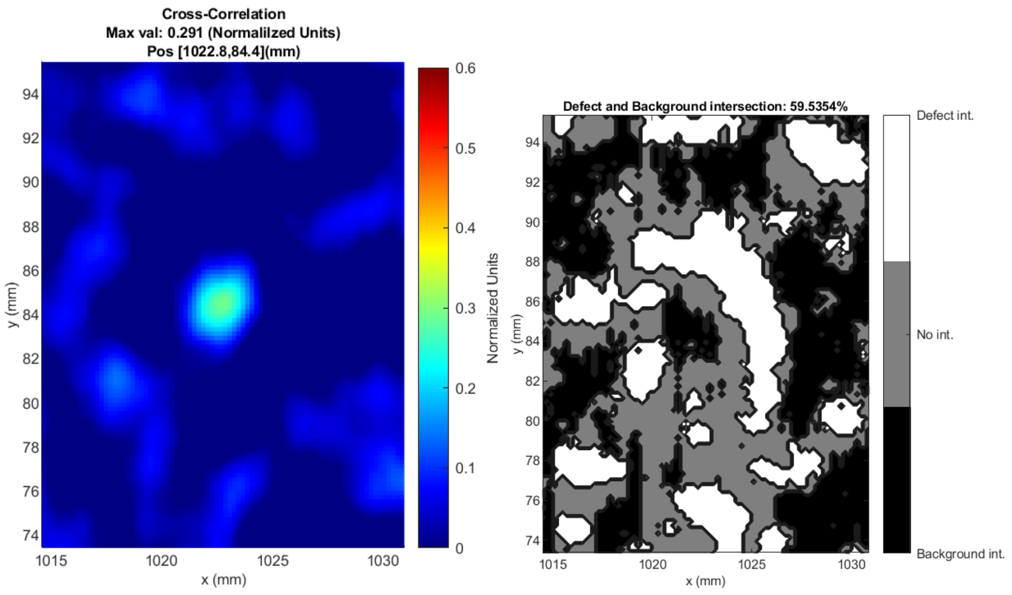


(d) Intersection

Figure 35 Concave sample b1, defect D3, fast scan.

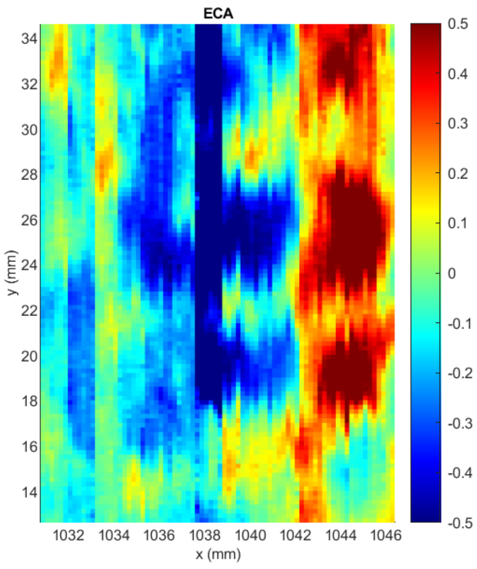


(a) ECA (b) VHX

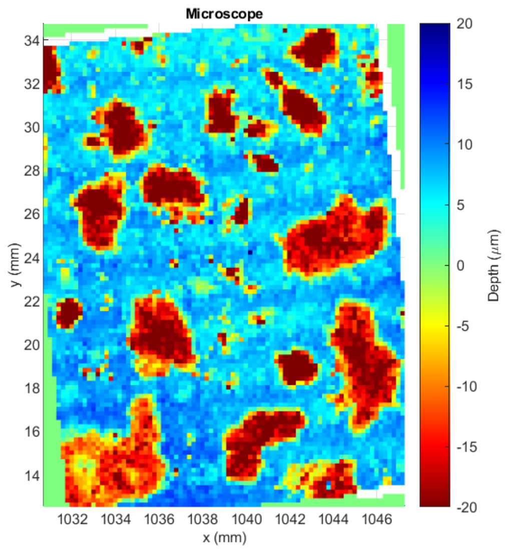


(c) Xcor (d) Intersection

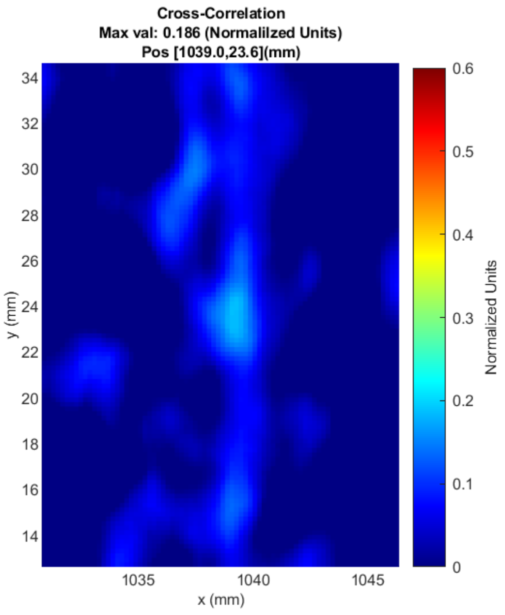
Figure 36 Concave sample b2, defect D1, fast scan.



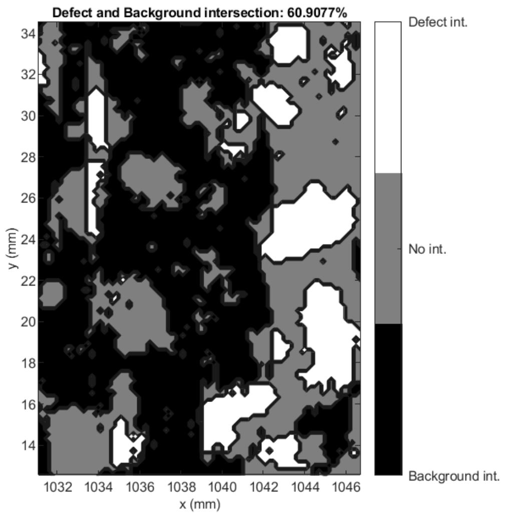
(a) ECA



(b) VHX

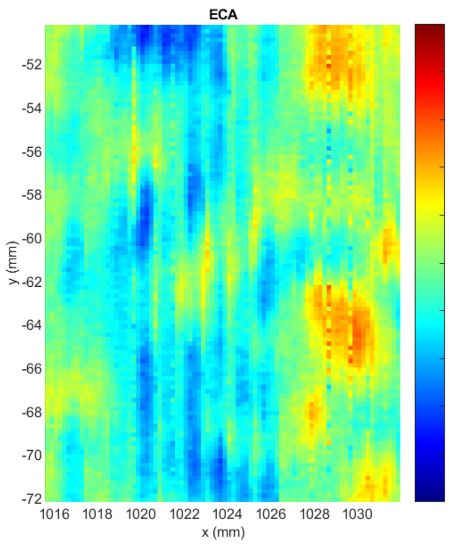


(c) Xcor

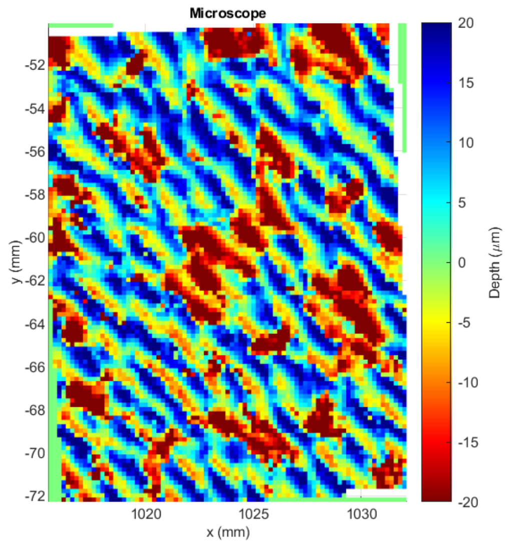


(d) Intersection

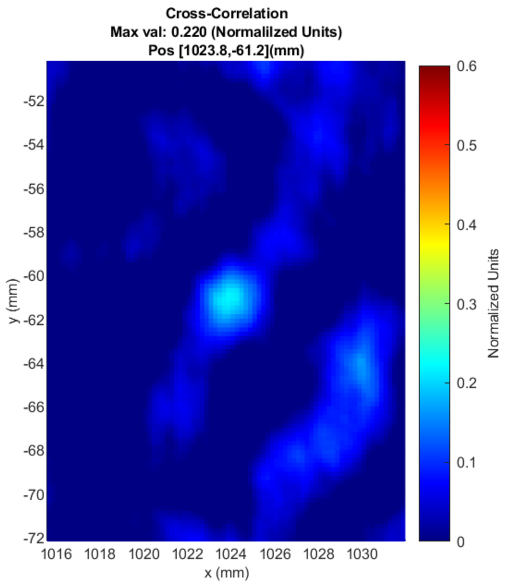
Figure 37 Concave sample b2, defect D2, fast scan.



