Vision-guided robotic abrasion and water-break inspection of free-form composite panels

by

Bhavin Narendrakumar Dharia

A thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Mechanical Engineering
University of Manitoba
Winnipeg

Abstract

Carbon fiber and fiberglass composite panels are commonly used in the aerospace and automotive industries. During the manufacturing process, composite panels undergo surface treatment processes such as priming and coating. For coating layers to adhere strongly to the surface, the part surface must be free of any contamination such as grease. Detection and removal of contamination is an important step in the manufacturing of composite panels, particularly in the aerospace industry. This thesis develops a novel vision-based framework to enable fully automated inspection and robotic abrasion of free-form composite panels.

The first aim of the thesis is to automatically locate arbitrarily placed composite panels within the robot workplace. The proposed method employs iterative closest point (ICP) registration technique for locating the part in the robot cell. An additional module based on vision-based error correction is developed in the thesis for improving the accuracy of part localization. The proposed part localization technique eliminates the need for manual calibration of part placement in each robot cycle.

The ASTM-F22 water-break test is widely used for detecting hydrophobic contamination on a surface. As the second aim in this thesis, vision-based algorithms are developed to allow for the automated detection (inspection) of hydrophobic contamination during the water-break test. The developed vision-based algorithms can successfully detect the layer of hydrophobic contamination on the free-form panel.

The third aim of this thesis is to automatically generate robotic abrasion tool paths to remove detected contamination. In the proposed framework, the panel surface is divided into multiple grids. The point clouds of each grid are first used to reconstruct continuous parametric B-spline surfaces. B-spline surfaces are then used to make an abrasion tool path for each grid. The developed vision-based inspection algorithm determines which grids contain contamination and therefore must be abraded.

To conclude, the overarching goal of this thesis is to develop a fully automated vision-based framework for the detection and removal (robotic abrasion) of contamination on free-form composite panels. The developed techniques have been successfully implemented and verified on a Kuka KR6 industrial robot equipped with 2D and 3D vision cameras.

Table of Contents

List of Figures

Nomenclature

Symbols

${}^{b}T_{c}$	Transformation matrix from camera to robot's base
R	Rotation matrix
r_{ij}	Rotation matrix's term at i^{th} row and j^{th} column
t: (tx, ty, tz)	Translation vector
(X, Y, Z)	3D coordinate point
(A, B, C)	Euler angles for sequence 'ZYX'
s	Scaling factor of pin hole camera model
p_c	2D pixel point of pin hole camera model
A	Camera matrix
(f_x, f_y)	Focal length
(c_x, c_y)	Optical center
P_w (X, Y, Z)	3D world point of pin hole camera model
(U, V)	Pixel coordinate point
(Xc, Yc, Zc)	3D point in camera coordinate system
k_1, k_2, k_3	Radial distortion coefficients
p_1, p_2	Tangential distortion coefficients
${}^{f}T_{c}$	Transformation matrix from camera to robot's flange
${}^{b}T_{chess}$	Transformation matrix from chessboard to robot's base
${}^{c}T_{chess}$	Transformation matrix from chessboard to camera
${}^{chess}T_{c}$	Transformation matrix from camera to chessboard

$^{b}T_{zivid}$	Transformation matrix from Zivid camera to robot's base
$^{f}T_{triton}$	Transformation matrix from Triton camera to robot's flange
K	ICP correspondence set
p	Corresponding points of target point cloud
q	Corresponding points of source point cloud
$E(^{b}T_{w})$	ICP objective function
$^{b}T_{w}$	Transformation from workpiece to robot's base
Q_{PC}	Source point cloud
P_{PC}	Target point cloud
$^{b}CP_{nominal}$	Nominal corner pose
$^{b}T_{nominalCP}$	Nominal corner pose matrix
$^{w}CP_{nominal}$	Nominal corner pose in terms of nominal workpiece location
$^{w}T_{nominalCP}$	Nominal corner pose matrix in terms of nominal workpiece location
$^{b}CP_{new}$	New corner pose
$^{b}T_{newCP}$	New corner pose matrix
$^{b}CP_{refined}$	Refined corner pose
$^{b}T_{refinedCP}$	Refined corner pose matrix
$^{newCP}T_{refinedCP}$	Transformation from refined corner pose to new corner pose
$^{b}T_{w_{refined}}$	Refined part localization transformation
$sub(U,V)$	Subtracted image
$g(U,V)$	Contrast and brightness edited image
α	Contrast gain
β	Brightness bias

(H, S, V)	Hue, saturation and value
P_{grids_b}	Grid corner's 3D location with respect to robot's base
P_{grids_w}	Grid corner's 3D location with respect to composite panel
$P_{grids_{zivid}}$	Grid corner's 3D location with respect to Zivid
ROI_{PC}	Point cloud of selected region of interest
$Grids_{PC}$	Point clouds of grids
$S(X)$	Implicit surface representation
$S(u, v)$	Parametric surface representation
(u, v)	Surface parameters
$C(u)$	Parametric curve
$N_{i,p}(u)$	p^{th} degree B-spline basis function
N	Unit surface normal
$S_u(u, v)$	Surface partial derivative w.r.t u surface parameter
$S_v(u, v)$	Surface partial derivative w.r.t v surface parameter
R_N^w	Rotation matrix from surface normal to nominal workpiece's frame
$(\Delta u, \Delta v)$	Parametric difference between two adjacent tool paths
F	Vector in feed direction
r	Euclidean distance between two adjacent tool paths
${}^bP_{nominal}$	Pose on nominal tool path
${}^bTP_{nominal}$	Pose matrix on nominal tool path
${}^bP_{new}$	Pose on new tool path
${}^bTP_{new}$	Pose matrix on new tool path

Subscripts

b	Robot's base frame
c	Camera's frame
f	Robot's flange frame
$chess$	Chessboard pattern frame
PC	Point cloud
w	Workpiece's frame
CP	Corner pose
TP	Tool path

Acronyms

ASTM	American Society for Testing and Materials
DOF	Degrees of Freedom
2D & 3D	Two-Dimensional & Three-Dimensional
ICP	Iterative Closest Point
B-Spline	Basis Spline
CNC	Computer Numerical Control
DXF	Drawing Exchange Format
IR	Infrared Radiation
CAD	Computer Aided Design
RCNN	Region-Based Convolutional Neural Network
SVM	Support Vector Machine
FCN	Fully Convolutional Network
CFRP	Carbon Fiber Reinforced Polymers

NURBS	Non-Uniform Rational B-Spline
RGBD	Red-Green-Blue-Depth
RGB	Red-Green-Blue
TCP	Tool Center Point
KRL	Kuka Robot Language
EKI	Ethernet-KRL-Interface
XML	Extensible Markup Language
SVD	Singular Value Decomposition
ROI	Region of Interest
CMOS	Complementary Metal-Oxide Semiconductor
LED	Light-Emitting Diode
CFL	Compact Fluorescent Lamps
HSV	Hue-Saturation-Value

Acknowledgment

I would like to thank my research advisor, Dr. Matt Khoshdarregi, for his invaluable guidance throughout my studies. This project would not have been possible without his mentorship. I feel lucky and proud to call him my research advisor. He has been an ideal mentor and thesis supervisor.

I would also like to express my gratitude to my friends at the Intelligent Digital Manufacturing Laboratory for helping and encouraging me throughout my Master's Studies. Special thanks to Ali Maghami, Michael Newman, Shreyans Dhariawala, and Sina Alborzi for staying with me in the lab for long hours.

This research was financially supported by the MITACS Accelerate program. This program helps students find industrial internships and give them the opportunity to innovate in and contribute to real applications.

Finally, I would like to thank my parents and my wife. I would not be where I am today without their precious love and support.

1 Introduction

1.1 Background and motivation

Carbon fiber and fiberglass composite panels are widely used in the aerospace and automotive industries, e.g., in the fuselage of airplanes and body of vehicles. Many surface treatment processes such as priming, coating, and painting are performed on panel surfaces to improve their functional and aesthetic characteristics. The presence of hydrophobic contamination such as oil, grease, and silicon can prevent strong and uniform adhesion of the coating to the surface. In order for the coating to adhere strongly to the surface, the surface must be highly "hydrophilic" (attracted to water). To achieve this goal, two important steps are commonly performed prior to coating:

I. detection of hydrophobic contamination such as greasy patches,
II. abrasion (sanding) of the surface to remove contamination and to create micro-scratches that coating particles can stick to.

The American Society for Testing and Materials F22 (ASTM-F22) water-break test is a commonly used technique to identify hydrophobic contamination on the surface of panels. As illustrated in Figure 1.1, the water-break test is used to perform a go/no-go surface contamination inspection.

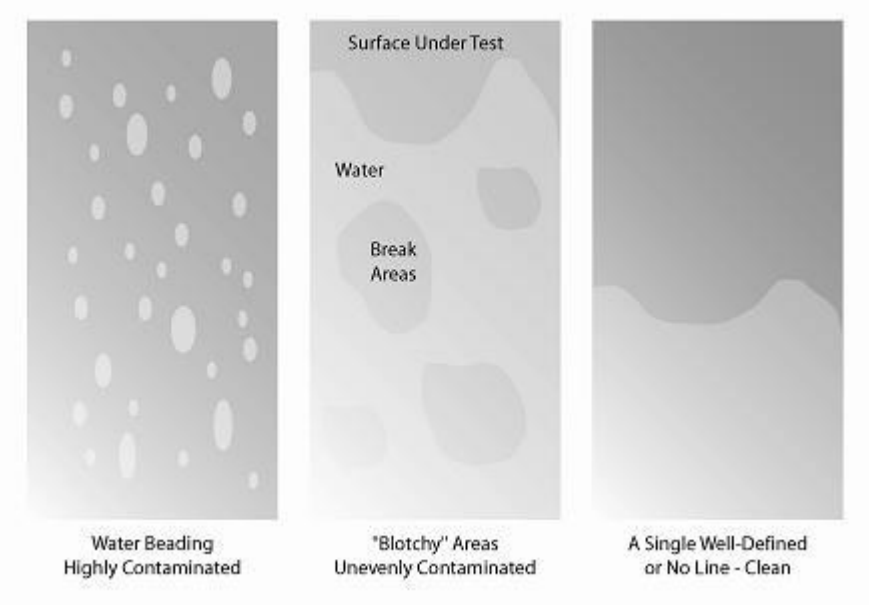


Figure 1.1: Schematics of water-break test results and interpretations [1]

According to the ASTM-F22 standard, during the water-break test, the workpiece is placed vertically and is doused with water. The water flowing down on the workpiece's surface can continue flowing without breaking if there is no presence of hydrophobic contamination on the surface. Hence, the broken region of water on the surface indicates the presence of hydrophobic contamination. The failed regions on the surface must be abraded to remove contamination.

Currently, the water-break test in most industries is performed manually using naked eye. Manual water-break inspection is a challenging task and highly susceptible to human error. During the water-break test, the surface must be monitored for 10-40 seconds to find areas where the water layer breaks. It can be difficult to manually keep track of the entire part over the given time frame by relying solely on human observation. This research aims to automate the process of water-break inspection using a vision system. The entire inspection process is captured by a camera, and then image processing algorithms are used to automatically detect contamination on the surface.

At present, after the detection of contamination with the help of the water-break test, the contaminated area is abraded manually to remove grease and create a hydrophilic surface. Manual abrasion is a labor-intensive and time-consuming process. As a result, it can interrupt

the rapid flow of parts through the production line. Moreover, manual abrasion may expose workers to hazardous chemicals such as composite particles. To tackle these challenges, industrial robotic arms can be used to automate the abrasion process.

However, a challenge in using industrial robotic arms is that they can only perform preprogrammed commands for calibrated location of the workpiece. Robotic arms cannot automatically account for the variations in the robot cell. The location of the workpiece with respect to the robot needs to be recalibrated for every part before the robot performs the desired operation on the workpiece. Another challenge is to generate an abrasion path that the robot must follow. This path depends on the location and size of the contaminated area. This thesis develops a methodology that automatically locates the workpiece and generates the abrasion tool path based on the detected failed regions captured by the vision system.

The remainder of this chapter is divided into two sections. Section 1.2 presents the problem statement and research objectives. Section 1.3 presents the general elements of the developed framework for autonomous robotic inspection and abrasion of composite panels.

1.2 Problem statement and research objective

The main goal of this thesis is to automate the process of abrading and inspecting composite panels prior to surface treatment steps such as painting and coating. This thesis employs a 6-DOF industrial robotic arm equipped with vision cameras to automate both the inspection and abrasion processes. To achieve this goal, three main problems (objectives) must be addressed, as summarized in Figure 1.2. Assuming that the panel is placed arbitrarily in front of the robot (which is often the case in fixtureless manufacturing settings), the first objective is to automatically and accurately localize (i.e. find the location and orientation of) the panel in the robot cell. The second objective is to autonomously detect hydrophobic contamination on the workpiece's surface during the water-break test. The third objective is to generate a smooth tool path for abrading the failed regions to remove contamination.

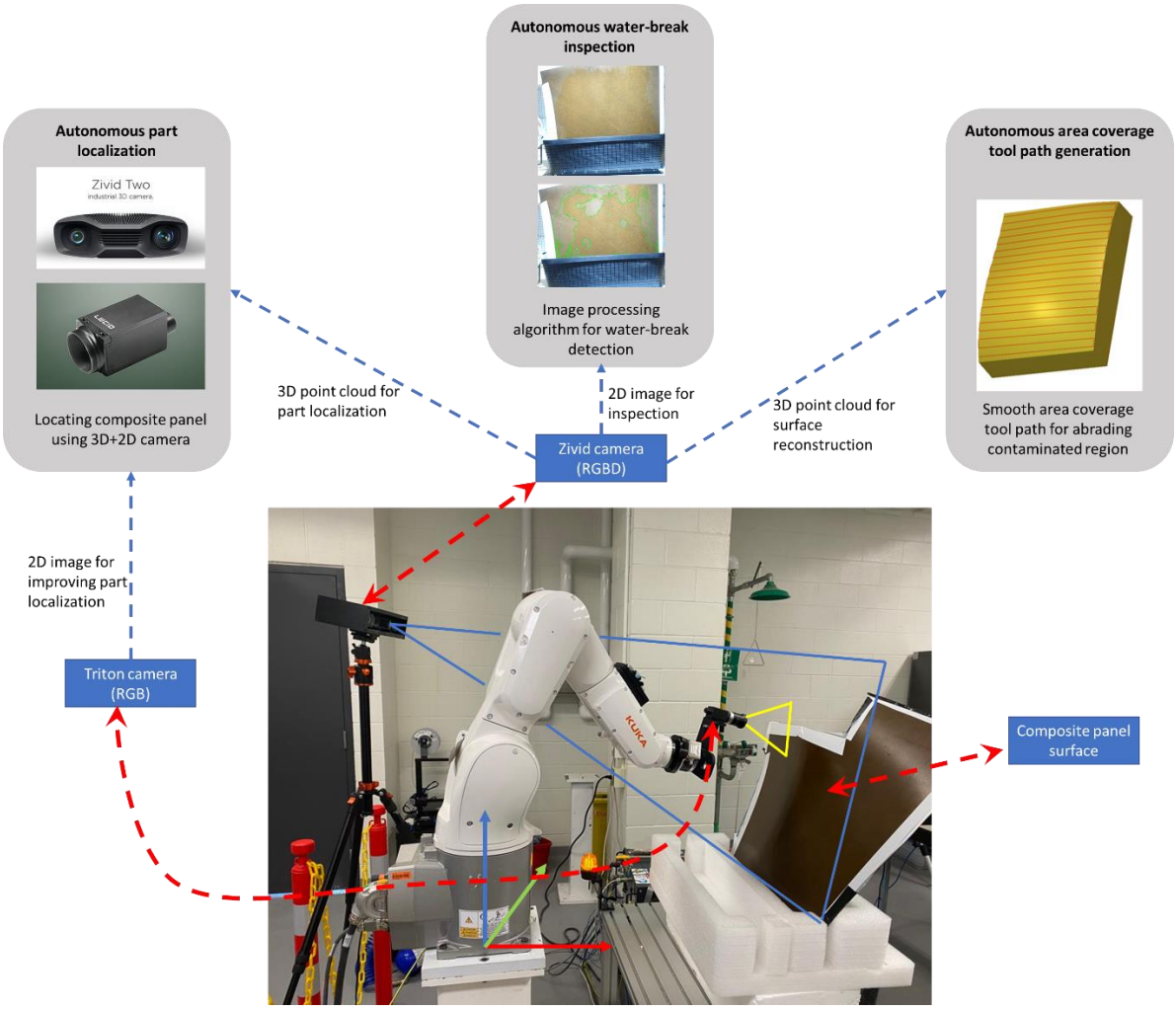


Figure 1.2: Overview of application and research objectives

1.3 Proposed solution and approach

This thesis develops a machine vision framework and the corresponding vision processing algorithms to automate the surface pretreatment process. The body of the thesis is divided into two main parts (chapters) that discuss the proposed solution in detail. The first part of the thesis, which is presented in Chapter 3, focuses on vision system calibration and automated part localization (Objective 1). The second part, which is explained in Chapter 4, presents the developed algorithms and methodology for automated water-break inspection

(Objective 2) and tool path generation (Objective 3). The contents of these two parts are outlined below in more detail.

Part 1: Vision system calibration and automated part localization

For any robotic system that uses machine vision, it is important to know the position and orientation of the vision sensors with respect to the robot's location. In the field of precision manufacturing, the location of each component of the machine cell is defined using a 3D coordinate system. This thesis proposes a part localization framework based on information from a 2D and 3D camera. Specifically, in this work, a LUCID Triton (Figure 1.3) has been used for 2D imaging, and a Zivid Two (referred to as Zivid in this thesis) has been used for capturing 3D point clouds.

Figure 1.3: Triton 2D camera (left) [2] **and Zivid Two 3D camera (right)** [3]

As explained in Chapter 3, the pin hole camera model is used to assign a 3D coordinate system to vision cameras. Then, a three-point hand eye calibration method is proposed that can obtain the transformation between the robot's coordinate system and the camera's coordinate system. The process of assigning a 3D coordinate system to the camera and finding the transformation between the camera and the robot is known as vision system calibration. This step is a prerequisite for integrating machine vision cameras in the robot cell used in this thesis.

Deploying robotic arms for performing machining operations such as abrasion needs precise information of the workpiece location with respect to the robot's coordinate system. Traditionally, a workpiece is localized in the robot's coordinates by manually touch probing the workpiece using the robot's end effector (Figure 1.4). Manual workpiece calibration is time-consuming and can increase the downtime of the robot cell.