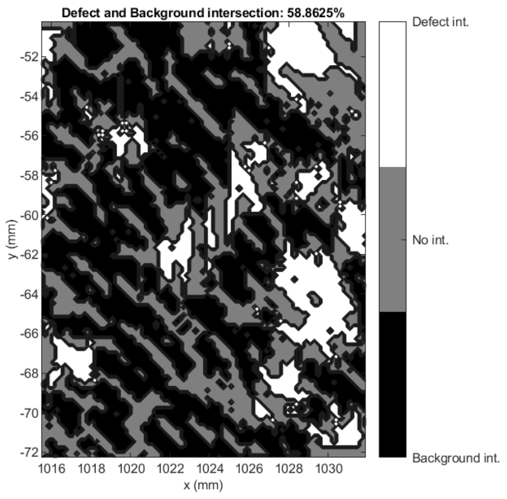
(a) ECA



(b) VHX

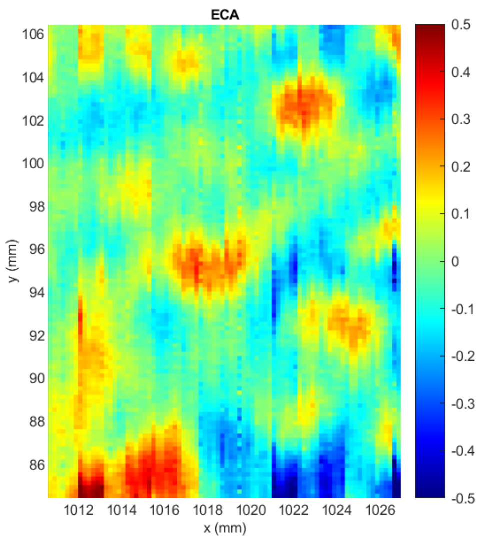


(c) Xcor

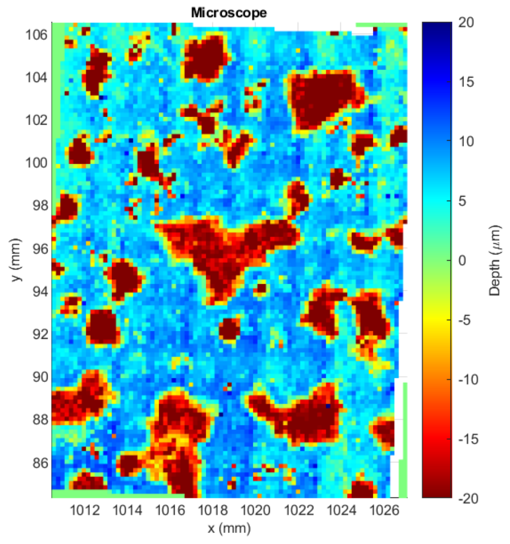


(d) Intersection

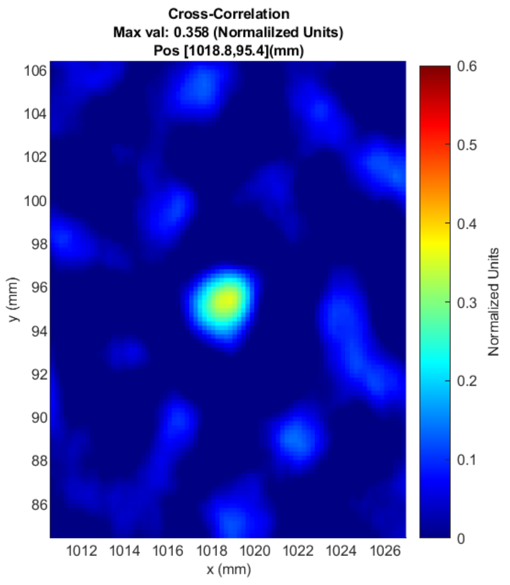
Figure 38 Concave sample b2, defect D3, fast scan.



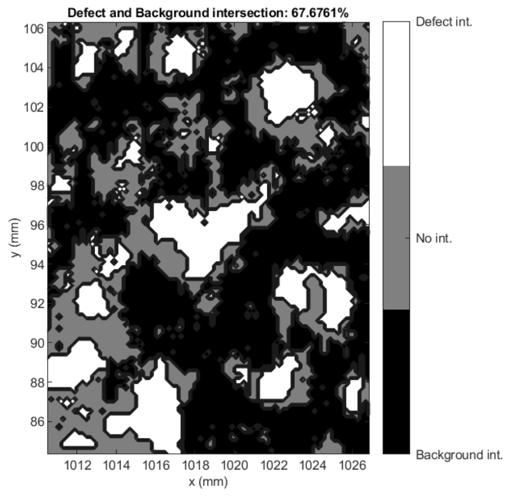
(a) ECA



(b) VHX

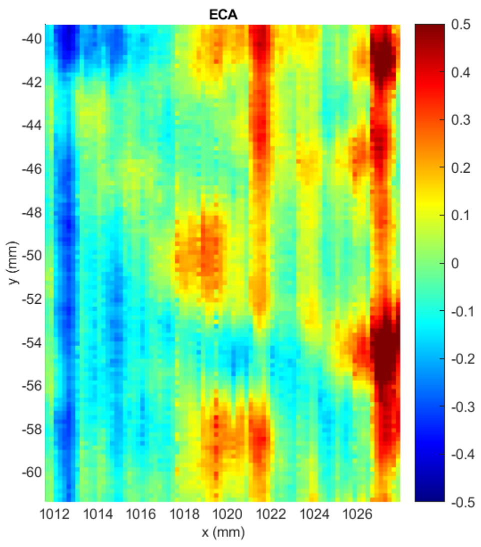


(c) Xcor

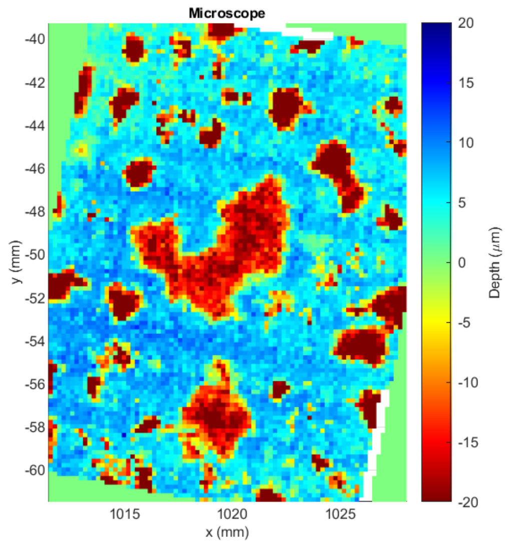


(d) Intersection

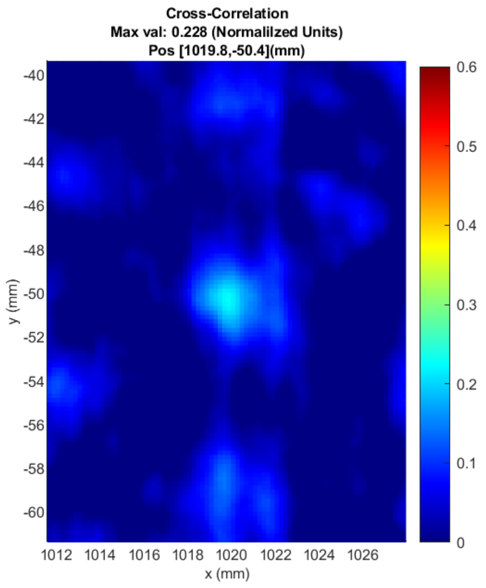
Figure 39 Concave sample b3, defect D1, fast scan.



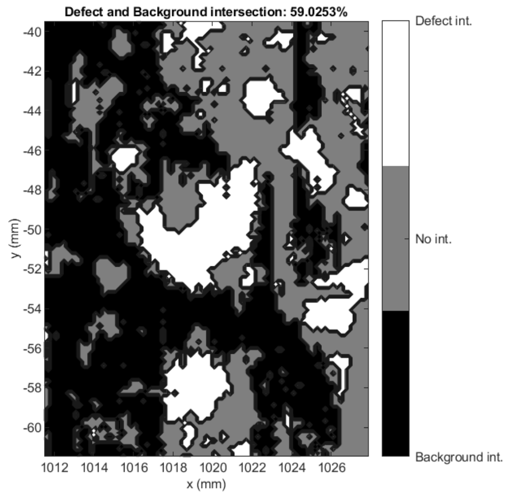
(a) ECA



(b) VHX

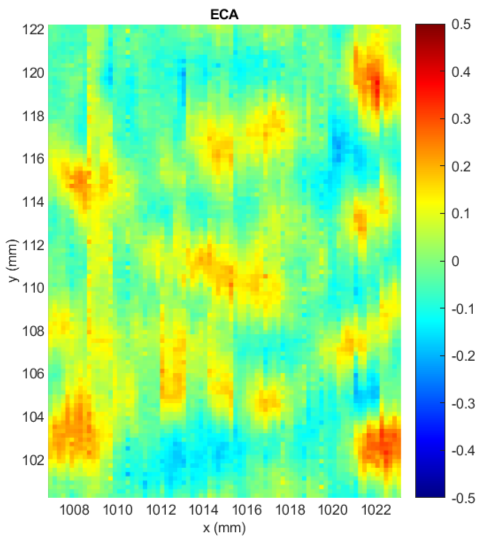


(c) Xcor

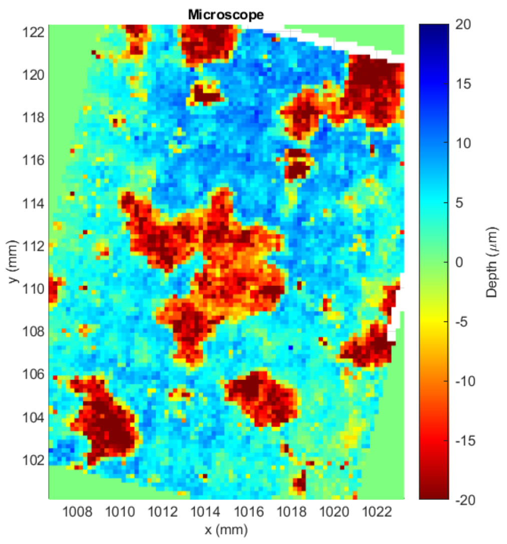


(d) Intersection

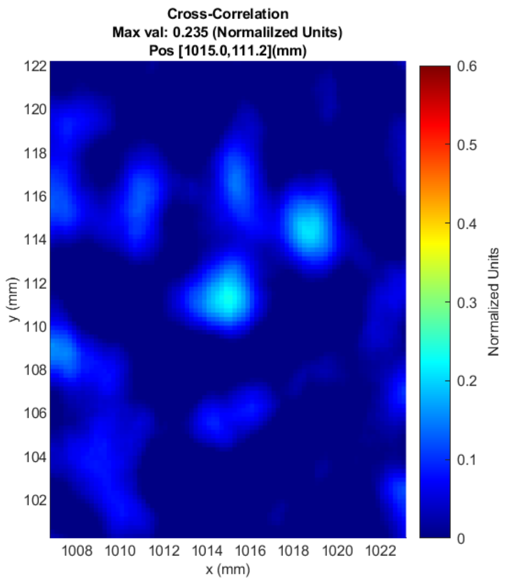
Figure 40 Concave sample b3, defect D2, fast scan.



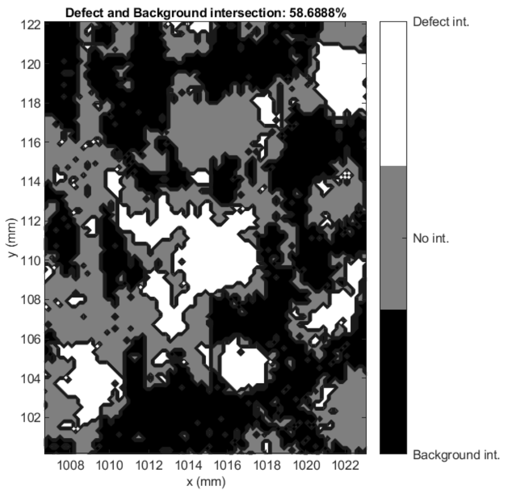
(a) ECA



(b) VHX

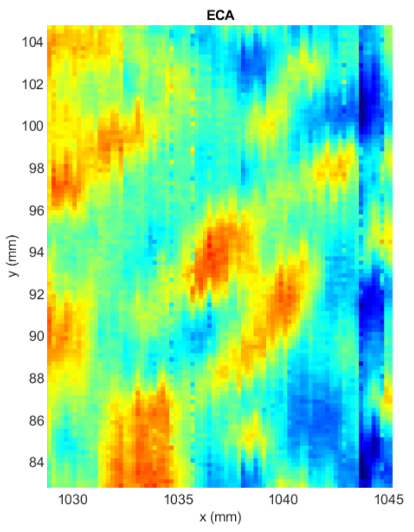


(c) Xcor

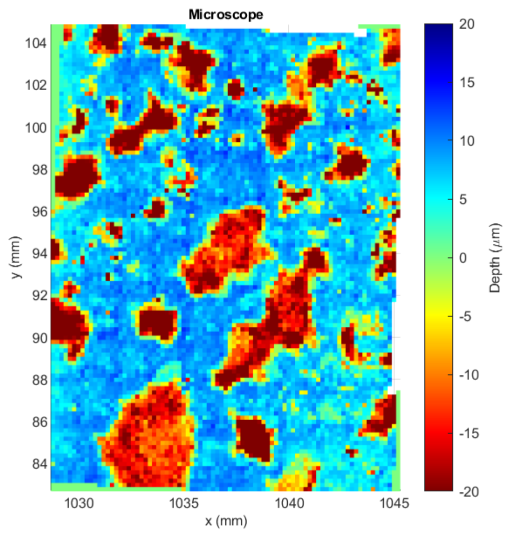


(d) Intersection

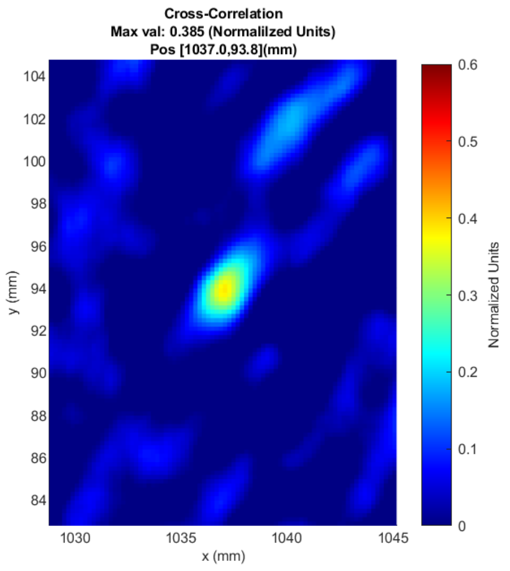
Figure 41 Concave sample b3, defect D3, fast scan.



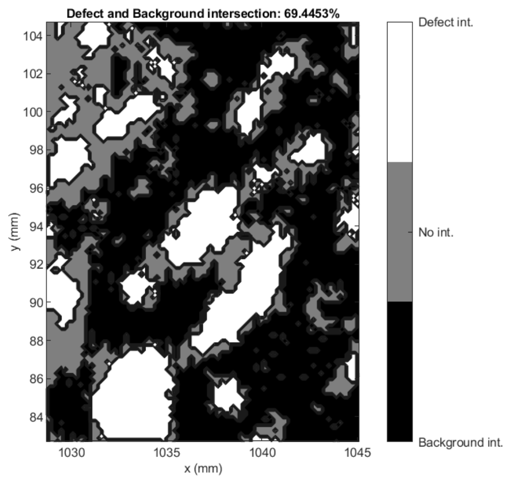
(a) ECA



(b) VHX

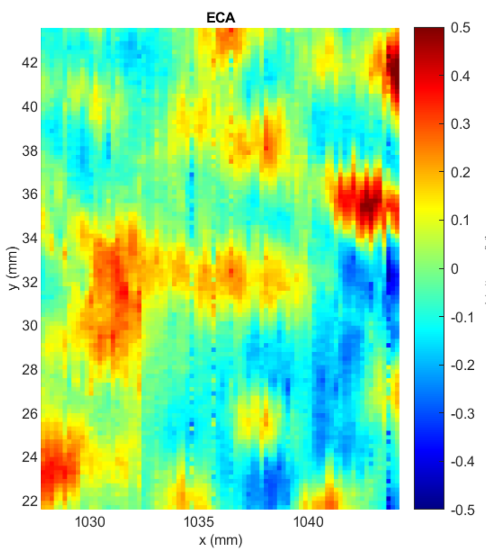


(c) Xcor

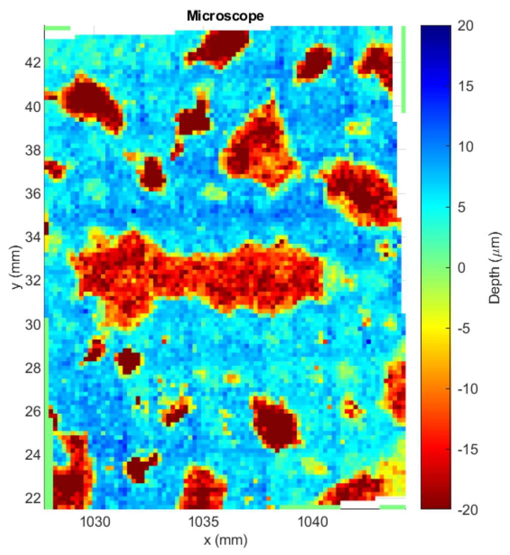


(d) Intersection

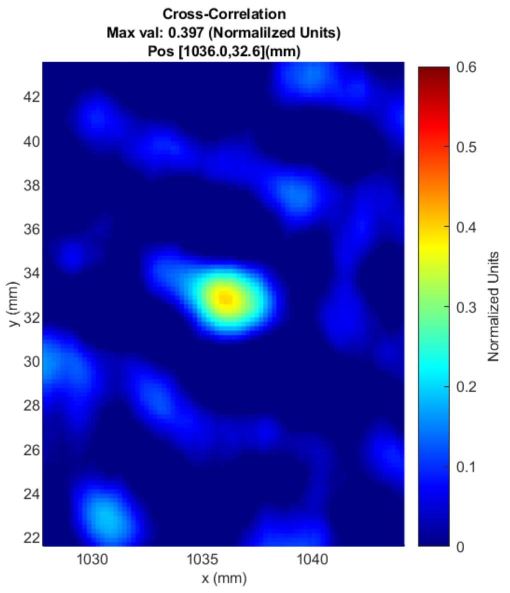
Figure 42 Concave sample b4, defect D1, fast scan.