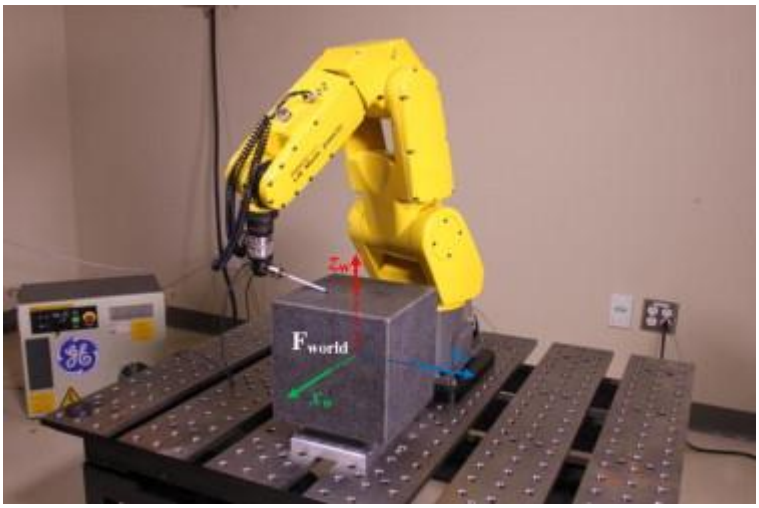


Figure 1.4: Manual touch probing part for workpiece location calibration [4]

Instead of manual workpiece calibration, this thesis proposes an automated framework that uses 2D+3D vision cameras for locating the workpiece in the robot's coordinates. The point clouds of the composite panel are captured using a Zivid camera, and Iterative Closest Point (ICP) algorithms are used for point registration. A vision-based workpiece corner detection algorithm is proposed for improving the accuracy of the part localization. The proposed part localization technique ultimately provides the transformation between the robot's and workpiece's coordinate systems (frames). Hence, the robot can perform desired operations on the part knowing the part's location relative to the robot.

Part 2: Automated water-break inspection and tool path generation

For performing the water-break test, water is sprayed over the surface of the composite panel. Images of water flowing downwards on the composite panel's surface are captured using the Zivid camera during the water-break test. Image processing algorithms are proposed to automatically detect the water-broken regions within the captured images.

Regions of the composite panel that contain water-broken (failed) areas need to be abraded. This thesis proposes to divide the surface of the panel into multiple grids. Hence, any grid that contains water-broken areas is abraded. For creating abrasion tool path for each grid, the point clouds of the composite panel and all grids are first converted to smooth basis-spline (B-spline) surfaces. Then, an algorithm for generating area scanning abrasion tool paths for B-spline surfaces is developed.

1.4 Thesis layout

The rest of this thesis is organized as follows. Chapter 2 presents the literature review and prior work related to automated part localization, vision-based inspection, and robot path generation. Chapter 3 discusses the vision system calibration and automated part localization in detail. Chapter 4 presents the automated water-break inspection and tool path generation. Finally, the findings of the thesis and future work are summarized in Chapter 5.

2 Literature Review

2.1 Overview

This chapter presents existing literature in the areas relevant to part localization, inspection of composite panels, and vision-guided robotic abrasion. The literature on part localization is reviewed in Section 2.2. Section 2.3 provides the prior art in the field of water-break inspection as a method for detecting contamination. Section 2.4 presents past research related to vision-based inspection. Finally, the literature on automated tool path generation for robotic treatment of free-form surfaces is reviewed in Section 2.5. Section 2.6 summarizes the chapter.

2.2 Part localization

In order for a robot or a machine to perform an operation on a part, the location and orientation of the part with respect to the machine coordinate system must be accurately known. Determining the workpiece coordinate system with respect to machine's coordinate system is referred to as part localization. Commonly, the workpiece is localized in the machine coordinate by manually touch-probing the part at multiple locations. There are several commercial touch probing systems available in the market, e.g. probing systems and software by Renishaw [5], Marposs [6], Heidenhain [7], and Mitutoyo [8]. However, as investigated by Srinivasan et al. [9], there are some challenges in using touch probe-based part localization systems. For example, they require extensive human intervention and manual decision making, which is a time-consuming process and leads to machine down time.

To automate the part localization problem, several researchers have employed machine vision. For example, Srinivasan [9] developed an automatic part localization system using 3D scans for Computer Numerical Control (CNC) machines. Zheng et al. [10] used a 2D monocular vision system for automated part localization in robotic grasping. Skotheim [11]

developed a part localization technique for material handling using 3D machine vision sensors. Rajaraman et al. [12] developed a vision-based part localization system for robotic welding, and Okarma et al. [13] used a 3D scanning system for workpiece positioning on CNC machines. The nature of the workpiece-placement setup defines the preference of the type of machine vision system to be used for part localization. For a setup where the part can only have translational placement error, a 2D image-based system is preferred. A 3D point cloud-based system, on the other hand, is preferred if the workpiece can have orientation placement errors as well.

Hernandez et al. [14] proposed a 2D machine vision framework for automatically recognizing a part and extracting its geometrical information from an existing library of Drawing Exchange Format (DXF) files. As an example of a 3D point cloud-based system, Fan et al. [15] developed a point cloud registration technique for localizing a part. They presented a point cloud-to-CAD model registration method based on edge matching.

Besl et al. [16] proposed the well-known Iterative Closest Point (ICP) method. The ICP method is a computationally efficient point cloud registration method. Since the part used in this thesis is a free-form surface, and the placement setup can have both position and orientation errors, this work uses the ICP method for initial part localization. A 2D vision-based system is then incorporated in the framework to improve the accuracy of the part localization in this thesis.

2.3 Water-break test

The water-break test is based on the fact that a contaminant-free surface should be able to hold a continuous water film without any breaks. On the contrary, a contaminated surface exhibits water-broken regions instead of a continuous film layer [17]. The water-break test is a widely used method for organic surface contamination detection and has been used in the industry for many years. Despite being referred to as a “nineteenth century approach” [18] , it is still used as the standard method for contamination detection in the aerospace industry.

The ASTM-F22 [19] standard provides a step-by-step procedure for performing the water-break test for detecting the hydrophobic contamination on a hydrophilic surface. The standard also suggests that the presence of hydrophobic contamination can prevent the adhesive bond of surface treatments like coating, priming, and painting.

Despite the fact that the water-break test is a widely accepted method for hydrophobic surface contamination detection, it has also faced some criticism. Ellis [20] argues that there are too many variables in the water-break test, e.g. surface roughness, presence of abrasive particles and their nature, and smearing from abrasive brushes. Monzyk et al. [21] state that the water-break test is extremely slow and limited to small workpieces. Kolenov et al. [22] acknowledge that the water-break test provides satisfactory results, but they find the test to be very time-consuming and expensive for large parts. Similarly, Ecault et al. [23] found the water-break test to be laborious for large parts.

This thesis aims to mitigate the issues in utilizing the water-break test on large surfaces by developing a fully automated water-break inspection system using machine vision. Amos et el. [24] in their patent discussed the potential hazards of manually inspecting the water-break test. Their patent develops an apparatus for detecting water-broken regions using a long infrared radiation (IR) range camera. As a motivation from their patent, the effects of near IR lights for detecting water-broken regions have been tested in this thesis.

The review suggests that the manual water-break test is slow and laborious. Therefore, this thesis develops a vision-based system and associated image processing algorithm that can automate the water-break detection process.

2.4 Vision-based inspection

Vision-based inspection is a non-destructive inspection method that uses vision sensors such as digital cameras and 3D scanners. Vision-based inspection relies on the visual appearance and shape of the surface defects that need to be detected. Vision-based inspection has been used widely in a range of applications. For example, Traband et al. [25] provided a foundation for research in the area of Computer Aided Design (CAD) directed inspection using solid-state cameras. Chen et al. [26] developed three-part machine vision algorithms

comprising segmentation, recognition, and analysis for automated inspection of production parts. Wang et al. [27] used back propagation neural networks for developing a machine vision algorithm for defect detection in textile fabrics. Bradley et al. [28] used neural network classifiers for developing an automatic inspection system for manufactured parts. They used a 3D vision system for part identification and part dimensional inspection. Koch et al. [29] presented the achievements and challenges in implementation of vision-based inspection systems for inspecting large concrete structures. Li et al. [30] proposed a multi-layer feature fusion network based on the faster region-based Convolutional Neural Network (Faster RCNN) to design an automatic Metro Tunnel Surface Inspection System. Menendez et al. [31] developed a preliminary system for inspecting the state of transmission lines based on detection of wires through artificial vision. Bong et al. [32] developed a "Support Vector Machine" (SVM) based algorithm to classify the type of defects on the surface of leather fabric. Maghami et al. [33] designed and trained a deep fully convolutional network (FCN) with the U-NET architecture for detecting damages and cracks in drilled holes in aerospace carbon fiber reinforced polymers (CFRPs).

Most of the research that uses 2D digital camera for inspection performs image processing as a preliminary step (for enhancing the image quality) or as the main algorithm for segmenting the defects. The review also suggests that researchers are employing neural networks for detecting defects in the inspection process. However, neural networks need a lot of training data. This thesis develops an image processing algorithm for segmenting the pixel coordinates of the water-broken regions that represent hydrophobic contamination on a composite's surface. Although this thesis has not employed neural networks for automating the inspection process, the detected pixel coordinates of the defects using the proposed image processing algorithm can be used for labeling training data for training the neural network.

2.5 Surface reconstruction and tool path generation

The problem of point cloud to 3D surface reconstruction falls under reverse engineering. Reverse engineering is the process of generating a geometric CAD model from scanned 3D points of an existing part [34]. As investigated by Abella et al. [35], reverse engineering has

evolved from a skilled manual process to an engineering tool using sophisticated software and measuring instruments [36]-[37]. The steps for reverse engineering a mechanical part is to first capture the point cloud data, then preprocess the point cloud for surface fitting, interpolate the surface using the processed points, and finally create a CAD model using the fitted surface. The literature for mathematical modelling of a free-form surface using point cloud data is reviewed in the following.

The mathematical modelling of a free-form surface falls under the problem of 3D reconstruction. 3D reconstruction is a process of reproducing a digital representation of an object in a computer. Non-Uniform Rational B-Spline (NURBS) [38] is one of the most widely used surface fitting models in the field of 3D reconstruction [39]. Several researchers have proposed different methods for reconstructing a surface using NURBS especially for unorganized point clouds. For example, Wu et al. [40] presented a parametric surface modelling method for accurately fitting tea leaf point clouds. Gálvez et al. [41] proposed new iterative two-step genetic-algorithm-based method for B-spline reconstruction. Leal et al. [39] presented a regression plane projection-based method for B-spline surface reconstruction from unorganized point cloud. The literature suggests that B-spline surface construction can be complex for unorganized point cloud. To avoid this issue, this work uses a 3D camera that outputs an organized point cloud. The organized point cloud can readily be used as the set of control points for generating desired B-spline surfaces.

Once a 3D surface is reconstructed from a captured point cloud, the next task is to generate an area scanning abrasion tool path for free-form surfaces. Area coverage machining tool paths for a free-form surface can be categorized as iso-parametric and iso-scallop tool paths [42]. Iso-parametric tool paths possess several advantages. For example, they require simpler calculations and are therefore more computationally efficient. However, the uneven 3D distance between iso-parametric curves defined on a free-form surface may cause redundant or under-cut machining [43]. Can et al. [44] proposed a novel iso-scallop tool path generation method for efficient five axis free-form surface machining. Randhawa et al. [45] focused on an algorithm that generates tool paths for free-form surfaces. Their work includes two components; first, it determines the maximum distance between two cutter

points. The second component then determines the maximum distance between two adjacent tool paths. In contrast to the iso-parametric and iso-scallop based tool path generation methods, Liu et al. [42] proposed a new free-form surface machining tool path generation method by introducing the tensor property of machining width.

Most of the methods mentioned above produce a geometrically uniform coverage tool path. Han et al. [46], [47] proposed a tool path planning method for physically uniform coverage instead of geometrically uniform coverage of polishing path. This work leverages Han's findings for generating the area covering tool paths for abrading free-form surfaces.

2.6 Summary

Prior literature on part localization, vision-based inspection, and abrasion path generation is reviewed in this chapter. Common issues with traditional touch probe-based part localization methods are presented, and 2D and 3D vision-based methods for part localization are studied. The reviewed literature motivated the point cloud registration-based part localization method used in this thesis. As a contribution, a novel 2D vision-based framework is developed in this thesis for improving the accuracy of part localization (Chapter 3).

Water-break test and vision-based inspection techniques are reviewed. It is found in the literature that the water-break test is difficult to implement for inspecting large surfaces. To tackle this challenge, this thesis presents a framework for autonomous vision-based contamination detection for water-break test. The concepts of 3D reconstruction and tool path generation are introduced. Different 3D reconstruction methods are reviewed. Different approaches for generating tool paths for free-form surfaces are studied. The reviewed literature motivated the method for generating an area scanning tool path developed in this work (Chapter 4).

To conclude, an autonomous vision-based framework for part localization, detection of hydrophobic contamination on composite panels, and abrasion path generation is contributed.

3 Vision system calibration and automated part localization

3.1 Introduction

The first step in the integration of any machine vision system is to calibrate the vision sensors. Digital cameras must first be calibrated to find their intrinsic characteristics such as focal length, optical center, and distortion coefficients. This first step is known as intrinsic calibration. Intrinsic parameters are used to assign a 3D coordinate system to the camera. Once the intrinsic parameters are found, the next calibration step is to find the so-called extrinsic parameters. Extrinsic parameters determine the location of the camera with respect to the machine. In a robot cell, extrinsic parameters encompass the rotational and translational transformation of the camera's coordinate system with respect to the robot's coordinate system. The process of localizing the workpiece coordinate system with respect to the machine's coordinate system is known as part localization. The current practice of touch probing the workpiece for localization is time consuming. Hence, an automated technique is needed for efficient and fast part localization.

This chapter is organized as follows. The Kuka robot cell used as the experimental platform in this research is first introduced in Section 3.2. Section 3.3 presents the convention for establishing the robot coordinate system. Section 3.4 discusses the homogeneous transformation matrix and its relationship with the pose format. The socket communication with Kuka robot is discussed in Section 3.5. Calibration of the vision system is presented in Sections 3.6 and 3.7. Sections 3.8 and 3.9 present the proposed method for accurate vision-based part localization. Finally, Section 3.10 summarizes the chapter.

3.2 Experimental Platform: Kuka robot cell

Figure 3.1 shows the robot cell used in this research. The cell comprises an industrial 6-DOF robotic arm (Kuka KR6-R700-2), a 3D colored point cloud camera (Zivid, RGBD), a 2D camera (Triton, RGB), and a free-form composite panel (Workpiece).

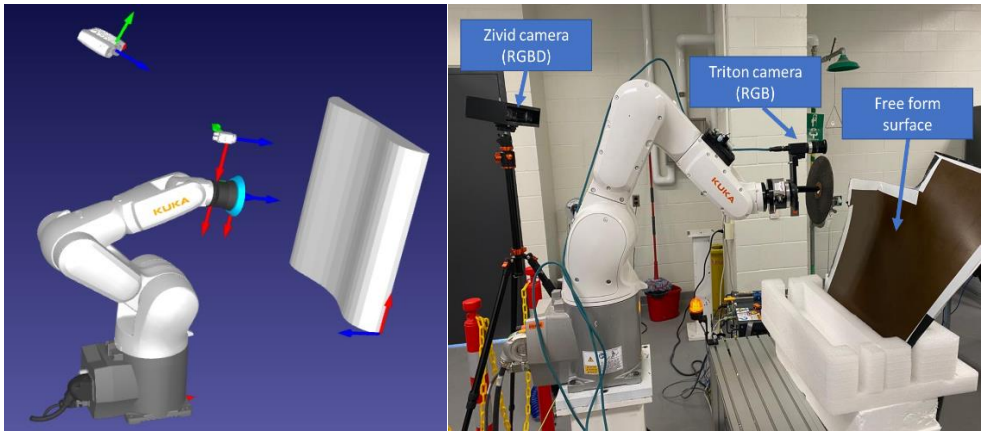


Figure 3.1 Experimental platform: Kuka robot, cameras, and composite panel

The Zivid camera can capture 2D color information (RGB color image) as well as 3D point cloud (X, Y, Z) information. The point cloud provided by the Zivid camera is organized and mapped on to the 2D image color channel. This means that the information of each color pixel of image has 3D position data. The point cloud captured by the Zivid camera is used for the part localization and tool path generation aspects of this thesis. The 2D color image captured by the Zivid camera is used for automatic water-break inspection. As shown in Figure 3.1, the Zivid camera is held stationary beside the robot.

The sanding tool along with the 2D color Triton camera are attached to the robot's flange. This 2D color camera can be moved around by the robot and is used for improving the part localization accuracy, as discussed in Section 3.9. The composite panel is placed arbitrarily in front of the robot while ensuring the panel is within the view and reach of the cameras and the robot. The goal is for the robot to automatically localize the panel, inspect it during the water-break test, and abrade the failed regions (i.e. contaminated areas).

3.3 Establishing coordinate systems in the robot cell

In robotic applications, the location and orientation of the workpiece with respect to the robotic arm must be known accurately. The pose, i.e. position and orientation, of the workpiece is defined in the Cartesian coordinate system of the robotic arm. Therefore, it is

important to understand the conventions and placement of the coordinate system of the robot. As illustrated in Figure 3.2, Kuka robots by convention have four coordinate systems, i.e., robot's root coordinates, world coordinates, tool coordinates, and base coordinates.

The robot's root coordinate system is generally located at the center of the base of the robot. The world coordinate system is assumed fixed at an arbitrary location in the robot cell. By default, the world coordinate system is chosen the same as the robot's root coordinates. The origin of the tool coordinates can be defined by the operator and is kept at the tool center point (TCP). Sometimes, it is easier for a robot programmer to define the waypoints of the robots with respect to the workpiece's position in the scene. For defining the workpiece coordinate system in the robot cell, the base coordinate system is used. The base coordinate system can be defined manually by the robot operator according to the position of the workpiece. It is defined in the form of a transformation with respect to the world coordinate system.

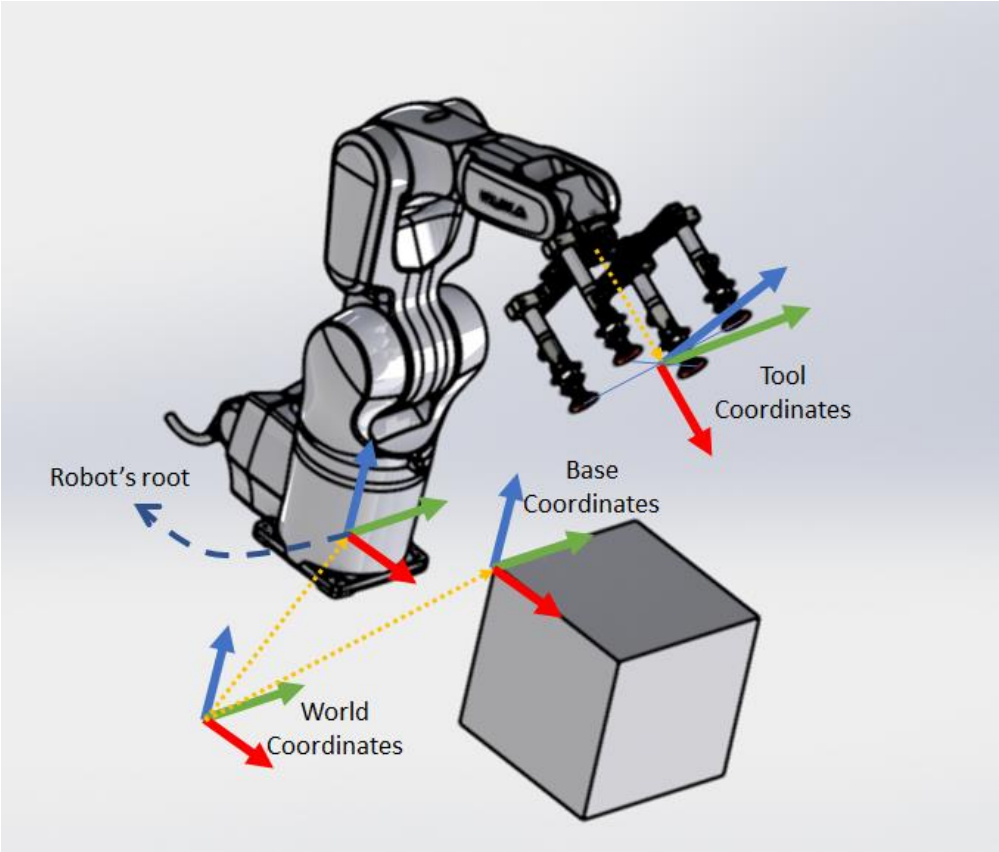


Figure 3.2 Generalized coordinate systems in a robot cell

As shown in Figure 3.2, the robot root's coordinates and the base coordinates are defined with respect to the world coordinates. The tool coordinate system, on the other hand, is defined with respect to the flange of the robot.

To simplify the transformations, and without loss of generality, this thesis assumes that the base coordinates and the robot's root coordinates coincide with the world coordinates. Moreover, the reference coordinate system of the free-form composite panel is also assumed coincident with the robot's root coordinates.

3.4 Transformation matrix and pose representation

Transformation matrices are used extensively in this work to map information (e.g. location data) between different coordinate frames. Therefore, this section briefly discusses the homogeneous transformation matrix and its conversion to the pose format. A homogenous transformation matrix is a 4x4 matrix that describes the rotation R and translation t between two coordinate systems. As illustrated in Figure 3.3, in robot vision applications we have two crucial coordinate systems: robot's base coordinate system, b, and camera's coordinate system, c. For a robot to perform any operation using the camera's view, all the points described in the camera's coordinate system must be transformed to the robot's base frame. In this work, homogenous transformation matrices are represented by ${}^{**}T_{*}$, where the subscript (*) represents the transformation frame from which the points are transformed, and the upper prefix (**) denotes the transformation frame where the points are transformed to. For example, the homogeneous transformation matrix ${}^{b}T_{c}$ maps points expressed in the camera's coordinate system c to the robot's base frame b.

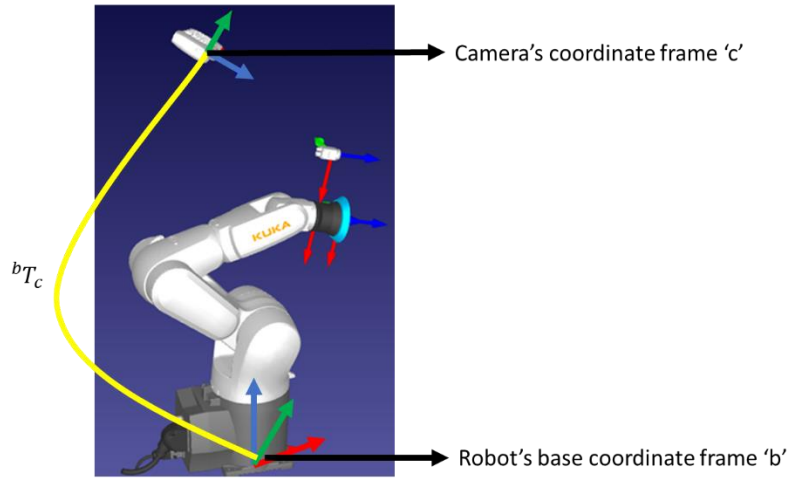


Figure 3.3: Frame transformation between robot and camera coordinates

$$ {}^{b}T_{c} = \begin{bmatrix} r11 & r12 & r13 & tx \\ r21 & r22 & r23 & ty \\ r31 & r32 & r33 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & R & & t \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad 3.1 $$

As seen in Eq. 3.1, the transformation matrix ${}^{b}T_{c}$ consists of 12 variables, out of which 9 elements ($r11,\ r12, \dots, r33$) represent the rotation R and 3 elements (tx, ty, tz) represent the translation t of the camera's frame relative to the robot's base frame.

An alternative way to represent the transformation between two frames is using 'pose'. The pose format is resented by only 6 elements as:

$$ Pose = (X, Y, Z, A, B, C), \quad 3.2 $$

where, X, Y, Z represent the 3D translation between two frames (equal to tx, ty, tz). A, B, C represent the 3D rotation between the two frames in terms of Euler angles. The Kuka robot uses the pose data with Euler angles of sequence 'ZYX'. Hence, A is the rotation along the Z-axis, B is the rotation along the Y-axis, and C is the rotation along the X-axis. Coordinate frame transformations are typically performed in the matrix format (Eq. 3.1), while the robot's controller accepts the transformation data in the pose format. Hence, it is important to know the relationship between the two formats.

If the transformation matrix is known, the translation part (X, Y, Z) of the pose format is the same as the *t* terms (tx, ty, tz). The formulation for the rotation part (A, B, C) of the pose format can be obtained as:

$$Euler\ angles \begin{cases} A = \arctan\left(\dfrac{r21}{r11}\right) \\ B = \arctan\left(\dfrac{-r31}{\sqrt{1 - r31^2}}\right) \\ C = \arctan\left(\dfrac{r32}{r33}\right) \end{cases}, \tag{3.3}$$

where the r_{ij} terms are the rotation terms as presented in Eq. 3.1. Conversely, if the transformation pose (X, Y, Z, A, B, C) is known, the formulation of the transformation matrix is given by:

$$T = \begin{bmatrix} \cos(A)\cos(B) & \cos(A)\sin(B)\sin(C) - \cos(C)\sin(A) & \sin(A)\sin(C) + \cos(A)\cos(C)\sin(B) & X \\ \cos(B)\sin(A) & \cos(A)\cos(C) - \sin(A)\sin(B)\sin(C) & \cos(C)\sin(A)\sin(B) - \cos(A)\sin(C) & Y \\ -\sin(B) & \cos(B)\sin(C) & \cos(B)\cos(C) & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.4}$$

Similar to the notation of the transformation matrix ${}^{**}T_{*}$, the upper prefix (target frame) and subscript (original frame) in the pose format are shown as $({}^{**}Pose_{*})$.

3.5 Kuka socket: communication between robot and vision system

Robot paths are provided to the robot controller in terms of waypoints. Waypoints define a series of linear motions that the robot needs to follow. The Kuka Robot Language (KRL) is the programming language used to control the Kuka robot. It is a proprietary programming language developed by the manufacturer, Kuka Robotics. Kuka robots provide a socket communication known as the Ethernet-KRL-Interface (EKI) module, which allows direct communication with the robot controller. The EKI module is a networking framework specifically made to enable external devices to communicate with the Kuka controller via an Ethernet connection.

As shown in Figure 3.4, there are three main components in the EKI framework, i.e., the server, the client, and the Extensible Markup Language (XML) configuration file. In socket communication, the server is the component that listens for the client's requests. In this research, the robot's controller is the server, and a python script which runs the developed vision processing algorithms is the client. The Ethernet connection is configured by an XML file that is stored in the robot's controller. This XML file is used to define the type of connection and the format of XML socket messages that are sent and received within the network.

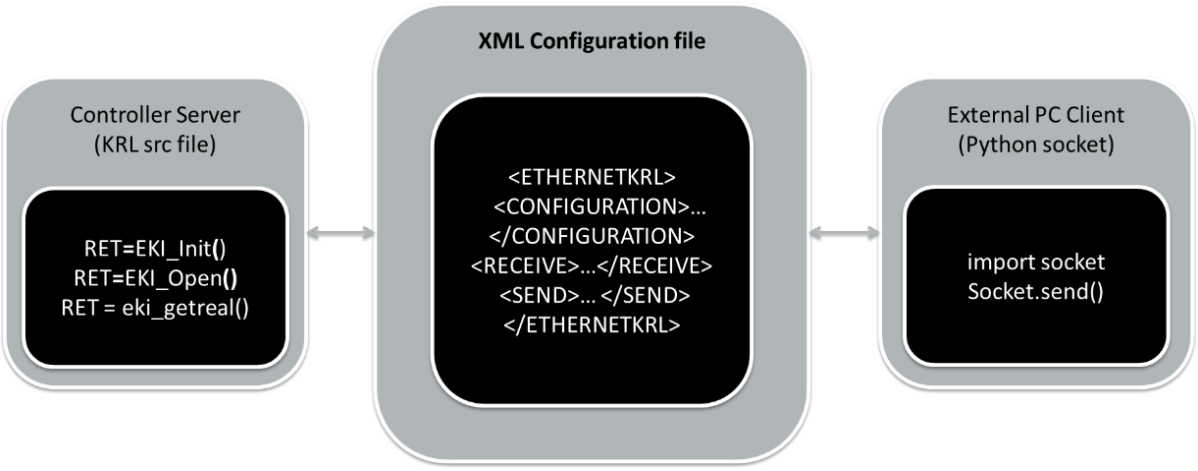


Figure 3.4: Kuka EKI networking socket for robot-vision communication

Considering the extensive use of robot-vision communication in this thesis, a python library for communicating with the Kuka controller via EKI was first developed. Table 3.1 presents the supported functions of the developed Kuka-python library.

Table 3.1: Supported functions in the Kuka-python library

Name	Function	Description
Linear motion	move_lin(pose)	This function sets the robot command for performing a linear motion through a series of waypoints 'pose'.

Point-to-Point motion	move_j(joints_pose)	This function sets the robot command for performing a point-to-point motion to the joint-angle configuration 'joint_pose'.
Receive current state	get_current_state()	This function returns the current joint pose, flange pose, and TCP pose.
Set base frame	set_base(base_frame)	This function sets the robot's base frame to 'base_frame'.
Set TCP frame	set_tcp(tcp_frame)	This function sets the robot's TCP frame to 'tcp_frame'.
Set acceleration and speed	set_acc_speed(acceleration,speed)	This function sets the robot's Cartesian speed and acceleration.
Control digital input	set_digin(number,value)	This function sets the robot's digital input to the Boolean value of True or False.
Control digital output	set_digout(number,value)	This function sets the robot's digital output to the Boolean value of True or False.
Run a predefined subroutine	call_subroutine()	This function calls a predefined subroutine.

3.6 Intrinsic camera calibration

A 2D camera maps the 3D world appearance to a 2D image. The image contains only 2D information of an object in the form of pixel values. To determine the transformation between the camera and the robot, a 3D coordinate system must first be defined for the camera. As illustrated in Figure 3.5, the pin hole camera model is widely used for mathematically modeling the coordinate system of a camera. In the pin hole camera model, the camera is defined with its own 3D coordinate system that houses the 2D image plane. The size and position of the 2D image plane in the camera's coordinate system depend on the focal length and field of view of the camera

A fundamental problem in the field of machine vision is that 2D images typically contain distortions caused by imperfections in optical lenses. Distortions can lead to error in extracting information from the image. The process of correcting the image distortion and finding the transformation between the camera's image plane and the real-world coordinate system is known as intrinsic calibration, or simply camera calibration.

The pin hole camera model

The pin hole camera model is commonly used for the camera calibration process. As illustrated in Figure 3.5, this model considers the camera as a box that has an infinitely tiny hole. This hole is considered as the camera's lens through which the light passes and makes an image on the other side of the lens.

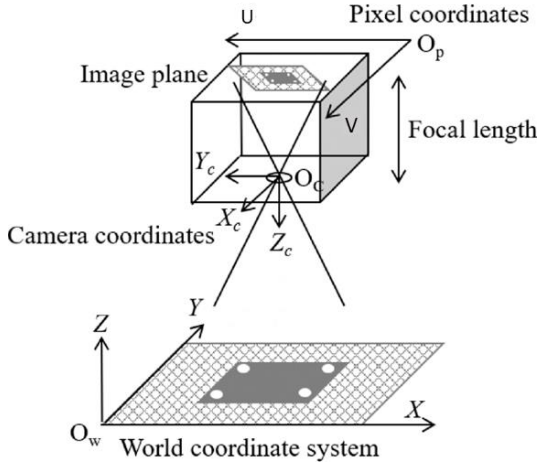


Figure 3.5: Pin hole camera model for frame representation [14]

As presented in Figure 3.5, the pin hole camera model can define a coordinate system for the camera (Xc, Yc, Zc) as well as the world (X, Y, Z). The origin of the camera's coordinate system is at the hole. The distance between the camera's coordinate system and the 2D image plane is the focal length of the camera. The pixel coordinate (U, V) is the 2D coordinate system of the image captured by the camera. The origin of the pixel coordinate system is typically at the top left corner of the image. The 3D data from the world coordinates are projected on the 2D plane of the image. This projection is modelled with the pin hole camera equation as [48]:

$$s.p_c = A\ [\ R|t\ ]\ P_w, \tag{3.5}$$

where s is a scaling factor, p_c is a 2D pixel (U, V) in pixel coordinates, and A is the so-called camera matrix. R and t are the rotational and translational components of the transformation matrix that transform a point expressed in the world coordinates to the camera coordinates, and P_w (X, Y, Z) is the corresponding pixel point in the world coordinates.

Pin hole camera calibration model solves for two main camera parameters known as intrinsic parameters and extrinsic parameters. The internal characteristics of the camera such as focal length, optical center, and the distortion coefficients that remain the same are known as intrinsic parameters. As seen in Figure 3.5, the pin hole camera model defines a coordinate system for the camera (Xc, Yc, Zc). The position of the workpiece seen in the camera image is defined in the world coordinate system (X, Y, Z). The location and orientation of the world's coordinate system with respect to the camera's coordinate system can be defined by a homogeneous transformation matrix containing a rotation matrix R and a translation vector t. This homogeneous transformation matrix is known as extrinsic parameter of the camera.

The camera matrix A is a 3x3 matrix that contains the focal length (f_x, f_y) and the optical center (c_x, c_y) of the camera:

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.6}$$

The pin hole model equation assumes that the camera is distortion free; however, this is not the case in reality. The camera always has some distortion due to imperfections in the lens. These distortions can be modelled mathematically and can be included in the pin hole camera equation. There are mainly two types of distortions in the image, i.e., radial distortion and tangential distortion.

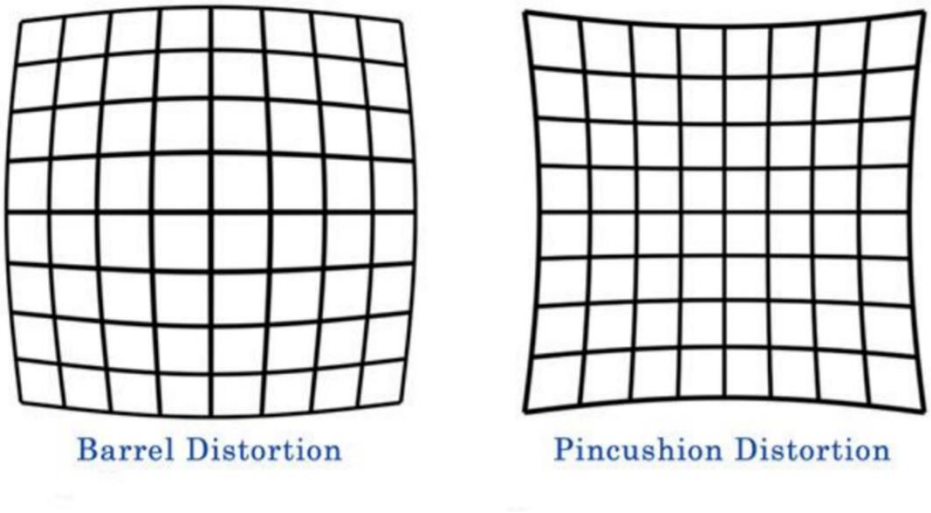


Figure 3.6: Two types of radial distortions [49]

Let us first examine the distortion-free pin hole camera model. The distortion free pin hole camera model is given by:

$$s\begin{bmatrix} U \\ V \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{3.7}$$

Eq. 3.7 is the matrix form of Eq. 3.5. Equation 3.7 is in terms of the world coordinate system. We can simplify Eq. 3.7 by converting it to the camera coordinate system. Since the rotational matrix R and translation vector t define the transformation from the world coordinates to the camera coordinates, any point in the world coordinate system can be expressed in the camera's coordinate system by:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad 3.8$$

By substituting Eq. 3.8 in Eq. 3.7, Eq. 3.7 becomes equivalent to the following:

$$\begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} f_x x_u + c_x \\ f_y y_u + c_y \end{bmatrix}, \quad 3.9$$

with

$$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix}. \quad 3.10$$

Equation 3.9 is the distortion-free mathematical model for projecting a point in the camera's coordinates into a pixel of the camera's pixel coordinates. To incorporate the distortion factor, Eq. 3.9 can be rewritten as follows:

$$\begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} f_x x_{dis} + c_x \\ f_y y_{dis} + c_y \end{bmatrix}, \quad 3.11$$

where

$$\begin{bmatrix} x_{dis} \\ y_{dis} \end{bmatrix} = \begin{bmatrix} x_f + 2p_1 x_f y_f + p_2\left(r^2 + 2{x_f}^2\right) \\ y_f + 2p_2 x_f y_f + p_1\left(r^2 + 2{y_f}^2\right) \end{bmatrix} \quad 3.12$$

and

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} x_u(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_u(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{bmatrix}. \quad 3.13$$

The terms k_1, k_2, k_3 are the coefficients of radial distortion. The terms p_1, p_2 are the coefficients of the tangential distortion according to the Brown Conrady's camera distortion [50] model, and $r^2 = {x_u}^2 + {y_u}^2$.