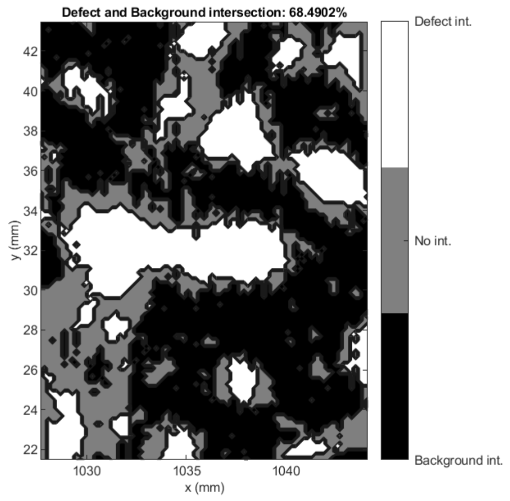
(a) ECA



(b) VHX

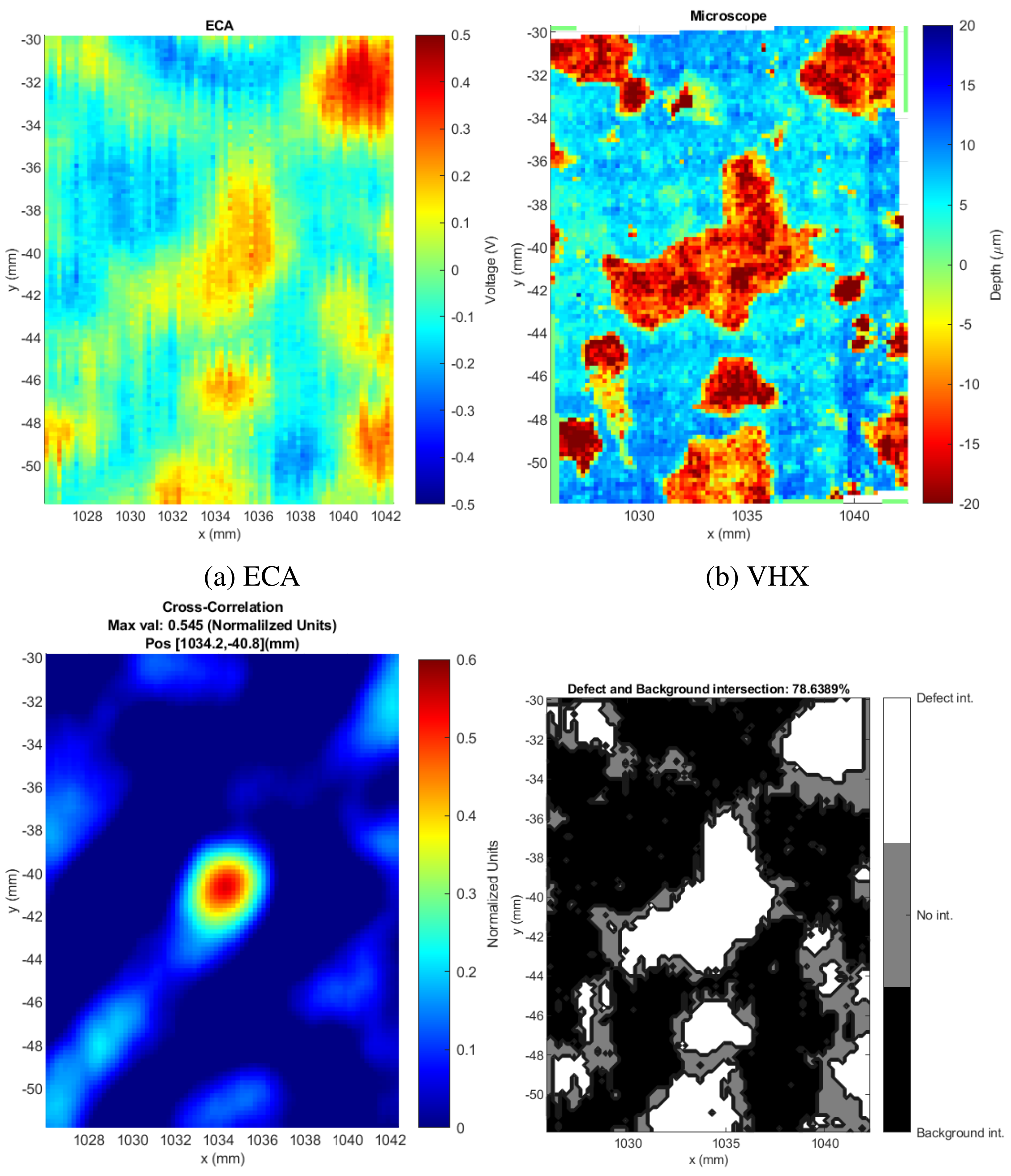


(c) Xcor



(d) Intersection

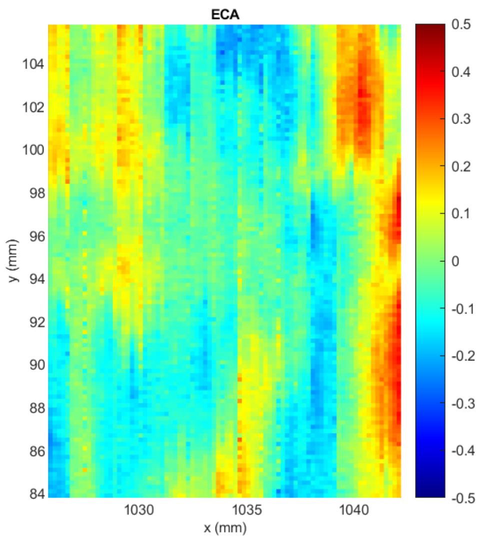
Figure 43 Concave sample b4, defect D2, fast scan.



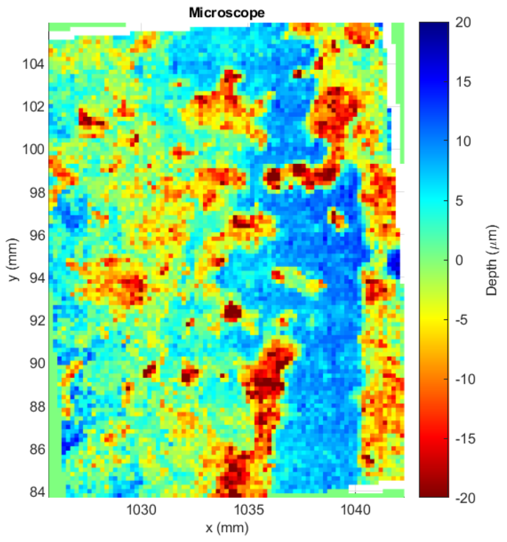
(a) ECA (b) VHX

(c) Xcor (d) Intersection

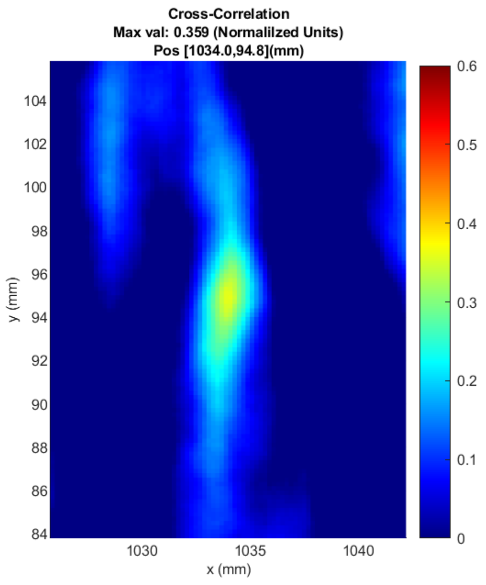
Figure 44 Concave sample b4, defect D3, fast scan.



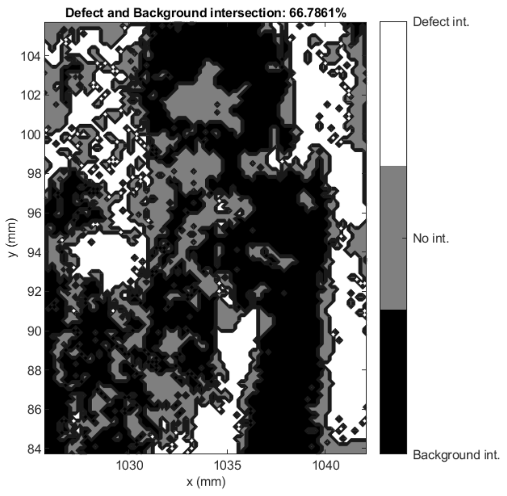
(a) ECA



(b) VHX

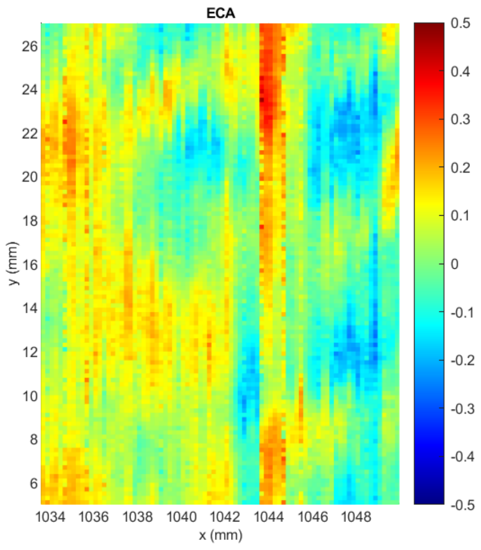


(c) Xcor

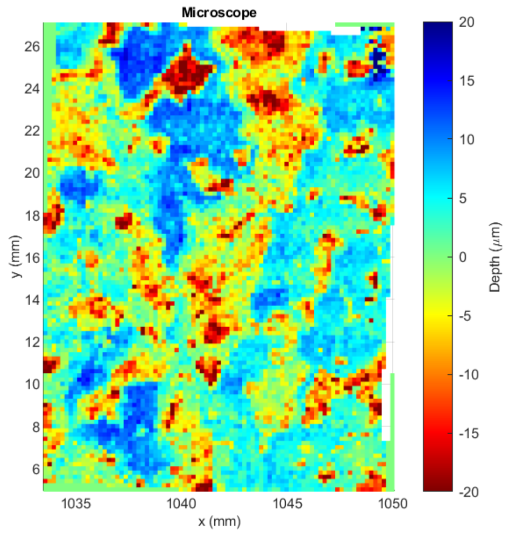


(d) Intersection

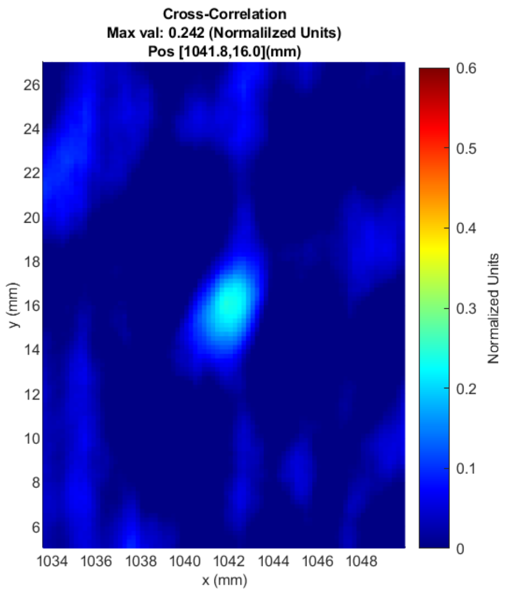
Figure 45 Convex sample c1, defect D1, fast scan.