The OpenCV library [51] provides a procedure for calculating the intrinsic parameters. In this thesis, camera calibration for determining the intrinsic parameters and

distortion parameters is performed for the Triton 2D camera. The Zivid 3D camera is factory calibrated, i.e. the camera matrix and distortion coefficients are provided by the manufacturer.

To model the coordinate system of the Triton camera, its intrinsic parameters are determined using the OpenCV library [52]. OpenCV provides a function known as 'cv.calibrateCamera(object_points, image_points, image)'. This function takes the input of 'object_points' and 'image_points', and it returns the intrinsic parameters of the camera as the output. The object_points are the coordinate points in the real world in terms of (X, Y, Z). The image_points are the corresponding image points of those real world points in terms of pixel coordinates (U, V). There are several calibration patterns that can be used as the target object for defining the object points such as charuco boards, circular pattern calibration boards, and checkerboards. Readers can refer to [53] for further details on the calibration patterns. These calibration patterns have known dimensions in the real world, which means that the 'object points' are already known. As shown in Figure 3.7, to calibrate the camera, multiple images of the target calibration object are first captured from different viewpoints. Then, the image points (image_points) of the target object are detected in the pixel coordinates. Finally, the object points and their corresponding image points are used as the input to the 'cv.calibrateCamera(…)' function to retrieve the intrinsic parameters of the camera.

In this thesis, the checkerboard pattern (Figure 3.7) is used as a target object that defines a world coordinate. The images of the chess board pattern are captured from multiple angles and saved in a folder.

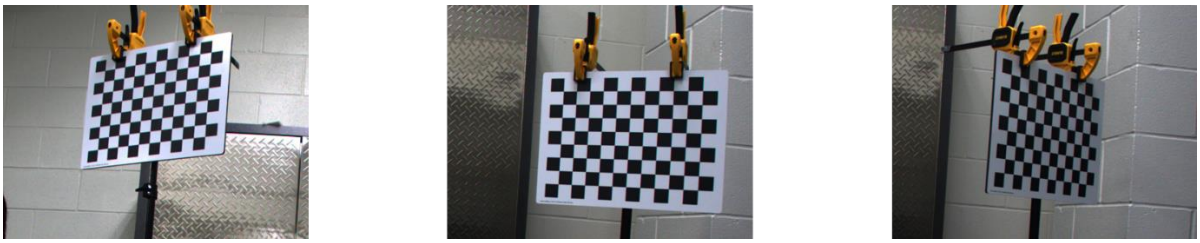


Figure 3.7: Few images of chess board pattern captured by Triton 2D camera

The findChessBoard() function, provided by OpenCV, detects the chess board corner pixels in all the images. Hence both the object points and the image points of the checker

board are known. Using those as the input for calibrateCamera(…), the intrinsic parameters for the Triton camera are determined.

3.7 Hand-eye calibration: finding the robot-camera transformation

The robot cell used in this thesis has two cameras, i.e. Triton 2D camera (RGB) and Zivid Two 3D camera (RGBD). The Triton 2D camera is attached to the robot's flange and the Zivid 3D camera is mounted at a fixed location. The captured images of the 2D camera and the point clouds captured by the 3D camera are represented in the camera's coordinate system and therefore must be transformed to the robot's coordinate system. To transform the image or point cloud data to the robot's coordinate system, the transformation between the camera's coordinate system and the robot's coordinate system must be known. The process of finding this transformation is known as the hand-eye calibration.

This section discusses the procedure for finding two transformations: 1) The transformation between the onboard Triton camera and the robot, and 2) the transformation between the fixed Zivid camera and the robot's coordinate system. As illustrated in Figure 3.8, these two cameras are mounted in different configurations, one is moving and one is fixed. Hence, they need different types of hand-eye calibration. For the moving Triton camera (attached to the robot's flange), 'eye-in-hand calibration' is performed, while for the Zivid camera (fixed in the robot cell), 'eye-to-hand calibration' must be performed. The step of finding the hand-eye transformations is a prerequisite for the robot guidance and inspection tasks studied in this thesis.

For the fixed Zivid camera (Figure 3.8, left), eye-to-hand calibration is performed. The eye-to-hand calibration retrieves the transformation between the camera's coordinate system and robot's base coordinate system, i.e. ${}^{b}T_{c}$. Since the Triton camera is attached to the robot's flange, the eye-in-hand calibration is performed (Figure 3.8, right). The eye-in-hand calibration retrieves the transformation between the camera's coordinates and the robot's flange, i.e. ${}^{f}T_{c}$.

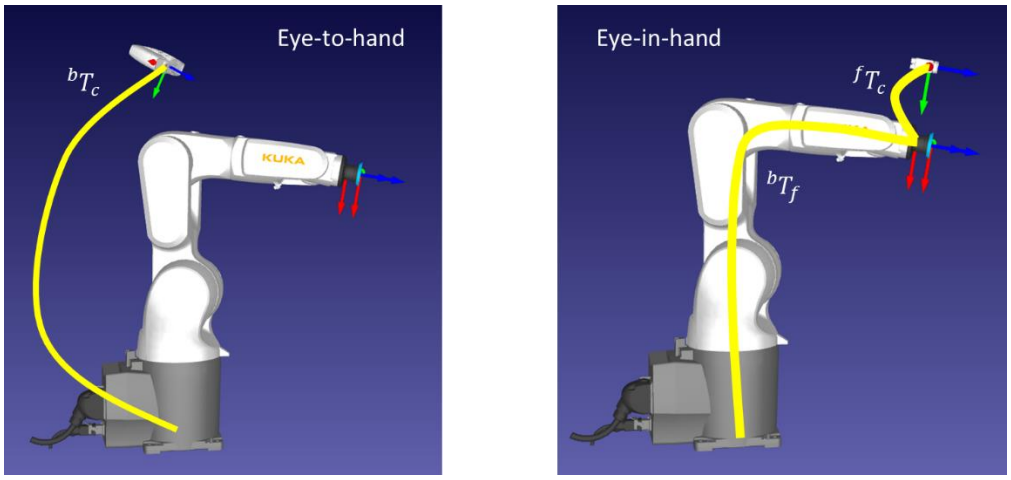


Figure 3.8: Hand-eye calibration for fixed Zivid (left) and onboard Triton (right) cameras

A three-point calibration method is developed in this thesis for retrieving the transformations for both configurations. The proposed three-point method uses a chessboard pattern as a connecting link between the robot and the camera. The three points of the chessboard are used to find the transformation between the robot and the chessboard. Then the transformation between the camera and the chessboard is found using the perspective-n-point problem. These two transformations are used to obtain the desired transformation between the robot and the camera.

The first step for performing hand-eye calibration is to calibrate the tool center point (TCP) of a sharp tool. The four-point TCP calibration function provided in the robot's controller is used for calculating the offset of the TCP from the flange.

Figure 3.9: Robot tool calibration (TCP offset calculation)

After the TCP calibration process, the chessboard is placed in front of the robot. A coordinate system is arbitrarily assigned to the chessboard pattern. Figure 3.10 shows the coordinate convention for chessboard pattern used in this thesis.

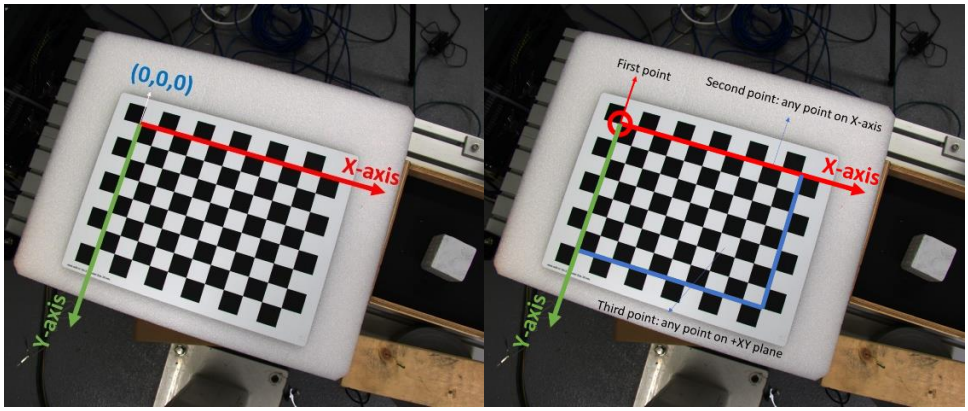


Figure 3.10: Chessboard coordinate system and three defining points

The sharp TCP is manually moved to the three points on the chessboard and the (X, Y, Z) data of these points w.r.t the robot's base are saved manually from the robot's teach pendant. The first point P1: $(P1_x, P1_y, P1_z)$ is the origin of the chessboard, the second point P2: $(P2_x, P2_y, P2_z)$ is any point on the X-axis of the chessboard, and the third point P3: $(P3_x, P3_y, P3_z)$ is any point on the positive XY plane of the chessboard. These three points on the chessboard w.r.t the robot's base are used to find the transformation between the chessboard and the robot's base, i.e. ${}^{b}T_{chess}$.

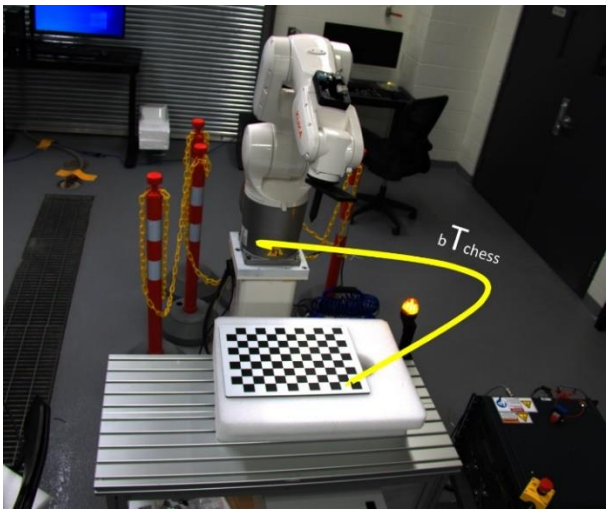


Figure 3.11: Transformation between the chessboard and robot's base

The translation component of transformation ${}^{b}T_{chess}$ is the same as $(P1_x, P1_y, P1_z)$, since that point is the origin of the chessboard. The rotation component of the transformation ${}^{b}T_{chess}$ can be retrieved by finding the vector of three basis axes of the chessboard w.r.t the robot's base. The x-axis (Vector X), as presented in Eq. 3.14, is simply found by subtracting point P1 from P2 and then normalizing the result.

$$\text{Vector X: } (X_x, X_y, X_z) = norm(P2 - P1) \tag{3.14}$$

The three points altogether can define a plane that gives the z-axis $(Vector\ Z)$ of the chessboard w.r.t the robot's base. The formulation for finding $Vector\ Z$ is given by:

$$Vector\ Z\text{: } (Z_x, Z_y, Z_z) = cross(Vector\ X, Vector\ V), \tag{3.15}$$

where $Vector\ V = norm(P3 - P1)$. Following the right-hand rule, the y-axis $(Vector\ Y)$ can be found by the cross product of the z-axis and x-axis, as seen below:

$$Vector\ Y\text{: } (Y_x, Y_y, Y_z) = cross(Vector\ Z, Vector\ X). \tag{3.16}$$

Hence the transformation ${}^{b}T_{chess}$ can be represented as below:

$${}^{b}T_{chess} = \begin{bmatrix} X_x & Y_x & Z_x & P1_x \\ X_y & Y_y & Z_y & P1_y \\ X_z & Y_z & Z_z & P1_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.17}$$

Eye-to-hand calibration for the fixed Zivid camera

For the case of eye-to-hand calibration, where the camera is fixed in the robot's cell, the transformation between the robot's base and the camera $({}^{b}T_{c})$ is calculated as follows.

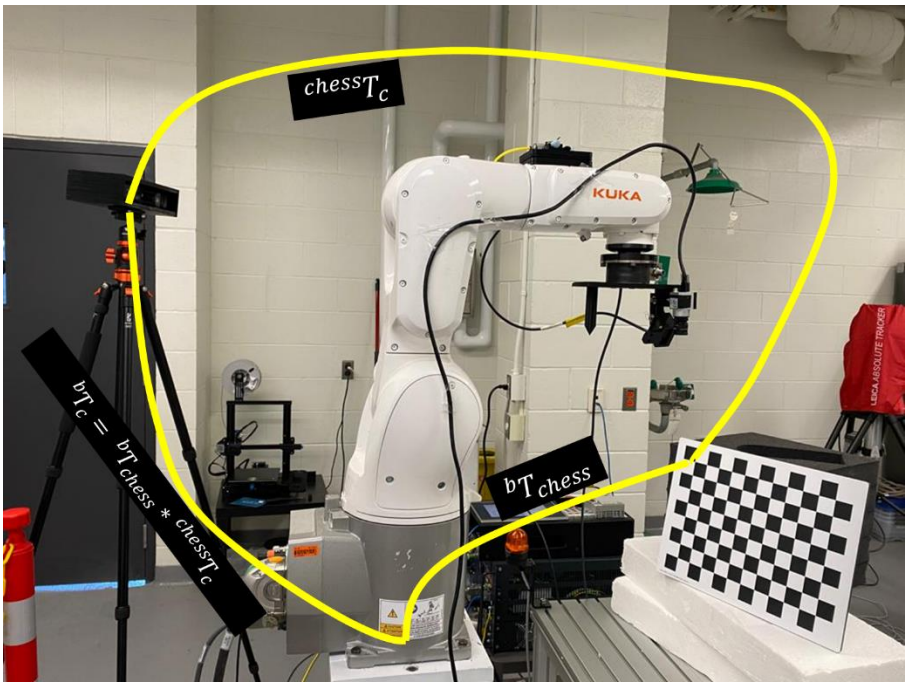


Figure 3.12: Eye-to-hand transformation chain for the fixed Zivid camera

Using the chain of transformations, the transformation ${}^{b}T_{c}$ can be given by:

$$ {}^{b}T_{c} = {}^{b}T_{chess}.{}^{chess}T_{c} \tag{3.18}$$

where, ${}^{chess}T_{c}$ is the transformation between the chessboard and the camera and must be obtained. In this thesis, the transformation between the camera and the chessboard is estimated using the perspective-n-point method. Perspective-n-point is a problem that estimates the pose of the camera in relation to the real-world coordinates given a set of object points in the 3D world space and the corresponding 2D image points (U, V). The OpenCV function 'solvePnP()' solves for perspective-n-point problem. This function takes the input of 3D object points, their corresponding image points, the camera matrix 'A', and the distortion parameters of the camera.

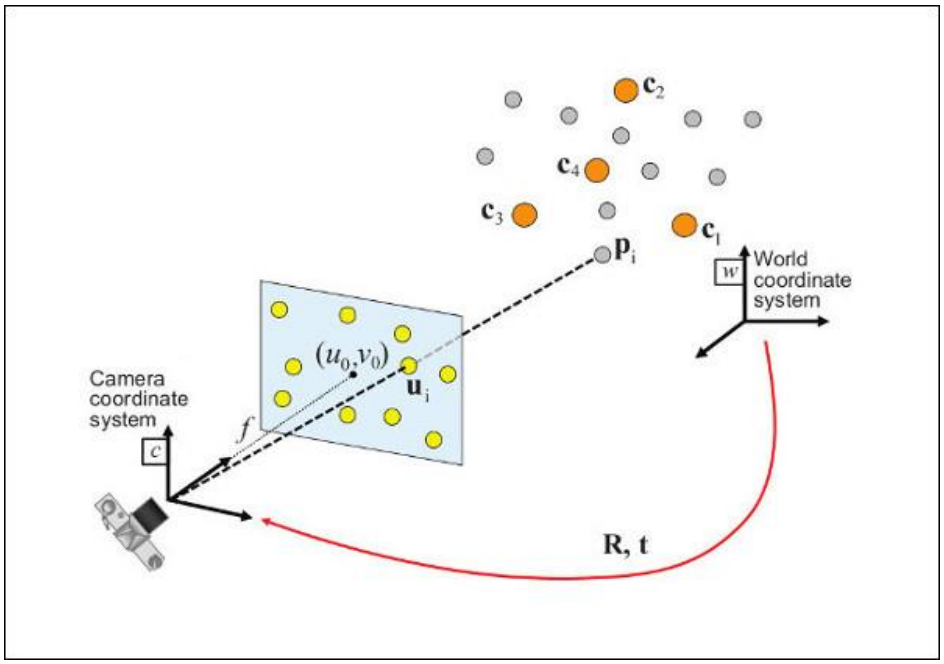


Figure 3.13: Camera pose estimation using perspective-n-point [54]

Since the real-world dimensions of the chessboard are known, the corners of the chessboard are taken as the 3D object points, and the corresponding corner points of the chessboard in the captured image are taken as the image points. For finding the chessboard corners in the image, the OpenCV function 'findChessboard()' is used. Finally, the transformation between the chessboard and the camera, ${}^{c}T_{chess}$, is found using the solvePnP() method. The inverse of ${}^{c}T_{chess}$, i.e. ${}^{chess}T_{c.}$, is then used in Eq. 3.18 to calculate ${}^{b}T_{c}$. For the setup used in this thesis, since eye-to-hand calibration is performed for the fixed Zivid camera, camera-robot transformation (${}^{b}T_{c}$) is referred to as ${}^{b}T_{zivid}$.

Eye-in-hand calibration for the onboard Triton camera

For the case of eye-in-hand calibration, where the camera is attached to the robot's flange, the transformation between the robot's flange and the camera, i.e. ${}^{f}T_{c}$, is calculated as follows.

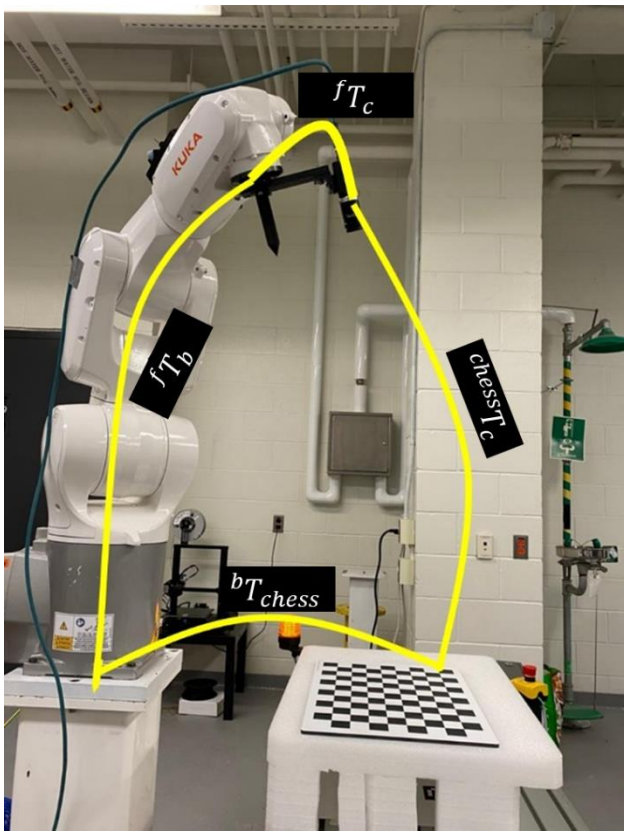


Figure 3.14: Eye-in-hand calibration for the onboard Triton camera

Using the chain of transformations, the transformation ${}^{f}T_{c}$ can be given by:

$$ {}^{f}T_{c} = {}^{f}T_{b}.\,{}^{b}T_{chess}.\,{}^{chess}T_{c} \qquad 3.19 $$

The only new unknown in Eq. 3.19 is ${}^{f}T_{b}$, i.e. the transformation between the robot's base and flange at the pose of image capturing. The pose of the robot while capturing the image of the chessboard is retrieved from the robot's teach pendant and converted to the homogeneous matrix format using Eq. 3.4. This homogeneous matrix provides the transformation ${}^{b}T_{f}$. The inverse of ${}^{b}T_{f}$, i.e. ${}^{f}T_{b}$, is used in Eq. 3.19 to obtain the camera-robot transformation, ${}^{f}T_{c}$. For the setup used in this thesis, eye-in-hand calibration is performed for the onboard Triton camera. Hence, the transformation between the Triton camera and the robot's flange is denoted as ${}^{f}T_{triton}$.

3.8 Automated part localization using vision system

In order to eliminate the need for part-specific fixturing in a robot cell, the robot must be able to automatically find the location and orientation of the part. The part's coordinate system is generally defined with respect to the robot's base frame by touch probing the workpiece with the robot's end effector. This manual process is time consuming and leads to robot downtime. The aim of the part localization aspect of this thesis is to develop a vision-based solution to automate the part localization process. As explained in the following section, the iterative closest point (ICP) registration technique is employed to autonomously localize the free-form composite panel in the robot's base coordinate system.

Iterative closest point registration

The ICP algorithm is a two-step iterative algorithm that finds the mapping (i.e. 3D transformation) between two sets of point clouds. In the first step, the correspondence between the two point clouds are found. In the field of point cloud registration, there are many different techniques to determine the correspondence set K,

$$Correspondence\ set\ K = \{(p, q)\}, \tag{3.20}$$

where p is the correspondence points of the second point cloud where everything is transformed to, and q is the correspondence points of the first point cloud from which everything is transformed. Some examples of point correspondence methods are closest point technique [16], closest compatible point technique [55], [56], normal shooting technique [57], projection-based approaches [58], [59], etc. In the second step, the objective function $E({}^{b}T_{w})$, presented in Eq. 3.21, is minimized by updating the value of ${}^{b}T_{w}$, where, ${}^{b}T_{w}$ is the transformation between the robot's base 'b' and the new location of the workpiece 'w',

$$E({}^{b}T_{w}) = \sum_{(p,q)\epsilon k} \| p - {}^{b}T_{w} q \|^{2}. \tag{3.21}$$

The value of ${}^{b}T_{w}$ is found using Singular Value Decomposition (SVD) for minimizing the objective function $E({}^{b}T_{w})$. These two steps are iterated until both the point clouds are aligned tightly.

Implementation of ICP using open3D

In this research, the point cloud of the free-form composite panel at the nominal location is captured using the 3D Zivid camera. Using the open3D library [60], this point cloud is first cropped to remove the background. The cropped point cloud is represented in the camera's coordinate system. For simplicity, the nominal workpiece's coordinate system is kept at the same location and orientation as the robot's base frame. Hence, the point cloud needs to be first transformed to the robot's base coordinate system. The transformation between the robot's coordinate system and the camera's coordinate system is already derived using the hand-eye calibration process explained in Section 3.7. The cropped point cloud is first transformed to the base frame of the robot using the eye-to-hand transformation, ${}^{b}T_{zivid}$. In the thesis, the transformed point cloud is referred to as the source point cloud (Q_{PC}), which represents the nominal location of the panel.

Suppose a new identical composite panel is placed on the fixture manually for the new robot cycle. Since the new panel is placed manually, it cannot be at the exact location as the nominal part. The point cloud of the new composite panel is scanned again using the fixed Zivid camera. Then it is transformed to the robot's base coordinates using ${}^{b}T_{zivid}$. The new transformed point cloud is referred to as the target point cloud (P_{PC}), which represents the new location of the panel.

The open3D function 'TransformationEstimationPointToPoint()' is used to perform the ICP registration between the source point cloud (Q_{PC}) and the target point cloud (P_{PC}), as illustrated in Figure 3.15. This method uses the closest point technique for identifying the correspondence set K. In the closest point technique, for every point of the nominal point cloud, the closest points on the new point cloud are found. All of these pairs of closest points are considered as the correspondence set K. The open3D ICP registration function uses the KD-tree search algorithm [61] to find the pairs of closest points. Once the correspondence set K is identified, the C++ eigen library's 'umeyama' function is used to minimize the objective function E(${}^{b}T_{w}$) by finding the best values of ${}^{b}T_{w}$. Readers can refer to [62] for further details on the minimization of the objective function. This entire two-step process is

iterated until the solution converges, hence the best value for the transformation ${}^{b}T_{w}$ is obtained.

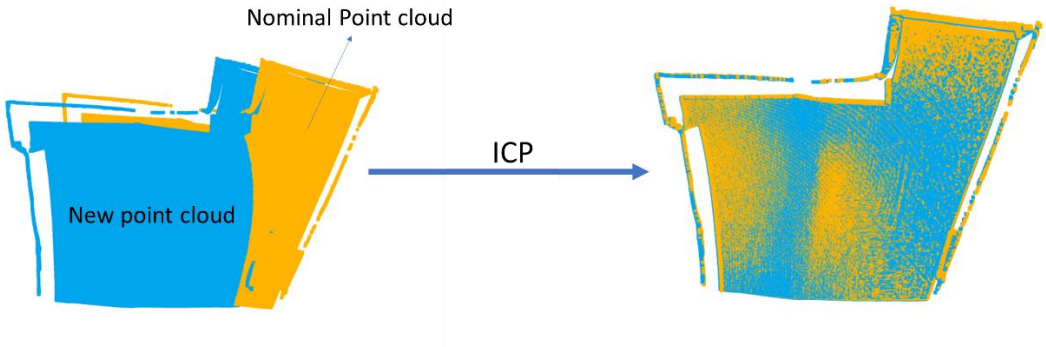


Figure 3.15: Illustration of iterative closest point (ICP) for point cloud registration

Since the coordinate system of the source point cloud is the same as the robot's base frame, the obtained ICP transformation readily provides the transformation between the robot's base frame and the workpiece's coordinate system. This process allows for the automation of the part localization by leveraging 3D point clouds captured by the Zivid camera.

3.9 Refining part localization using nominal corner pose

Due to camera measurement errors and point registration inaccuracies, the part localization framework introduced in the previous section may not be able to provide sufficient accuracy. This section proposes a novel, secondary layer for improving the translation accuracy of the proposed part localization method.

Since the transformation between the robot's flange and the onboard Triton camera (${}^{f}T_{triton}$) is already known, the pose format of ${}^{f}T_{triton}$ is used to set the robot's TCP in the controller to the Triton camera. After capturing and transforming the nominal point cloud (Q_{PC}) when the composite panel is still at its nominal position, the image of the curved panel's corner is captured using the onboard Triton camera (Figure 3.16). The robot's pose at which this image is captured is referred to as the nominal corner pose (${}^{b}CP_{nominal}$). Since

the coordinates of the workpiece and the robot's base are the same at the nominal workpiece location, the pose transformation ${}^{b}CP_{nominal}$ can equivalently be represented as ${}^{w}CP_{nominal}$. The image captured by the Triton camera at that pose is presented in Figure 3.16 (right). As explained below, a series of image processing steps have been developed to automatically detect the panel's corner at the nominal location. The nominal pixel coordinates (U, V) of the detected corner are then saved and used in the next steps.

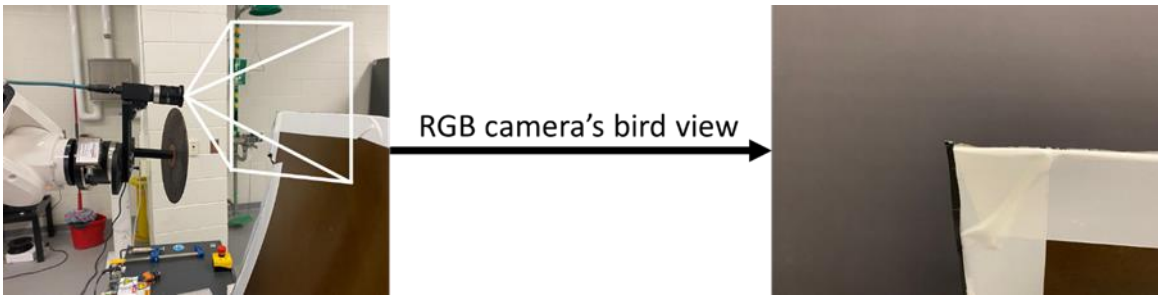


Figure 3.16: Nominal corner image at nominal corner pose

Image processing steps: Image processing is generally performed to improve the quality of captured images and segmenting important information from an image. In image processing, the image is considered as a matrix of dimension $U_{pixels} \times V_{pixels}$, where U_{pixels} and V_{pixels} are the width and height of the image in pixels, respectively. Each element of the image matrix contains the pixel data at that particular location of the image coordinates.

There are many different image processing techniques that can be performed on an image, for instance, image thresholding [63], image blurring [64], image subtraction [65], image segmentation [66], canny edge detection [67], image morphological transformation [68], image geometric transformation [69], etc. Each technique has its own applications.

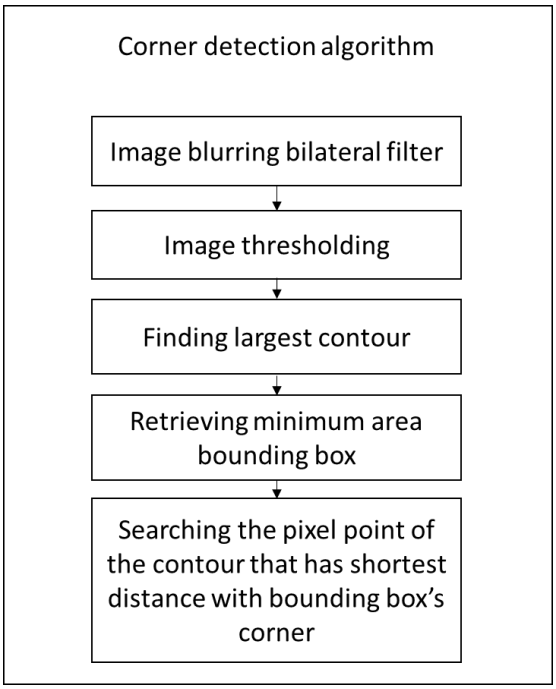


Figure 3.17: Image processing steps for automated corner detection

Figure 3.17 shows the flowchart of the performed image processing steps for automated 'corner detection'. As shown in Figure 3.18, the image is first blurred using bilateral filter. The bilateral filter removes unwanted noise from the image while maintaining the sharpness of edges.

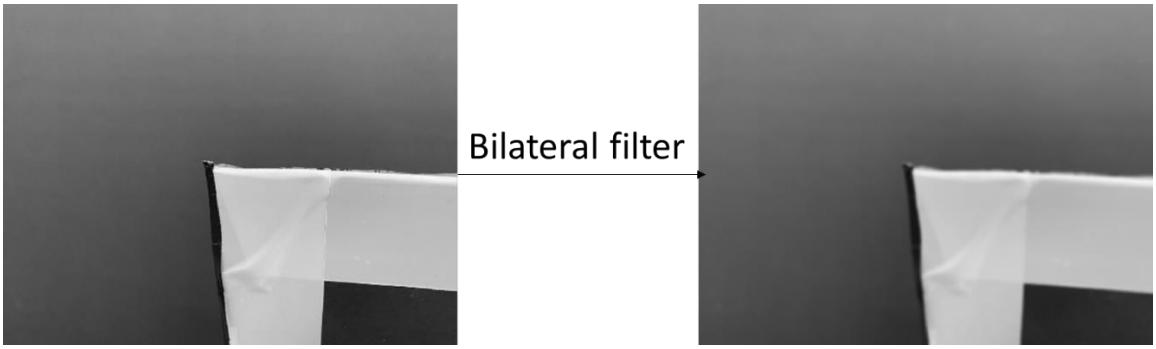


Figure 3.18: Implementation of bilateral filter for noise removal

After removing the noise using the bilateral filter, large objects in the image are segmented using the image thresholding technique. In image thresholding, each pixel of the image is converted into either 1 or 0. Hence, the resultant image becomes a binary image (Figure 3.19). The value of the pixel in the binary image is set to 1 if the image pixel is within

the defined threshold boundaries. Otherwise, it is set to 0. The threshold values are found empirically based on the appearance of the part.

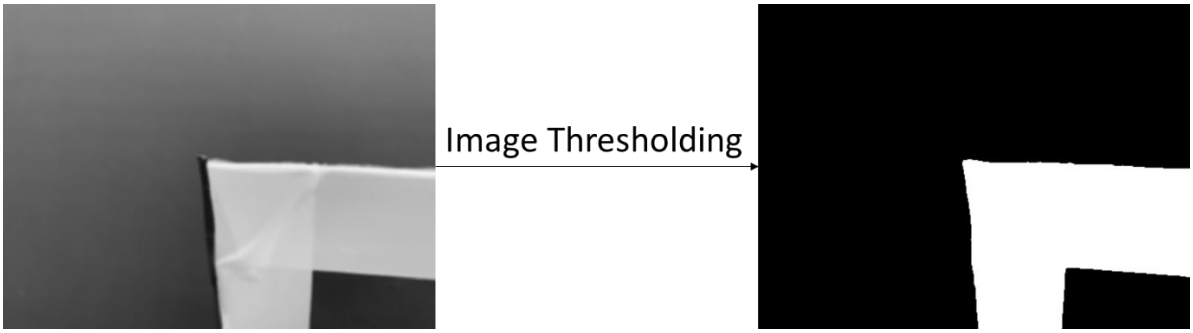


Figure 3.19: Image thresholding for corner detection

Using the OpenCV cv2.findContours() function, the contours in the binary image are found. The largest contour found in the image is of the curved panel. As shown in Figure 3.20, a minimum area bounding box is then obtained for the largest contour using the cv2.minAreaRect() function in OpenCV.

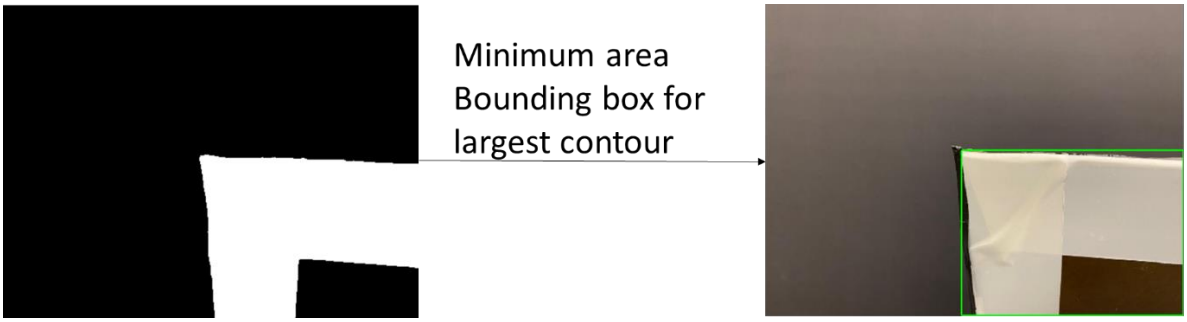


Figure 3.20: Minimum area bounding box for largest contour

As illustrated in Figure 3.21, the distance between each contour point and the top left corner of the bounding box is computed. The contour point that has the shortest distance is marked as the pixel coordinates of the panel's corner in the image. This pixel point is saved in the database. Further in the thesis, this pixel point is referred to as the 'reference corner pixel point'.