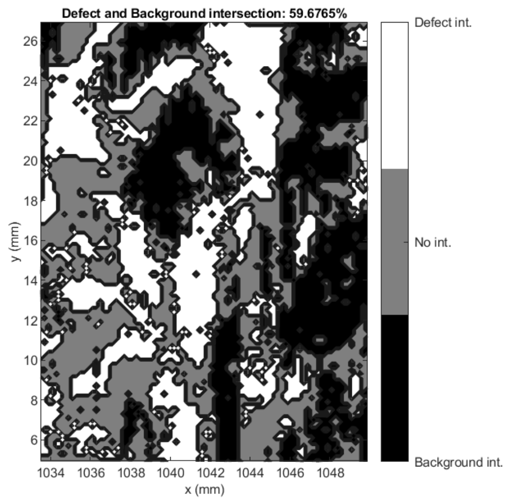
(a) ECA



(b) VHX

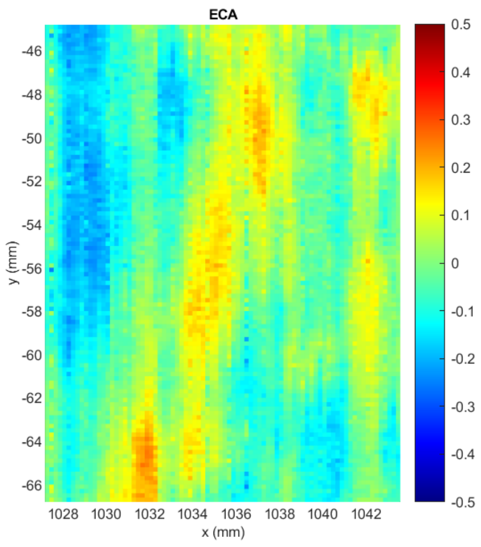


(c) Xcor

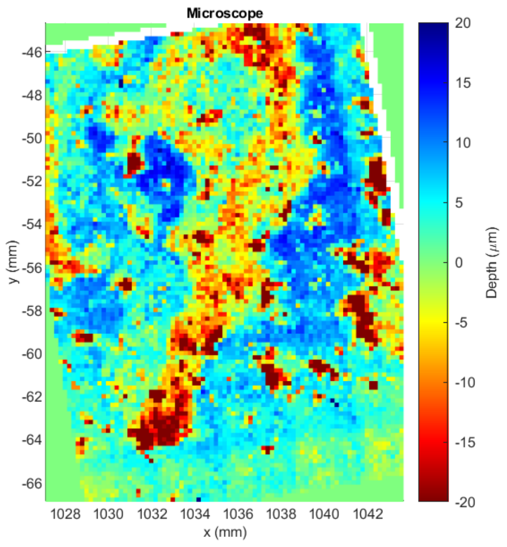


(d) Intersection

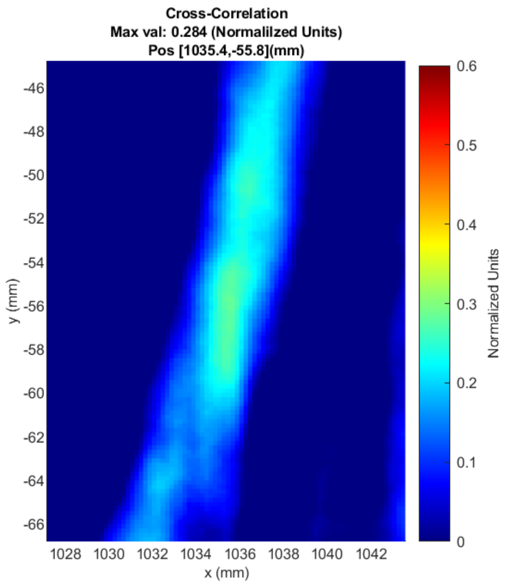
Figure 46 Convex sample c1, defect D2, fast scan.



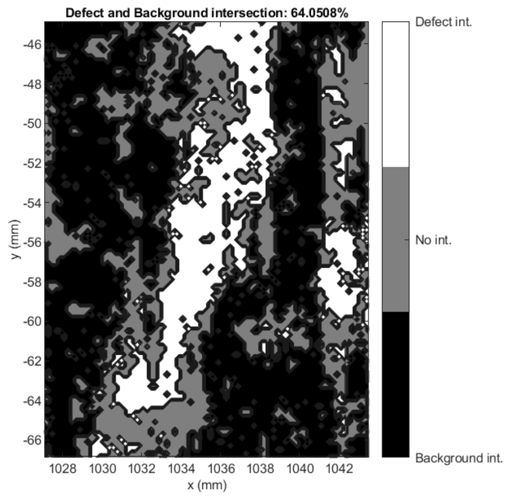
(a) ECA



(b) VHX

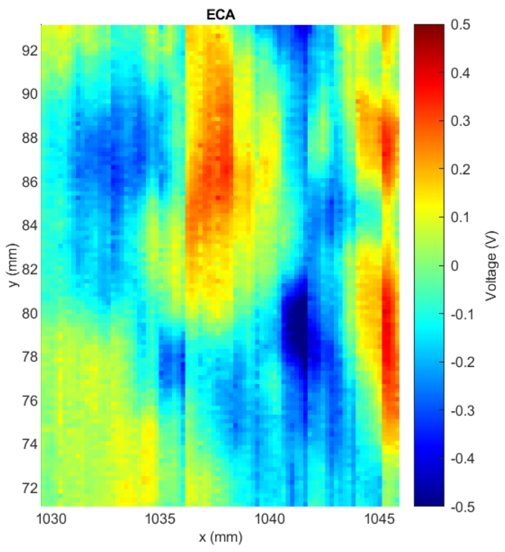


(c) Xcor

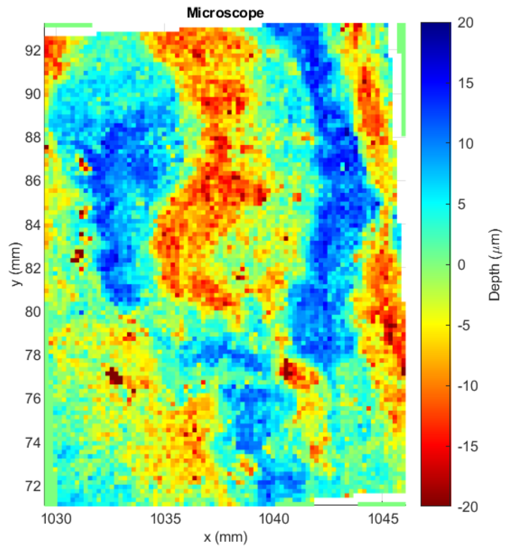


(d) Intersection

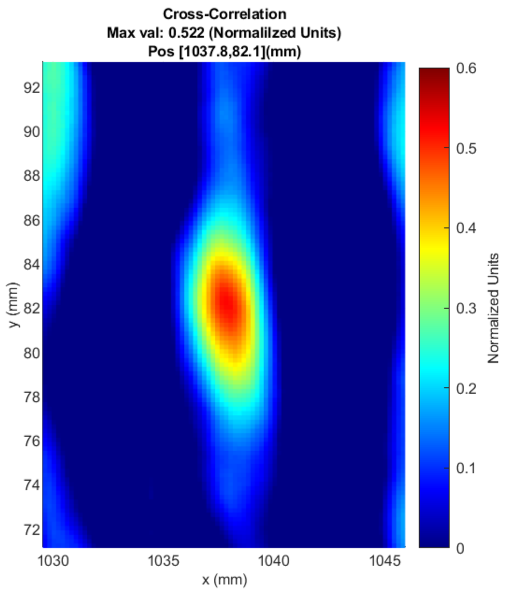
Figure 47 Convex sample c1, defect D3, fast scan.



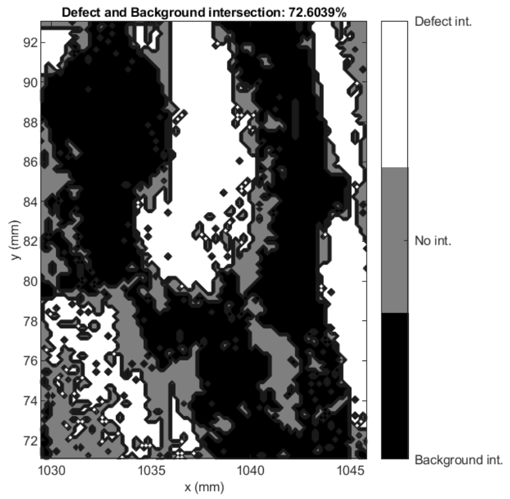
(a) ECA



(b) VHX

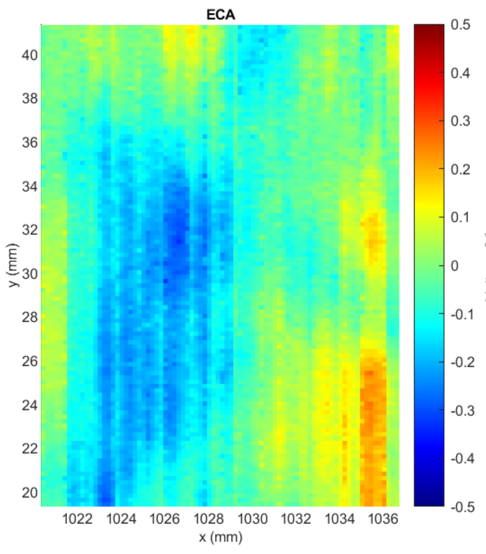


(c) Xcor

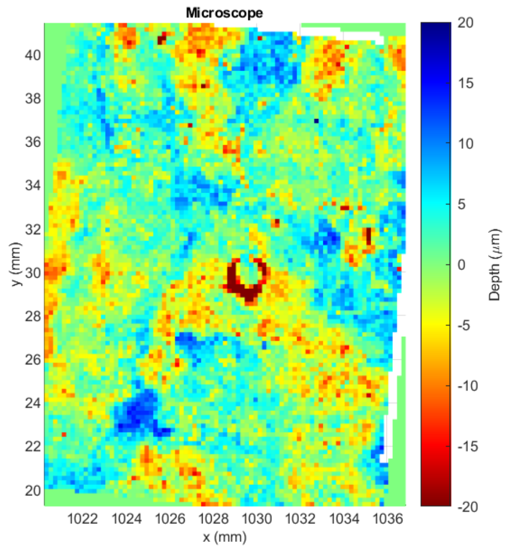


(d) Intersection

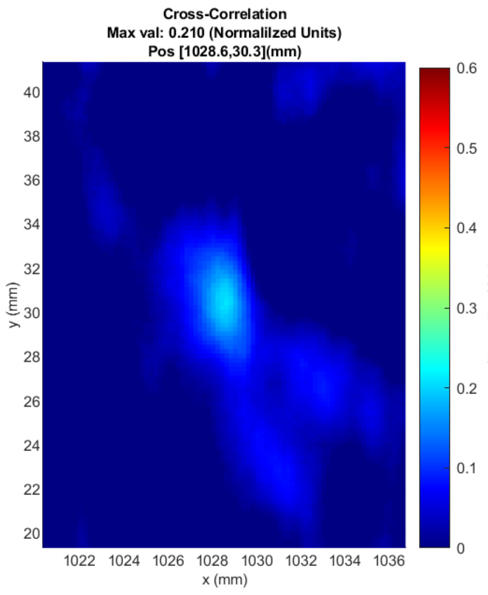
Figure 48 Convex sample c2, defect D1, fast scan.