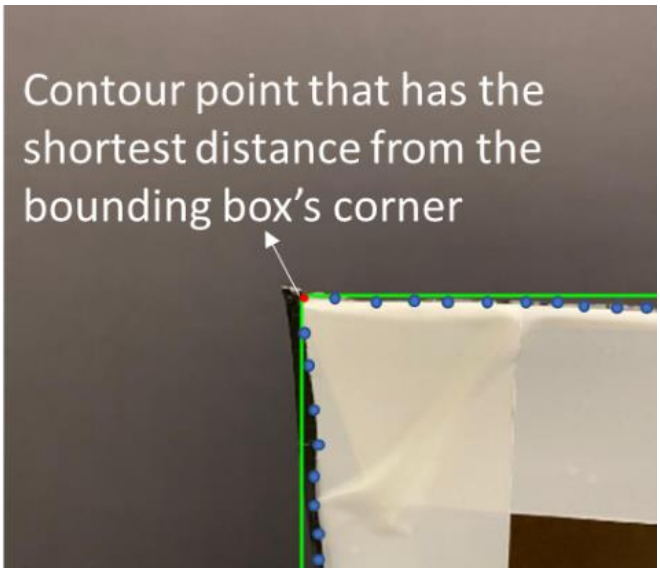


Figure 3.21: Detected corner point using automated image processing

Correcting the corner pose: The nominal corner pose when the workpiece is placed at the nominal configuration is denoted as ${}^{w}T_{nominalCP}$. Now, assume that the part has been placed arbitrarily in front of the robot. Let us denote the new corner pose as ${}^{b}CP_{new}$. The new corner pose represented in the matrix form can be calculated as:

$$ {}^{b}T_{newCP} = {}^{b}T_{w}.{}^{w}T_{nominalCP}. \qquad 3.22 $$

where ${}^{b}T_{w}$ is the part localization transformation obtained as described in Section 3.8. The matrix form of new corner pose (${}^{b}T_{newCP}$) is converted to the pose format, i.e. ${}^{b}CP_{new}$.

In order to implement the proposed method for refining part localization, the robot is first moved to this new corner pose (${}^{b}CP_{new}$). The image of the free-form composite panel's corner is captured at this new corner pose. The corner detection image processing algorithm is used to detect the corner of the composite panel in the image. Figure 3.22 shows the image of the new corner point in the image captured from the new corner pose. The red dot in the image represents the pixel point for the nominal corner point and the blue dot is the pixel point of the new detected corner. As can be seen, due to inaccuracies in the ICP transformation (${}^{b}T_{w}$), the detected corner point may not exactly match the projected nominal corner point.

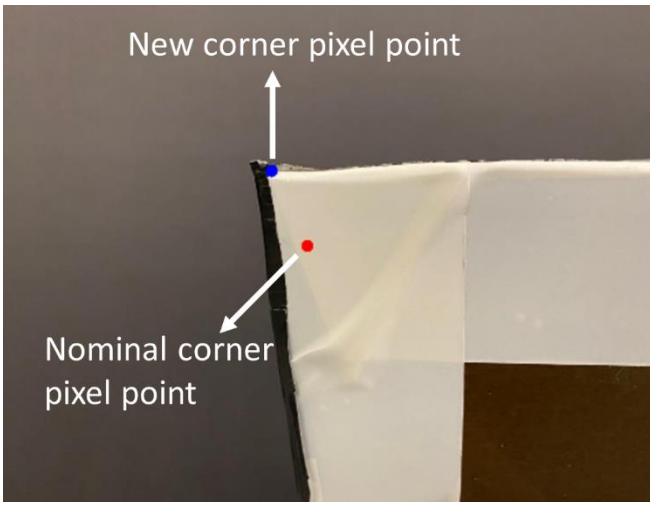


Figure 3.22: Detected corner point at the new corner pose

To correct this error, the robot needs to move in such a way that the new corner point in the image matches the nominal corner point. Based on the pin hole camera theory (Section 3.6), the U and V direction of the image represent the X and Y coordinates of the camera, respectively. Hence, the robot must be moved in X and Y direction of the camera's coordinates to correct this error. The TCP of the robot is set to the Triton camera. The robot is moved in small amounts in X and Y direction in the TCP's coordinate system until it reaches the position where the nominal and new corner points coincide within a defined error threshold. The pose of the robot at this corrected corner pose is referred to as the refined corner pose (${}^{b}CP_{refined}$) with matrix form ${}^{b}T_{refinedCP}$. The total motion in the X and Y axes in the tool's coordinates is saved and referred to as delta X and delta Y.

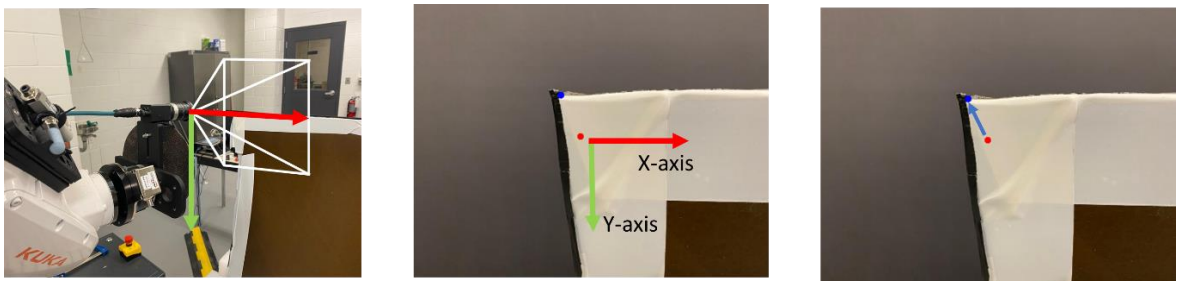


Figure 3.23: Robot motion for correcting the corner pose

The transformation between the new corner pose and the refined corner pose ($^{newCP}T_{refinedCP}$) in the tool's coordinate system can be given by:

$$^{newCP}T_{refinedCP} = \begin{bmatrix} 1 & 0 & 0 & delta\,X \\ 0 & 1 & 0 & delta\,Y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad 3.23$$

Now the refined part localization transformation $^{b}T_{w_{refined}}$ can be found using matrix manipulation:

$$^{b}T_{w_{refined}} = {}^{b}T_{w}.\,{}^{w}R_{newCP}.\,{}^{newCP}T_{refinedCP}.\,{}^{newCP}R_{w}, \quad 3.24$$

where $^{w}R_{newCP}$ is the transformation between the workpiece frame and the new corner pose frame ($^{w}T_{newCP}$) with translation part kept as zero $(x, y, z = 0)$. $^{w}T_{newCP}$ is given by:

$$^{w}T_{newCP} = {}^{w}T_{b}.\,{}^{b}T_{newCP}, \quad 3.25$$

where $^{w}T_{b}$ is the inverse of the ICP transformation matrix $^{b}T_{w}$.

Figure 3.24 summarizes the sequence of steps in the proposed automated part localization and refinement framework.

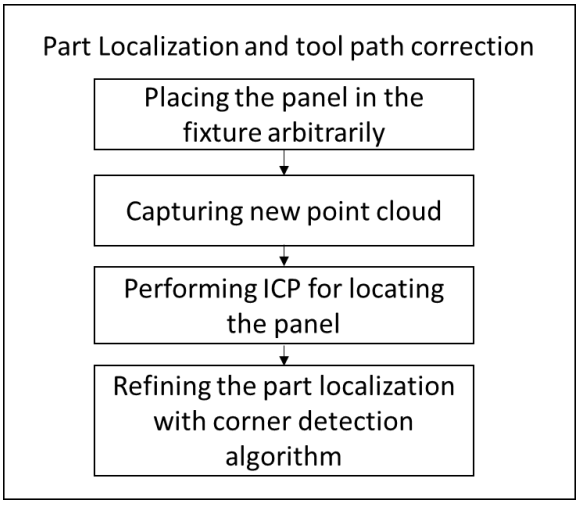


Figure 3.24: Flowchart of the developed algorithms for automated part localization and refinement

3.10 Summary

In this chapter, the problem of manual calibration of the workpiece location with respect to the robot's base coordinates is addressed. The coordinate systems used in this research are first established. The process of calibrating the intrinsic and extrinsic parameters of cameras is outlines. Hand-eye calibration is implemented to obtain the homogeneous transformation between the cameras and the robot. This homogeneous transformation is used for transforming the point cloud from the camera's coordinate system to the robot's base coordinate system.

A novel framework for automating part localization is proposed using the ICP point cloud registration technique. The ICP technique takes in the sets of two point clouds and gives out the transformation matrix that can transform one point-cloud on top of the other. A corner detection and error compensation algorithm is developed to further improve the accuracy of part localization.

4 Automated water-break inspection and tool path generation

4.1 Introduction

Once the composite panel is localized, the surface is sprayed with water for ASTM-F22 water-break inspection. The aim of this chapter is to automatically detect water-broken regions, i.e. contaminated areas. An image processing-based approach is proposed to detect water-broken regions. To provide consistent results, the effect of different lighting conditions and filtering is studied.

After contamination detection, contaminated regions must be abraded. Abrasion removes contamination and also creates micro scratches on the surface to facilitate the adhesion of coating to the surface (hydrophilic). This thesis proposes to divide the panel's surface into multiple grids and automatically generates abrasion tool paths for all grids.

This chapter is organized as follows: Section 4.2 discusses the vision-based water-break inspection and lighting selection. Section 4.3 presents the image processing algorithm developed in this thesis for autonomous water-broken region detection. Section 4.4 discusses the region of interest (ROI) selection and grids reprojection technique. Section 4.5 presents the method for converting discontinuous point clouds to continuous B-spline surfaces. Section 4.6 presents the theory for finding the surface normal on each point of the B-spline surface. Section 4.7 presents the method for finding Euler angle for robot poses from the surface normal of the B-spline surface. Finally, the methodology for abrasion tool path generation is discussed in Section 4.8.

4.2 Vision based water-break inspection and lighting selection

Once the part is fixed on a vertical fixture, water is sprayed on the composite panel. The behavior of the water flowing downwards on the surface is captured using the Zivid camera for 25 seconds. The captured video of the water behavior is then examined by the developed image processing algorithm for detecting the water-broken regions that represent

faulty (contaminated) areas on the surface. Figure 4.1 shows the water-broken region on a test workpiece.

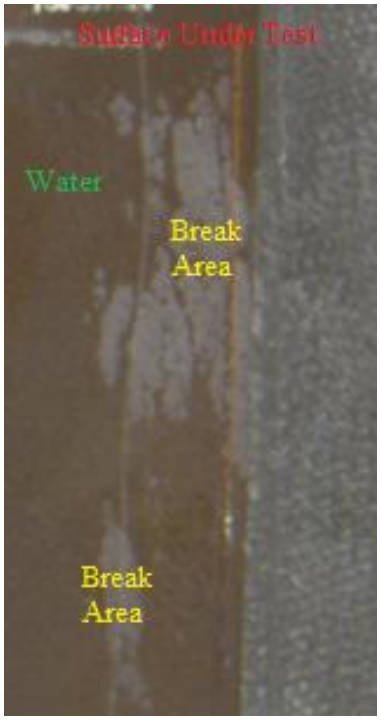


Figure 4.1: Water-break regions on a test surface

The lighting condition in any machine vision application is a critical factor that affects the quality and robustness of the vision-based inspection process. Therefore, appropriate lighting condition must be employed for generating robust results in detecting water-broken regions. Martin [70] presents a practical framework for identifying the ideal lighting conditions for typical machine vision applications. He suggests accumulating and analyzing data in three main areas. First, the knowledge of lighting types, cameras, sensor quantum efficiency, spectral range, and illumination techniques. Second, familiarity with vision geometry, light pattern, light wavelength, and wavelength filters. Third, the detailed analysis of inspection environment and light interactions with respect to unique samples. This research leverages this information to maximize the contrast on water-broken regions and minimize the contrast elsewhere. As described in the following, several tests are performed by subjecting the composite panel to different types of lights and illumination techniques for selecting the best possible lighting condition.

Testing Infrared (IR) lights:

The patent developed by Amos [24] detects water-broken regions on a metal work piece by using a long-range (7-14 μm) IR light camera. It suggests that due to the low emissivity of metal surfaces, a long IR range is more appropriate. However, long-range IR lights and cameras are very expensive and thus impractical for production systems. Besides, the workpiece used in this research is made of carbon composite instead of metal. Nevertheless, the idea of using cheaper near-infrared lights for detecting water-broken regions is worth investigating as water has a high absorption coefficient.

Figure 4.2 shows the absorption coefficient of water for different light wavelength. Higher light absorption can reduce the reflected light from wet surfaces, thus making them appear darker compared to dry areas. Using IR lights can also remove the effect of inconsistent ambient light.

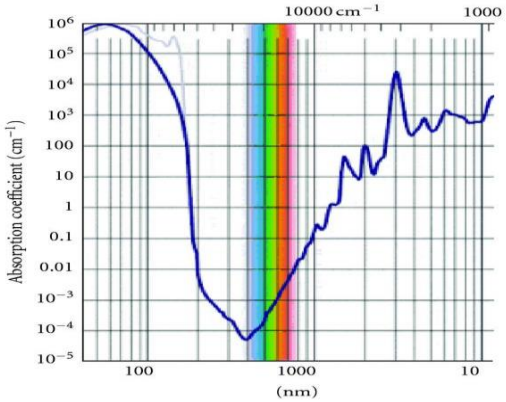


Figure 4.2: Absorption coefficient vs light wavelength plot for water [71]

As shown in Figure 4.3, water-break inspection on the curved panel is tested by projecting diffused IR lights on the workpiece. The camera is equipped with a high pass IR wavelength filter for rejecting the visible lights and accepting only the IR reflected lights.

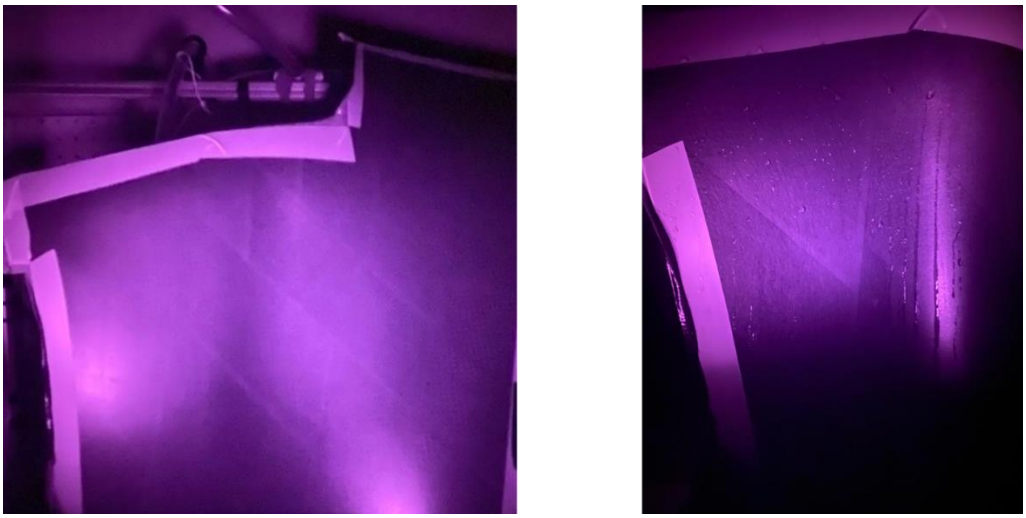


Figure 4.3: Projecting IR lights on the curved panel.

By nature, IR lights are more focused than normal visible lights. As can be seen in Figure 4.3, illumination is not uniform over the workpiece even after diffusing the light source. This nonuniform lighting condition can introduce specular reflections. Moreover, the camera cannot capture a good quality image in this lighting condition because of low quantum efficiency of the complementary metal-oxide semiconductor (CMOS) sensor at higher wavelength (Figure 4.4).

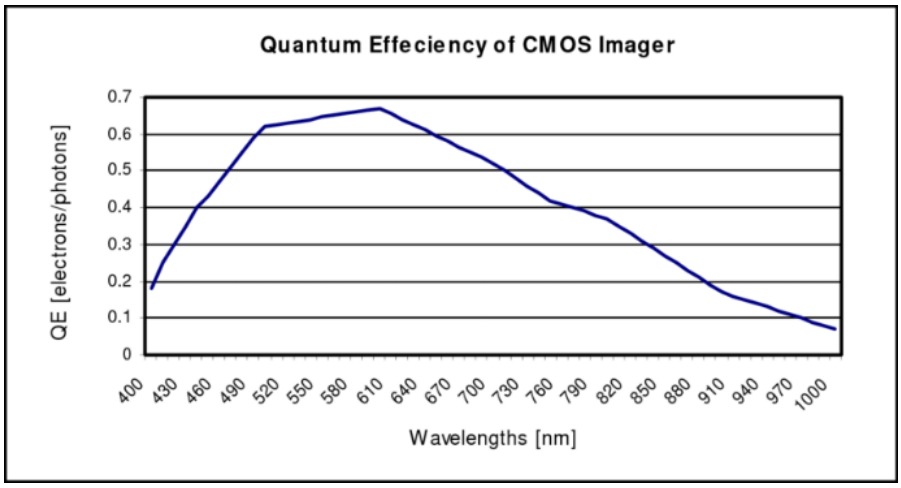


Figure 4.4: Quantum efficiency of CMOS sensor [72]**.**

The quantum efficiency of an imager is a measure of the efficiency of the imager to convert incident photons into electrons, subjected to different light wavelengths. A low

quantum efficiency can introduce noise to the image. Therefore, the idea of using IR lights with CMOS sensor was not pursued further in this thesis.

Testing Light-Emitting Diode (LED) and Compact Fluorescent Lamps (CFL):

In order to investigate the part appearance subject to LED lights, the workpiece is projected with low-angled multilight LED system as presented in Figure 4.5. The LED lights are projected from different directions in order to analyze the effect of light direction and specular reflections on the composite panel.

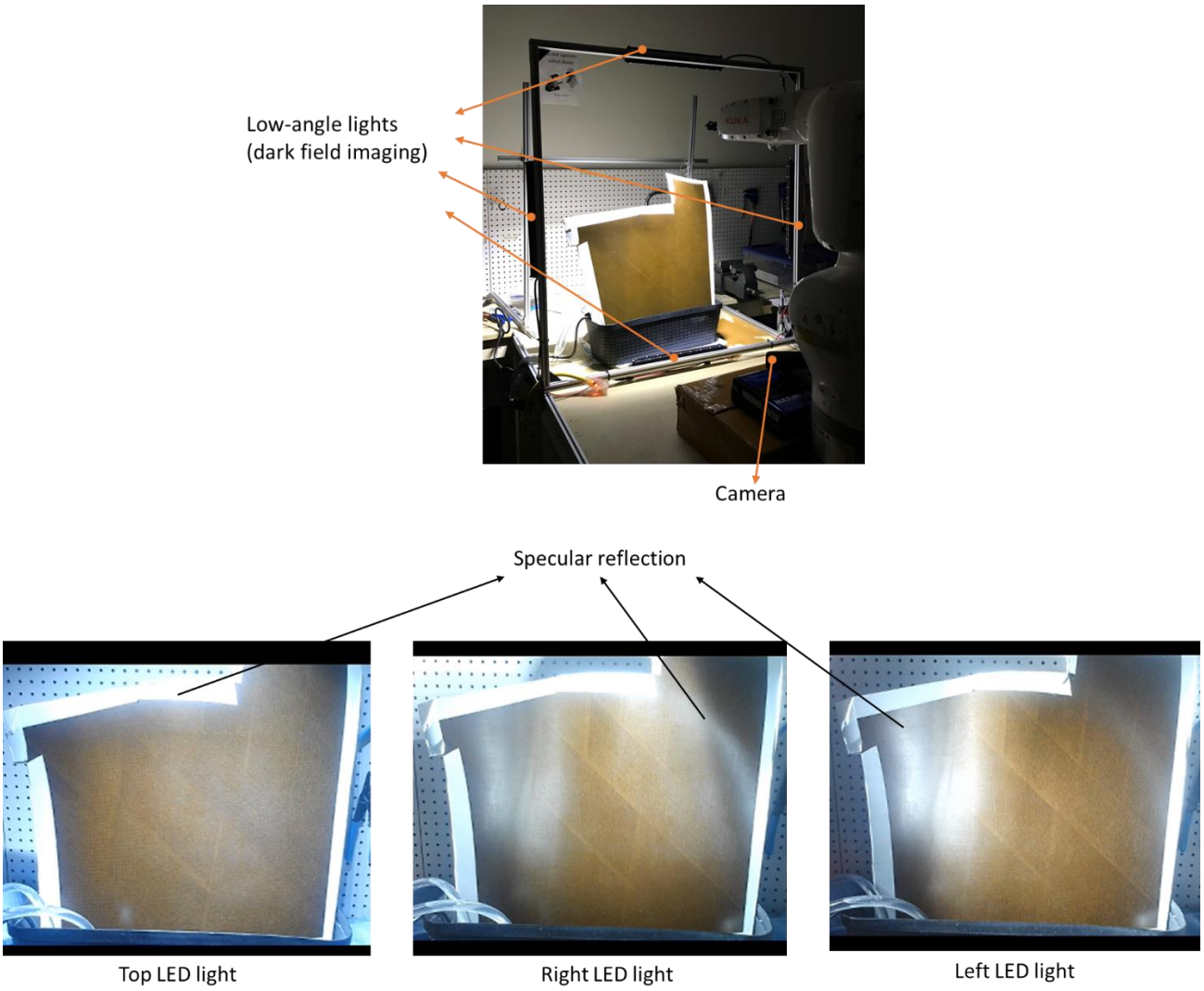


Figure 4.5: Panel subject to multiple LED light system.

The low-angled multiple LED light system tends to produce specular reflection on regions that are close to the light source. This is because LED lights have large intensity and therefore are not ideal for large-area inspection. Hence, the multi-light LED system was not pursued further.

Figure 4.6 suggests that the best type of light for large area inspection is fluorescent. As shown in Figure 4.7, a Compact Fluorescent Lamp (CFL) light bulb with a diffuser is projected on the workpiece from the top.

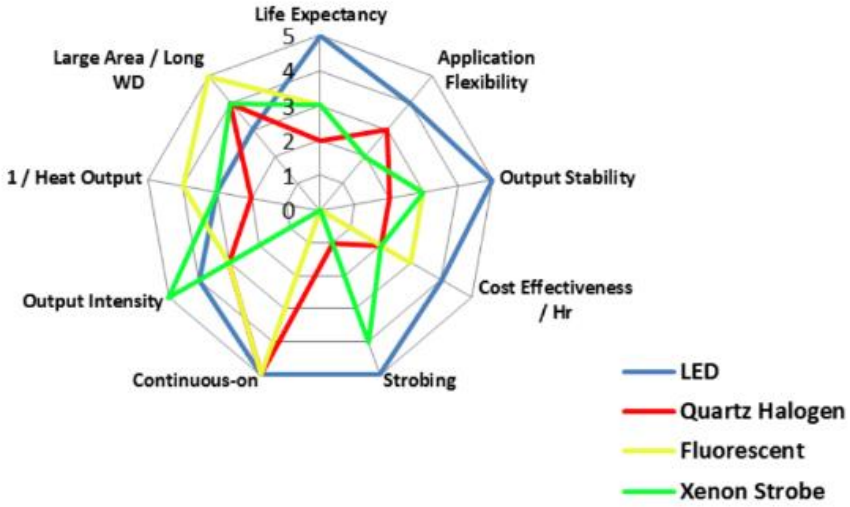


Figure 4.6: Comparison of different types of light used in machine vision applications [70]**.**

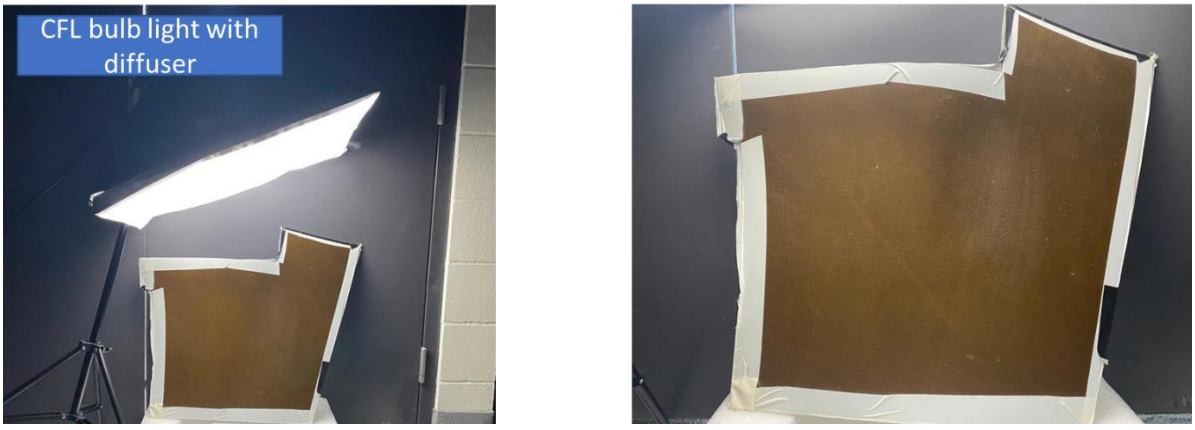


Figure 4.7: Workpiece appearance subject to CFL light.

As can be seen in Figure 4.7, fluorescent lights with the diffuser produce the least amount of specular reflection. Moreover, due to the higher quantum efficiency of the camera at visible wavelength lights, the quality of the image is not compromised. With minimum specular reflections and the best image quality, this lighting environment is finally selected as the ideal lighting condition for this research. After selecting the suitable lighting condition, different types of image processing algorithms are tested for detecting the water-broken region.

Testing different color channels:

The next lighting test is done by investigating the appearance of each color channel of the camera individually. The red, green, and blue channels of the image are split into three different grayscale images. These images show the effect of three different light wavelengths on the appearance of the wetted workpiece. Figure 4.8 shows the grayscale images of each color channel.

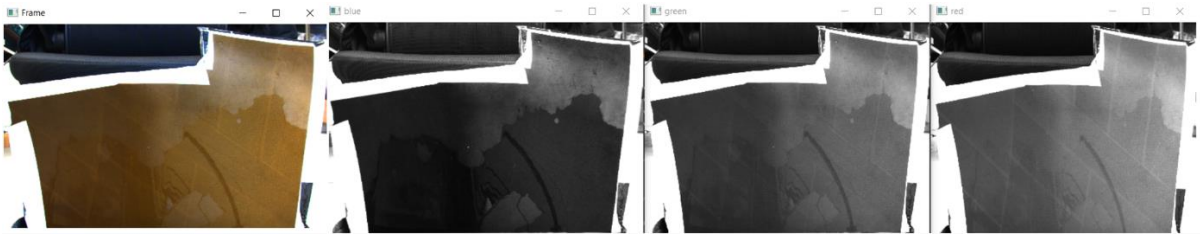


Figure 4.8: Visualizing each color channel separately

As can be seen in Figure 4.8, the red channel does not create a good contrast between the wet and dry portions of the composite panel. This is because the color of the part is close to red, which makes the red channel the brightest. It is noticed that wetting the curved panel makes the part appear even redder because of greater light absorption of water. This makes the wet portion of the blue channel image darker. While the image from the blue channel alone can potentially be sufficient for inspection, it was determined that using the original color image, i.e. leveraging all three channels, provides the highest level of robustness. Therefore, the image processing algorithms presented in the following section take advantage of full color images during the inspection process.

4.3 Image processing for automated water-break inspection

The proposed image processing algorithm is discussed in this section. During the water-break test as water flows over the panel surface, the camera captures a video for 25 s. The image-processing algorithm is run on each frame of the captured video in a loop. All frames of the video remain nearly the same until the water layer starts breaking. This makes the water-break inspection problem equivalent to detecting and marking changes between subsequent images in the video. Ergo, the proposed framework is built on the idea that the image that contains the dry area (water-broken region) of the surface and the image of the same workpiece that does not contain any dry area have changes just at the location of the dry areas. To detect changes between two images, several image processing techniques can be used [73], e.g. image ratioing, image subtraction, change vector analysis, principal component analysis, etc. Image subtraction is the most widely technique for detecting changes due to its simplicity and computational efficiency. This research uses the image subtraction technique to detect the changes between two images.

Image subtraction routine: In image subtraction, a reference image is subtracted from the test image pixel by pixel. If the images contain more than one channel, e.g. Red-Green-Blue (RGB) channels in our case, each channel is subtracted from its corresponding channel. The OpenCV function 'absdiff (reference_image, test_image)' is used to subtract the test frames of the video from the reference frame. This function takes in the reference image and test image as inputs, and gives out the absolute difference between the images:

$$subtracted\ image, sub(U,V) = saturate(|reference(U,V) - test(U,V)|) \qquad 4.1$$

Eq. 4.1 presents the formulation for 'absdiff(…)' function, where U and V indicate the pixel location. The absdiff function uses its internal saturate function that, if needed, converts both images to the same data type. It then returns the final absolute subtracted image.

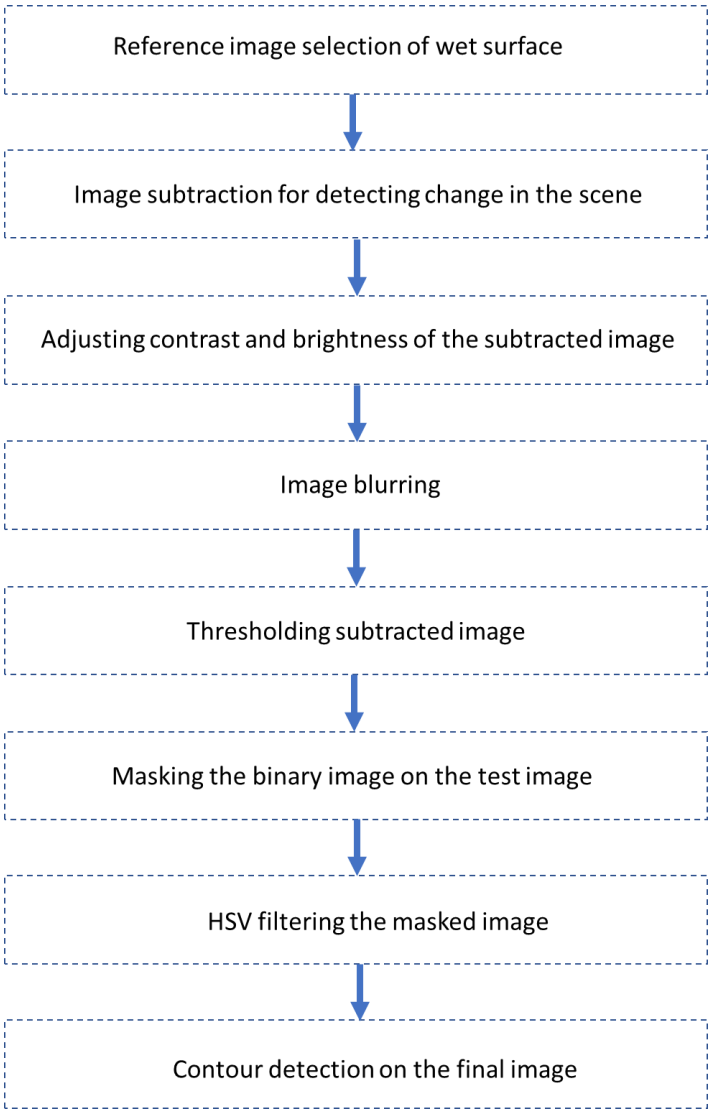


Figure 4.9: Proposed image processing algorithm

Figure 4.9 shows the flowchart of the proposed image processing algorithm. The first step in the flow chart is to select the reference image. For this research, the first frame of the video after the surface is sprayed with water is selected as the reference image. This is because the first image contains the image of the curved panel when it is entirely wet. The image with the curved panel entirely wet is the best candidate for the reference image because the dry area (water-broken region) in the test image can directly appear while subtracting the images.

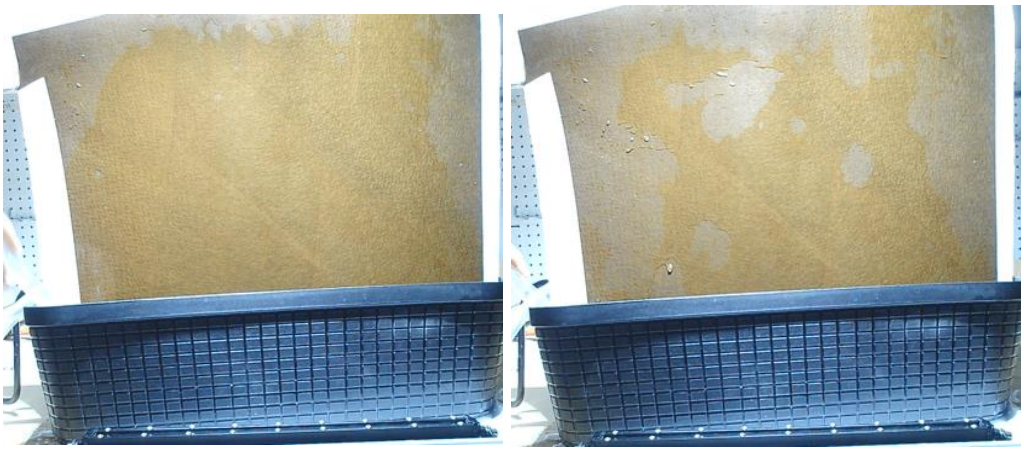


Figure 4.10: Reference image with wet composite panel (left) and Test image with water-broken region (right)

Figure 4.10 shows the reference and test images. Now, the test image is subtracted from the reference image. Figure 4.11 shows the resultant subtracted image.

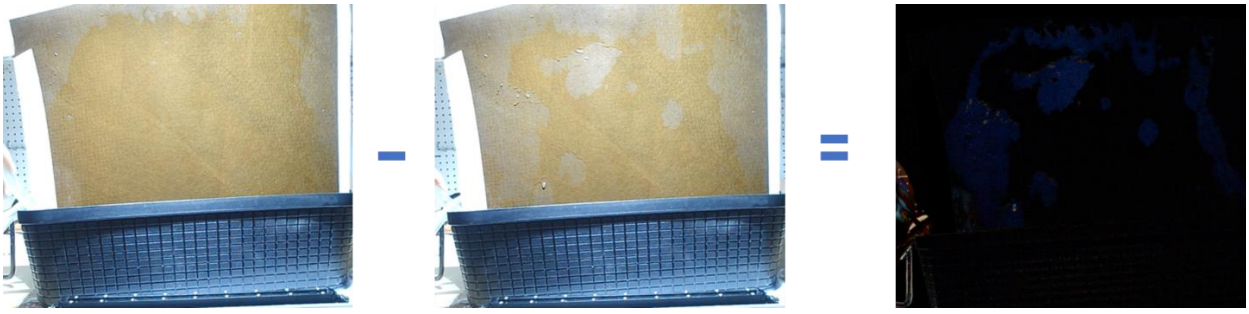


Figure 4.11: Result of image subtraction

The issue with the subtracted image is that it is dark, and the detected dry regions are not visible. To address this problem, the brightness and the contrast of the subtracted image are adjusted. The contrast and brightness can be changed using the following equation:

$$g(U,V) = \alpha \cdot sub(U,V) + \beta, \qquad 4.2$$

where α denotes the gain that controls the contrast, and β is the bias that controls the brightness of the image. The values for α and β are found emphirically.

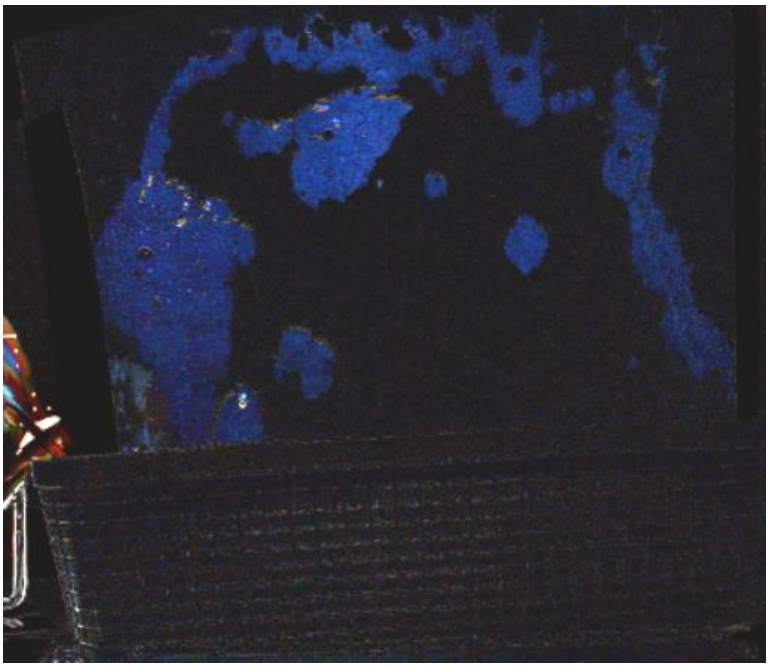


Figure 4.12: Subtracted image with adjusted brightness and contrast

As presented in Figure 4.12, the adjusted image can properly segment the dry area on the curved panel. However, it can also be noticed that the image has a lot of salt and pepper noise. To remove the salt and pepper noise, the median blur algorithm is used on the image.