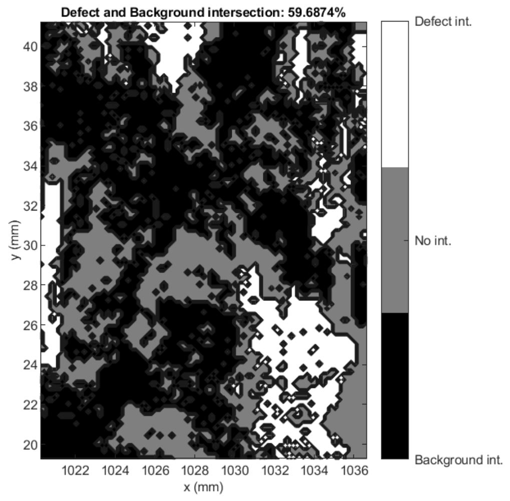
(a) ECA



(b) VHX

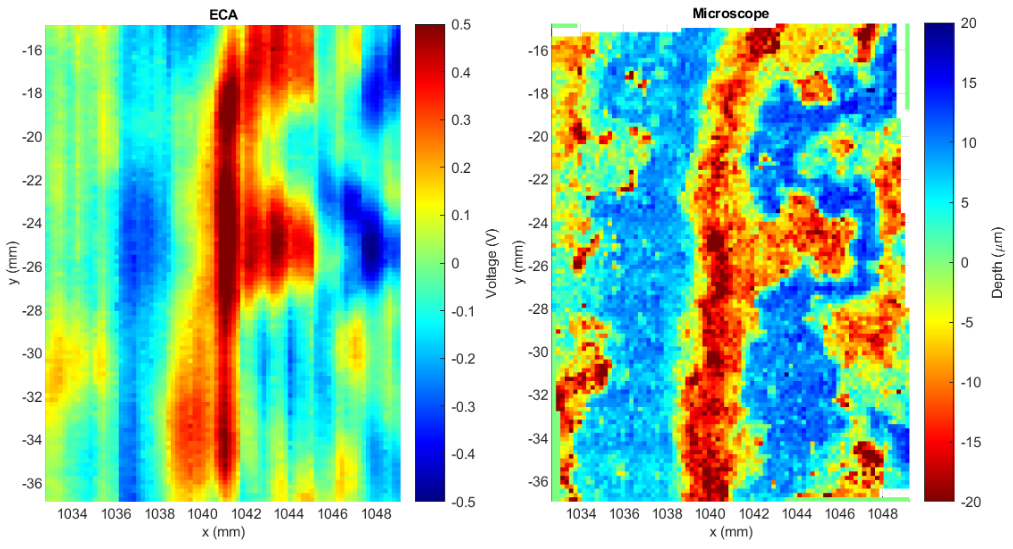


(c) Xcor

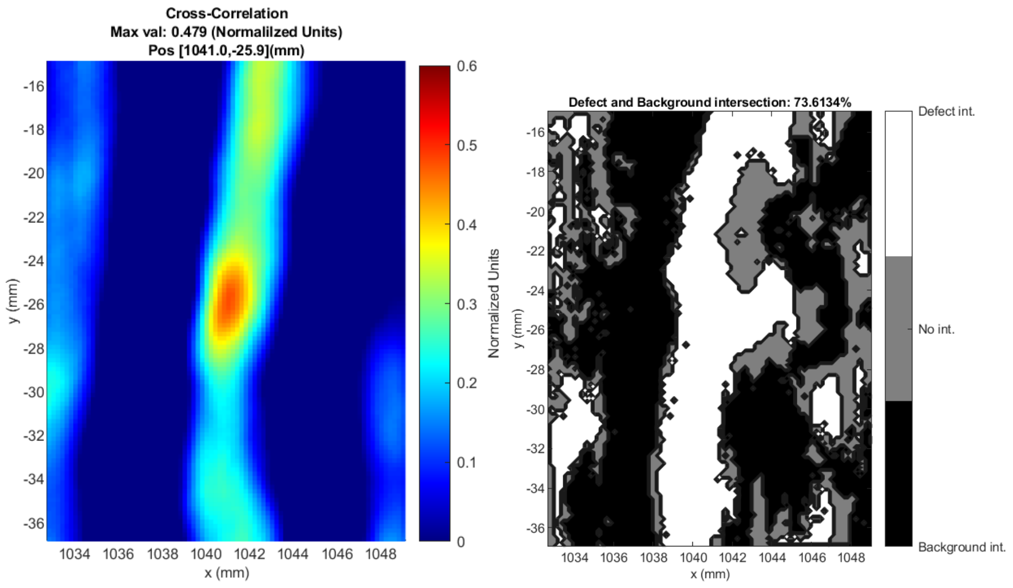


(d) Intersection

Figure 49 Convex sample c2, defect D2, fast scan.

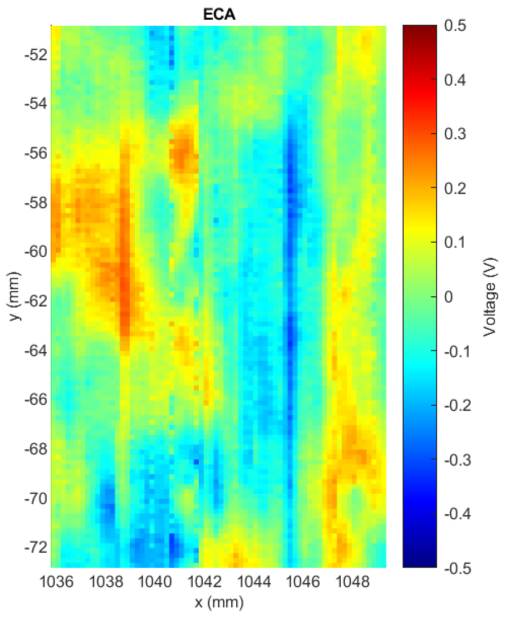


(a) ECA (b) VHX

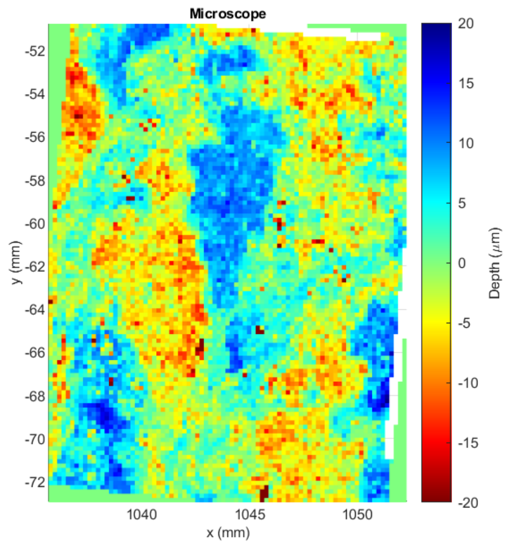


(c) Xcor (d) Intersection

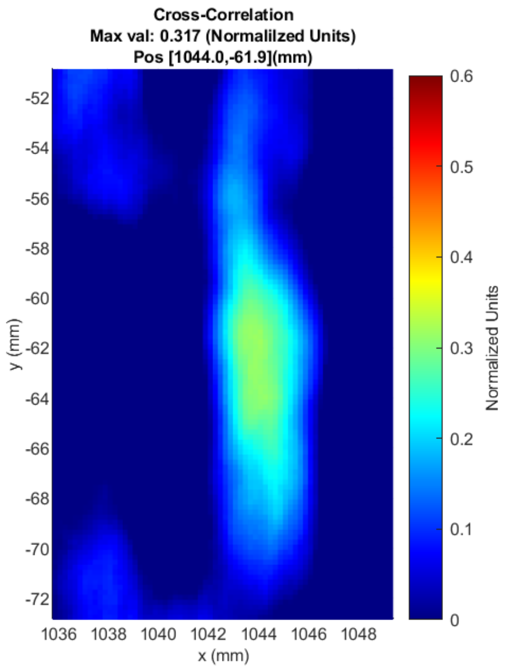
Figure 50 Convex sample c2, defect D3, fast scan.



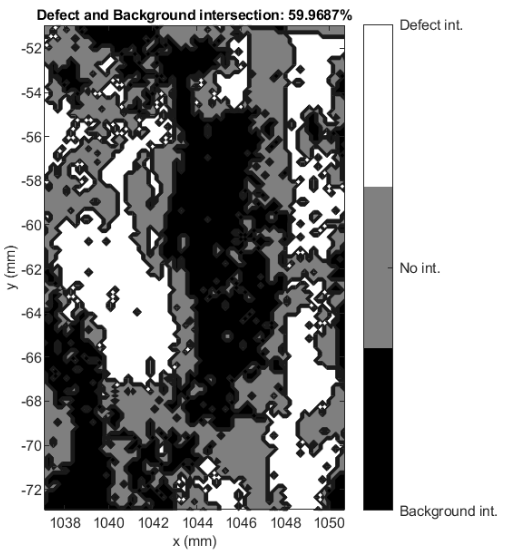
(a) ECA



(b) VHX

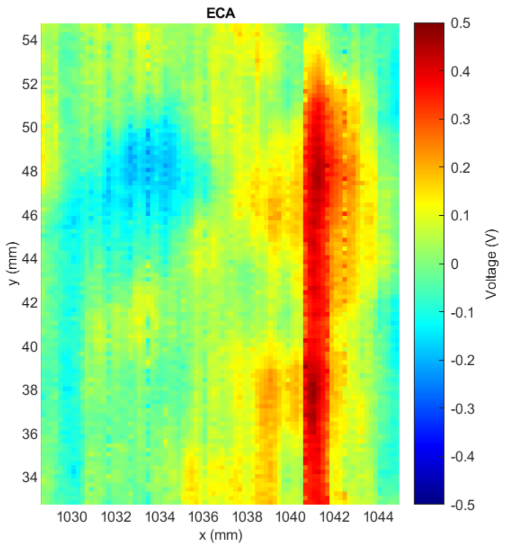


(c) Xcor

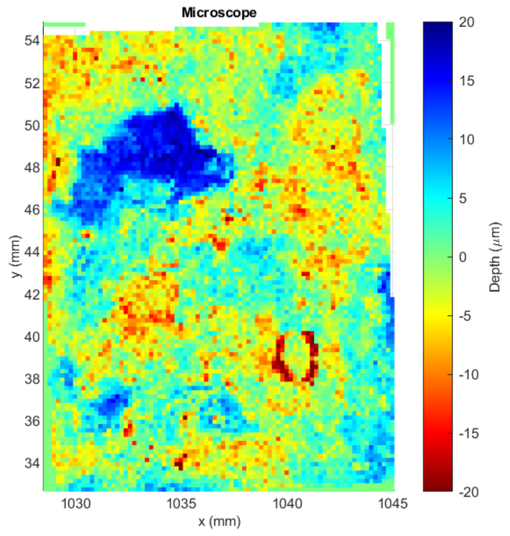


(d) Intersection

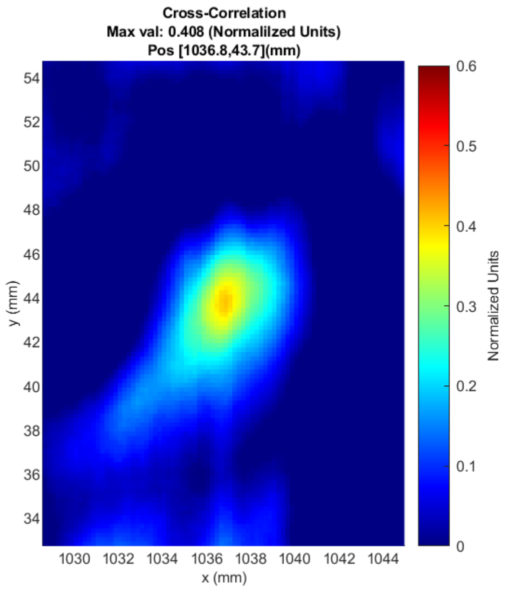
Figure 51 Convex sample c3, defect D1, fast scan.



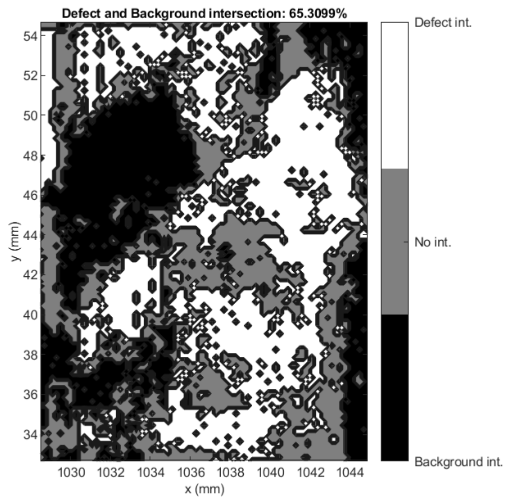
(a) ECA



(b) VHX

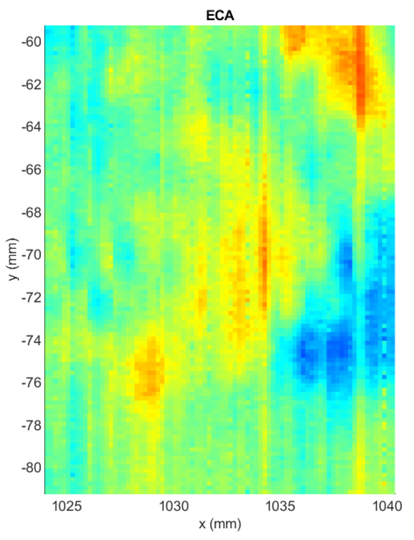


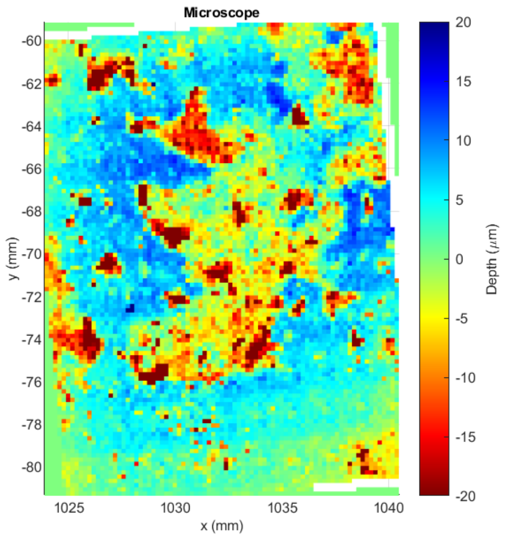
(c) Xcor



(d) Intersection

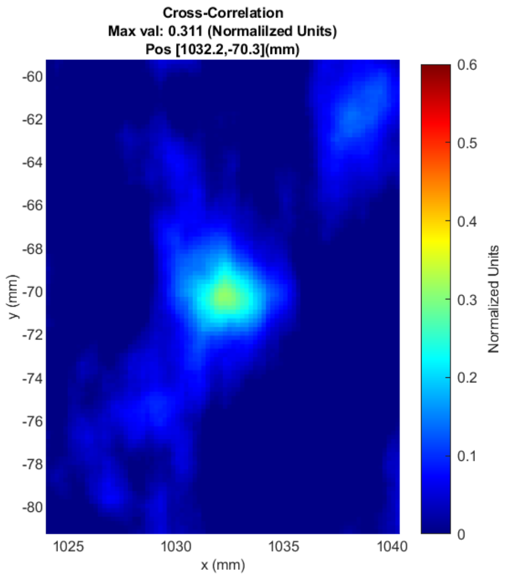
Figure 52 Convex sample c3, defect D2, fast scan.

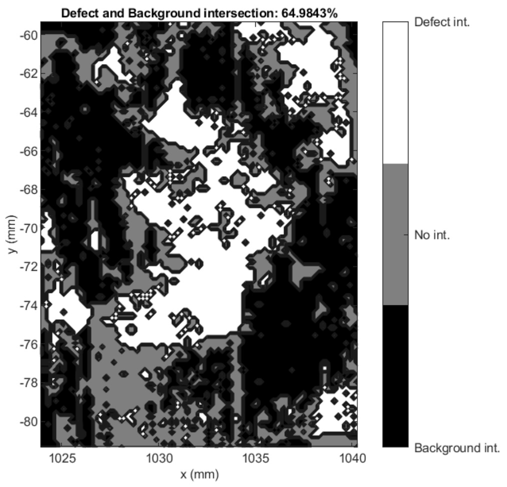




(a) ECA

(b) VHX

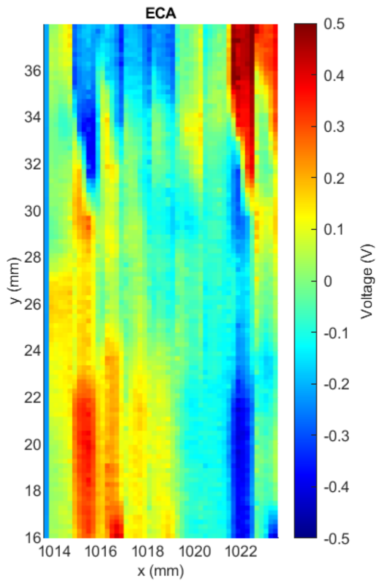




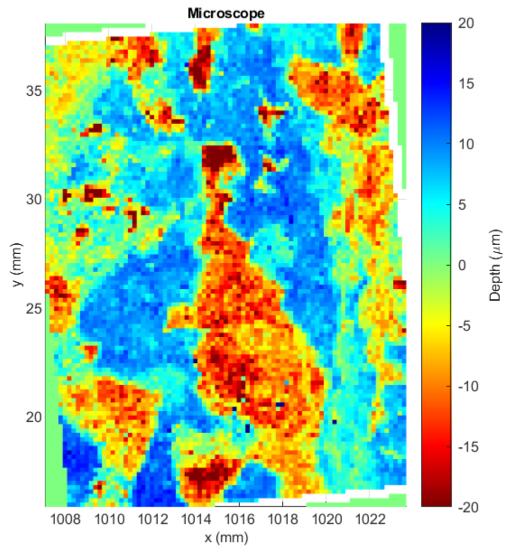
(c) Xcor

(d) Intersection

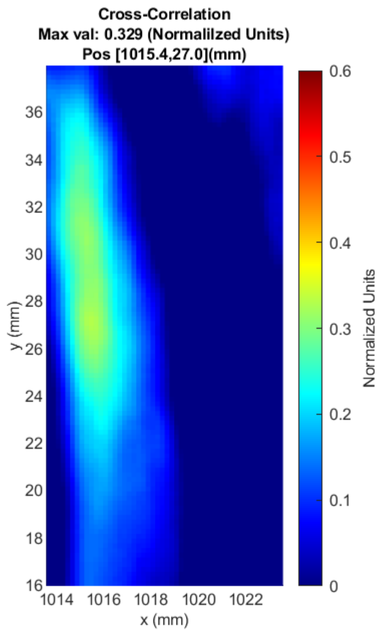
Figure 53 Convex sample c3, defect D3, fast scan.



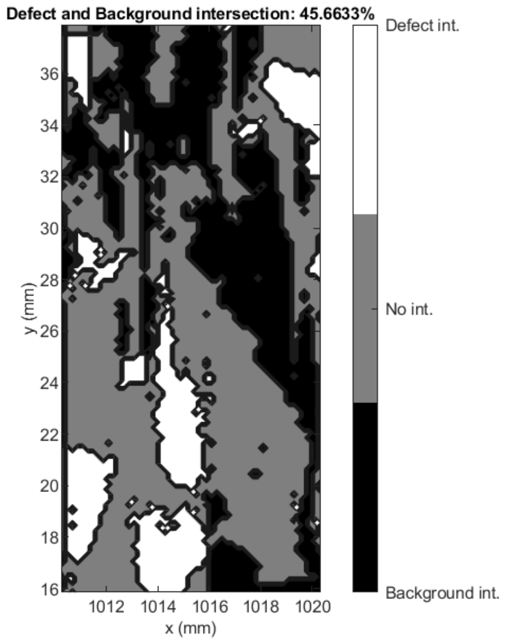
(a) ECA



(b) VHX

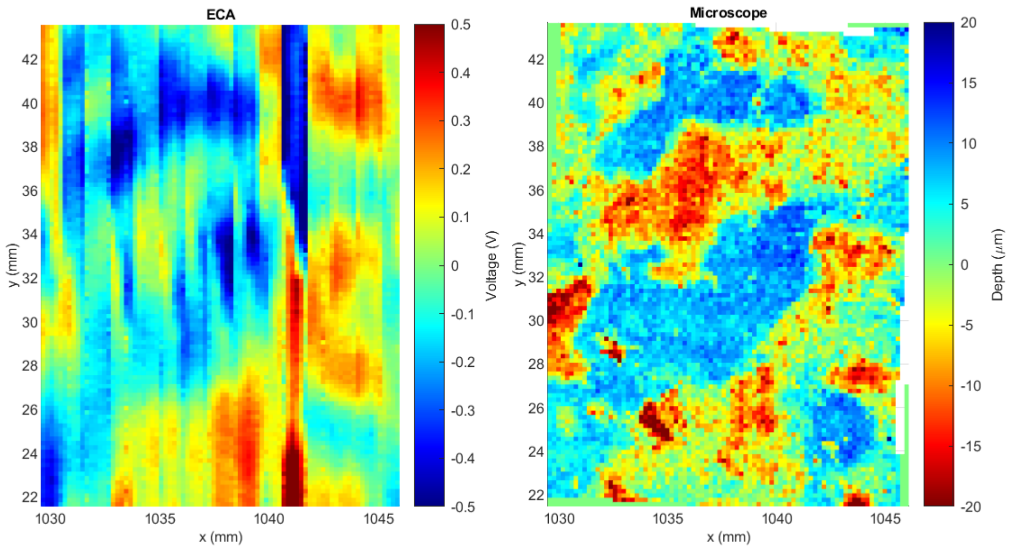


(c) Xcor

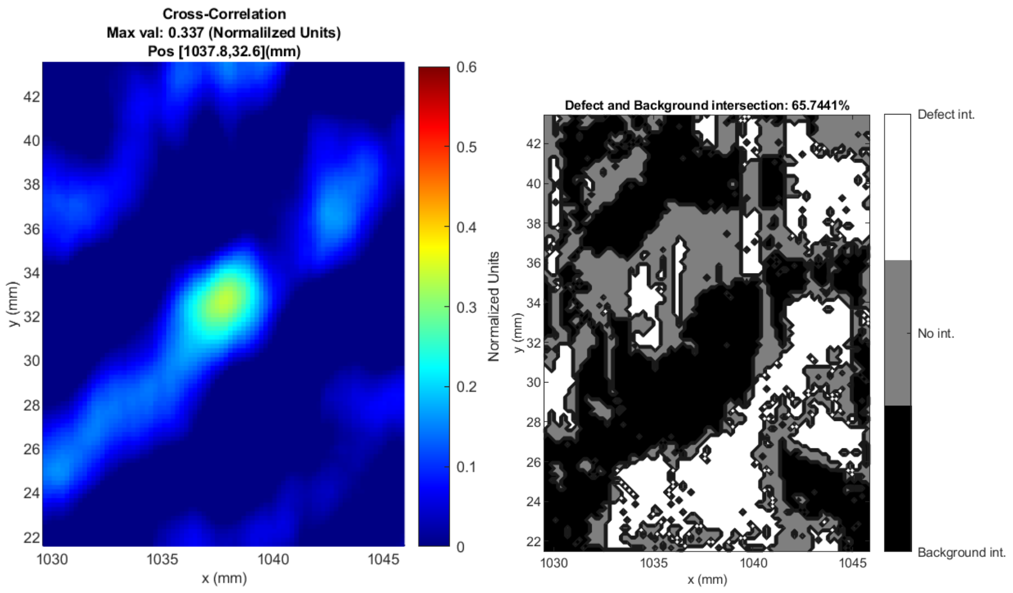


(d) Intersection

Figure 54 Convex sample c4, defect D1, fast scan.

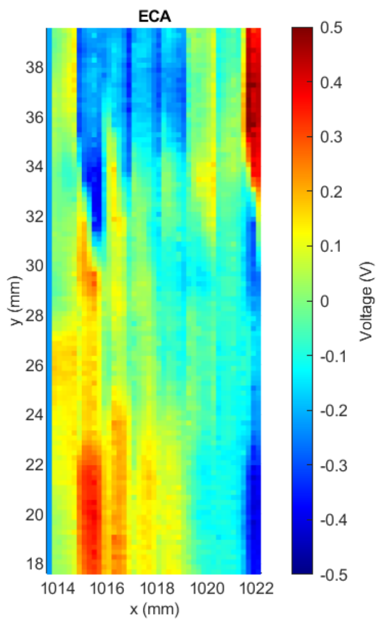


(a) ECA (b) VHX

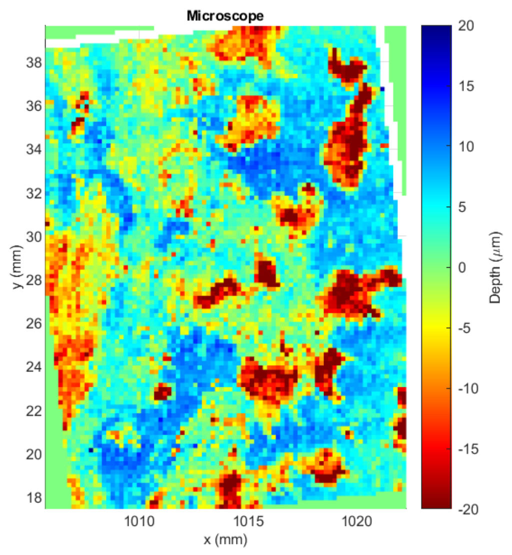


(c) Xcor (d) Intersection

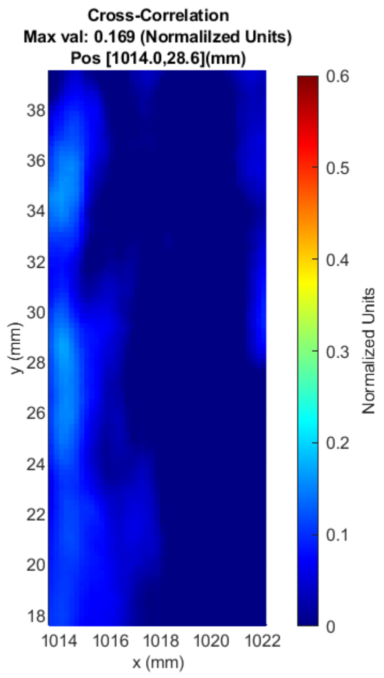
Figure 55 Convex sample c4, defect D2, fast scan.



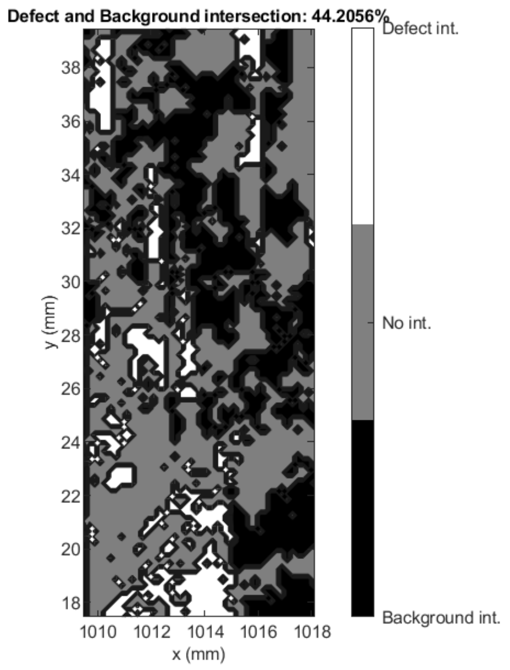
(a) ECA



(b) VHX



(c) Xcor



(d) Intersection

Figure 56 Convex sample c4, defect D3, fast scan.