Similar to the filtration of one-dimensional signals using high pass filters for removing noise, a two-dimensional image can be filtered by performing convolution between the source image and a kernel. In image processing, a kernel is a small matrix that, if convolved with an image, can filter the image for the desired quality. There are many different types of image filtering methods such as blurring, sharpening, embossing, etc. For each filtration technique, a different kernel matrix is used. In this research, the median blurring is performed successively for generating an image that has the least salt and pepper noise. In the case of median blurring, the median of all pixel values under the kernel area is first calculated. Then, the central element of that particular portion of the image is replaced by the median value. This process is performed for each pixel of the image.

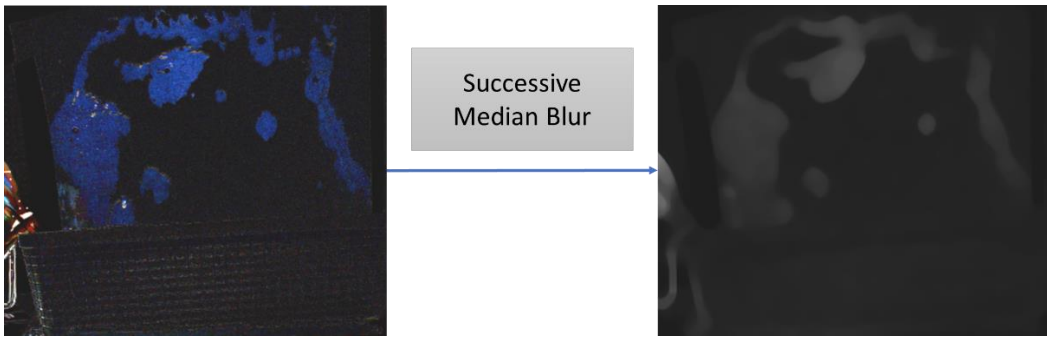


Figure 4.13: Result of applying a median blur filter

Once the image smoothing step is completed using the median blur filter, the blurred image is converted into a binary image by thresholding the image. The threshold values are found empirically based on the appearance of the part.

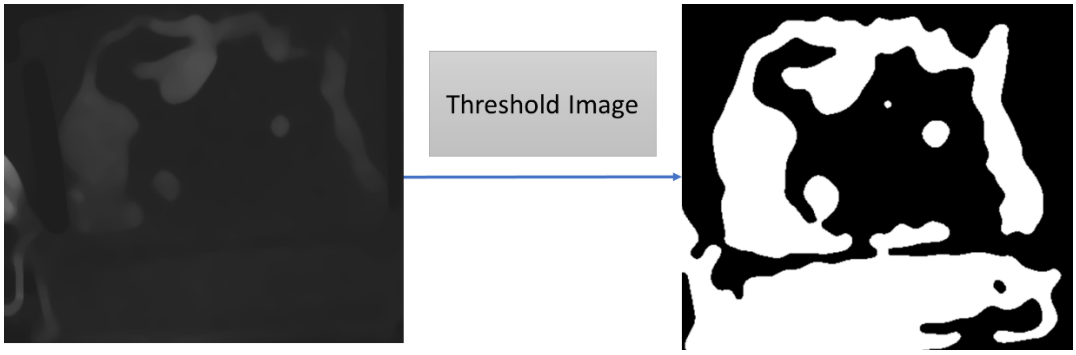


Figure 4.14: Resultant binary image after thresholding

The binary image is masked on top of the test image so that only the portion that is changed in the image is highlighted in the image.

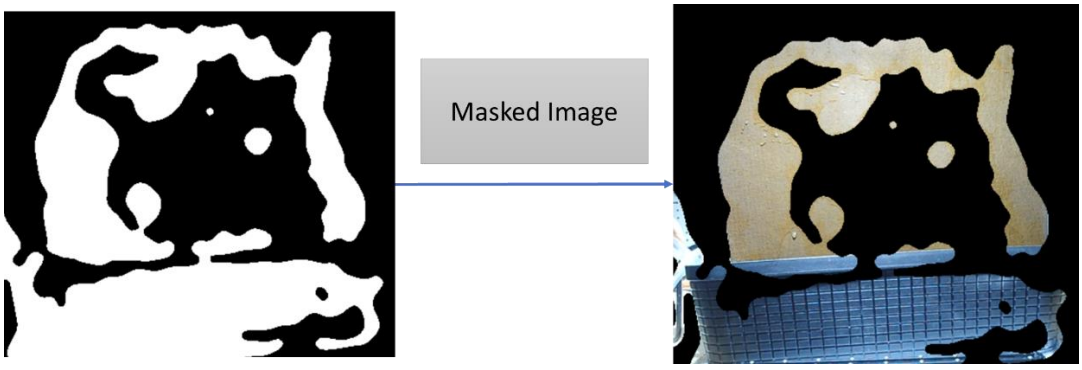


Figure 4.15: Masked image obtained by overlaying the detected areas over the original image

As seen in Figure 4.15, the masked image itself properly segments the dry area. However, this framework is not fully reliable as it may give false positives over areas with reflections and light shifts. To deal with this problem, an additional image processing routine based on Hue-Saturation-Value (HSV) filtering is developed to improve the robustness and reliability of water-break inspection.

HSV filtering routine: HSV is a similar color-space model as RGB. An HSV image has three channels, i.e., hue, saturation, and value. The hue channel represents the color of the pixel. The saturation channel represents the shades of gray in the pixel. The value channel represents the brightness or intensity of the pixel. Since the obtained masked image (Figure 4.15) is in the form of RGB, it is first converted into an HSV image as below:

$$Value, V = \max(R, G, B) \tag{4.3}$$

$$Saturation, S = \begin{cases} \dfrac{V - \min(R, G, B)}{V} & if\ V \neq 0 \\ 0 & if\ V = 0 \end{cases} \tag{4.4}$$

$$Hue, H = \begin{cases} \dfrac{60(G - B)}{V - \min(R, G, B)}, & if\ V = R \\ 120 + \dfrac{60(B - R)}{V - \min(R, G, B)}, & if\ V = G \\ 240 + \dfrac{60(R - G)}{V - \min(R, G, B)}, & f\ V = B \\ 0, & if\ R = G = B \end{cases} \tag{4.5}$$

Once the masked image is converted into an HSV image, the HSV image is filtered using a specific range of threshold values for all the three channels. The threshold values are found empirically in such a way that only the pixels that represent the dry region of the curved panel are kept. As shown in Figure 4.16, the proposed image subtraction method followed by HSV thresholding gives robust results in segmenting the water-broken regions in the image.

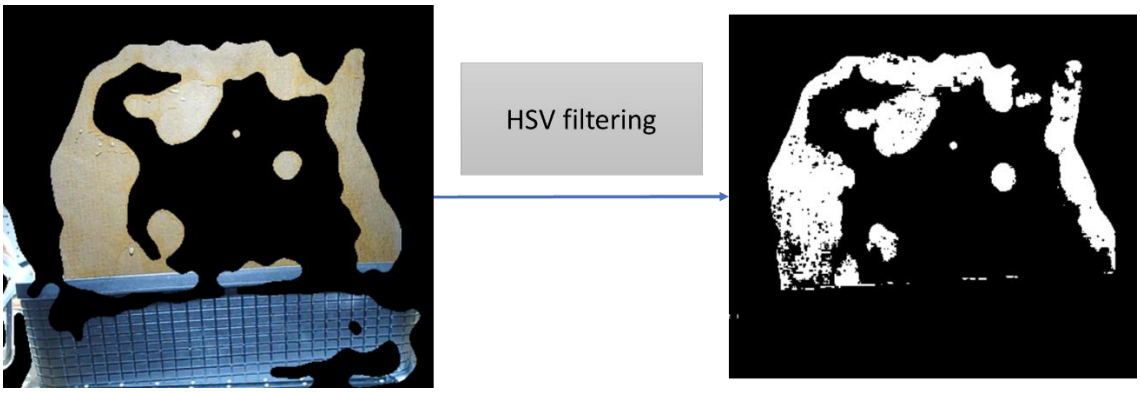


Figure 4.16: HSV filtering applied on the masked image

Finally, using the findcontours() [74] function in OpenCV, the contours in the image are detected. Contours are the shapes in the image that have a similar color or intensity. Since the HSV-filtered image is a binary image, the pixel values of the water-broken regions are 1, and all other pixel values are 0. The pixels that have a value of 1 are considered within the contour shape. These contours represent the location of the water-broken region in the pixel coordinates. Figure 4.17 shows the final result of the developed image processing algorithm for water-break inspection.

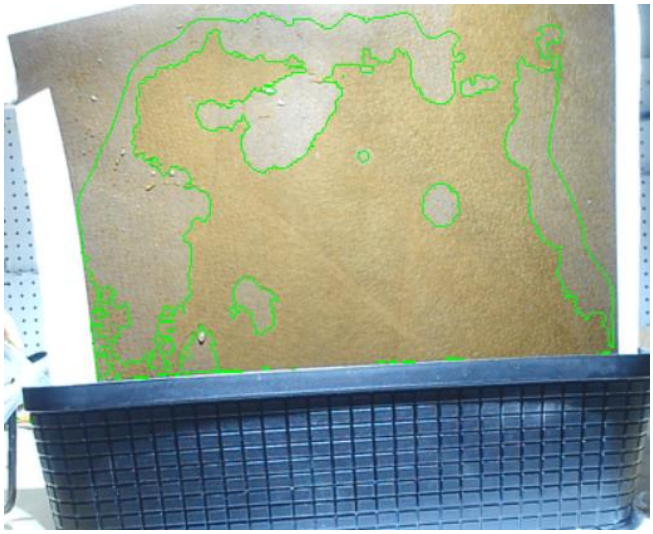


Figure 4.17: Detected contours (broken regions) during water-break inspection

4.4 Projection of part grids onto inspected images

After performing the water-break test and inspecting the surface, the failed (water-broken) areas must be abraded to remove contamination. Ideally, only the exact area of the broken regions must be abraded. However, considering the infinite possibilities in the shape and location of the failed regions, creating a robotic abrasion tool path directly based on the detected areas is not reliable. In this thesis, the part surface is divided into several grids, and an abrasion tool path is generated automatically for each grid. The proposed vision-based water-break inspection algorithm determines which grids contain broken regions. The corresponding grids are then abraded by the robot based on the generated tool path.

As shown in Figure 4.18, the region of interest (ROI) for water-break inspection is first defined in the 2D image of the part as captured by the Zivid camera. Using the corner pixels of the ROI, the point cloud for the selected ROI is extracted with the help of Zivid's python API. Since the nominal part coordinate system is kept at the same location of the robot's base coordinate system, this ROI point cloud is first transformed to the robot's coordinate system. The transformed ROI point cloud is denoted as ROI_{PC}. Furthermore, the image ROI is divided into multiple grids as presented in Figure 4.18 (right). The point cloud for each grid is extracted using the pixel points of each grid's corner. All the grid point clouds are then transformed to the robot's base coordinate system. The transformed grid point clouds are referred to as $Grids_{PC}$.

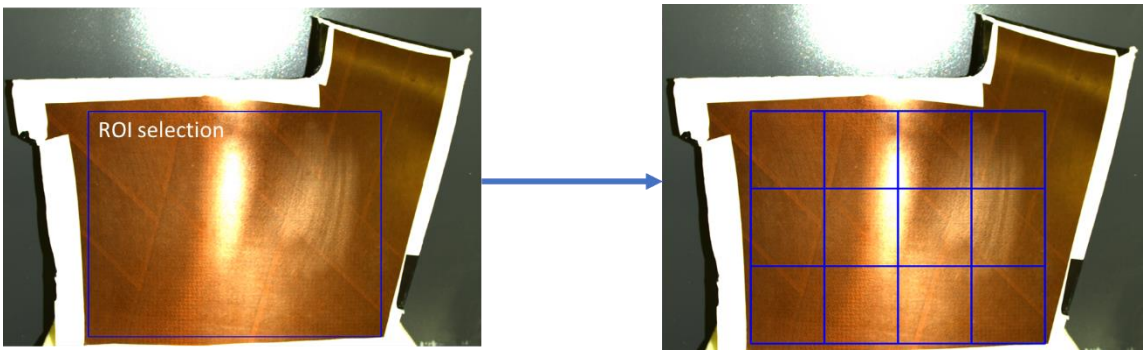


Figure 4.18: ROI selection and grid division of the RGB image captured by the Zivid camera

The grids shown in Figure 4.18 are valid only when the part is place at the nominal location. However, when the part is placed at a different location in the next robot cycle, the part grids will not map properly onto the image. In this research, a method is proposed to automatically map the part grids onto the image regardless of the part's location and orientation. In order to locate the exact region of interest on the new image of the arbitrarily placed part, the new pixel location of each grid must be known. To find the new pixel location of the grid corners, as discussed below, the pin hole camera model (Section 3.6) is used.

Eq. 3.11 can be used to find the 2D pixel location of a 3D point in the camera's coordinates. The nominal 3D location data of the grids' corners (in the camera's coordinate system) is extracted using its pixel points with Zivid's python API. This 3D data is first transformed to the workpiece coordinates (same as robot's base) and saved in the database at the ROI selection step. Now for each robot cycle, after the part localization step, using the part localization transformation ${}^{b}T_{w_{refined}}$ obtained in Section 3.8, the new grid corner's location with respect to robot's base (P_{grids_b}) is found by

$$P_{grids_b} = {}^{b}T_{w_{refined}}.\ P_{grids_w}. \tag{4.6}$$

where P_{grids_w} is the grid corner's location with respect to the composite panel. Eq. 4.6 provides the new 3D grid corner points in the base coordinates system. To find the 2D pixel location, the pin hole camera model needs this 3D points in the camera's coordinate system. The transformation between the Zivid camera and the robot's base (${}^{zivid}T_b$) is used to transform these grid corner points from the robot's base to the Zivid camera's coordinate system. The transformation matrix ${}^{zivid}T_b$ is found by inverting ${}^{b}T_{zivid}$. Finally, each grid corner point is transformed to the camera's coordinate system by:

$$P_{grids_{zivid}} = {}^{zivid}T_b.\ P_{grids_b}. \tag{4.7}$$

where $P_{grids_{zivid}}$ is the grid corner's location with respect to the Zivid camera.

Finally, Eq. 3.11 is used to find the new pixel coordinates of the grid's corners.

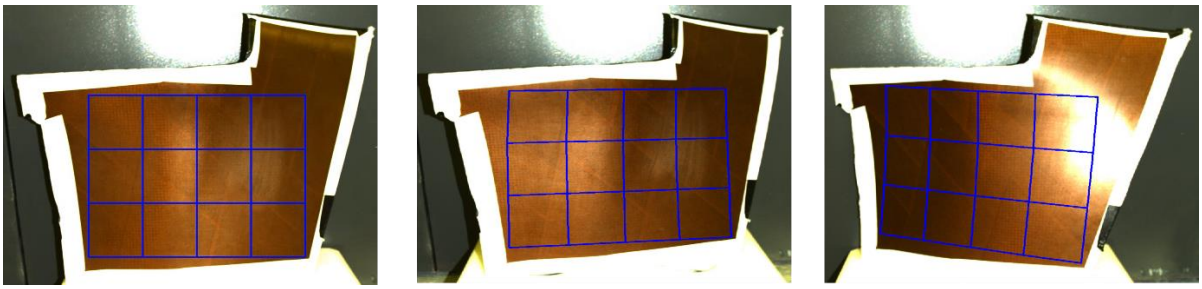


Figure 4.19: Grids reprojection for different poses of the curved panel

Using the reprojection of the new pixel locations of the grids, the grids can be overlaid on top of the composite panel's image as presented in Figure 4.19. Water-broken regions detected using the developed automated inspection framework can then be overlaid on top of the grids, as demonstrated in Figure 4.20. The grids that contain failed regions will need to be abraded. The proposed framework for generating an abrasion tool path for each grid is presented in the following section.

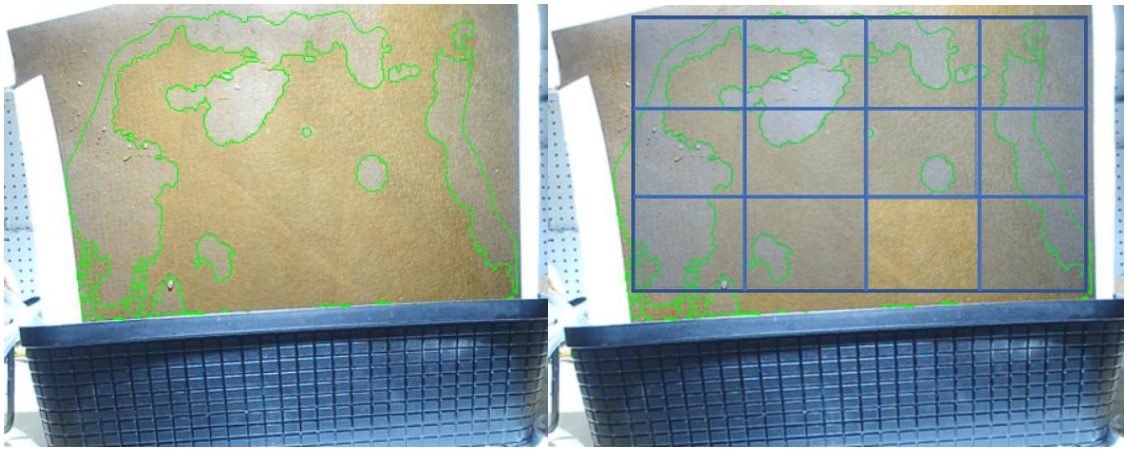


Figure 4.20: Grids overlaid on top of the water-break inspection results

4.5 Point cloud to B-spline surface reconstruction

In this thesis, it is assumed that abrasion tool paths are generated directly based on the captured point clouds of the part surface. The point clouds of the ROI (ROI_{PC}) and grid corners ($Grids_{PC}$) in the nominal part position and represented in the workpiece coordinate system are used to generate tool paths.

Due to measurement errors in 3D cameras, scanned point clouds are inevitably discontinuous and contain noise and outliers. That is, adjacent points can have abrupt changes in direction. The point cloud must be converted into a smooth surface that can be represented by a mathematical function to ensure smooth path generation for robotic abrasion. Mathematically, free-form surfaces can be represented in two general forms, i.e. implicit surface representation and parametric surface representation. In implicit representation, surfaces are described in the form of $S(X) = 0$, where X is a point on the surface that is implicitly described by the function S. For example, the implicit surface representation of a sphere is $S(X) = x^2 + y^2 + z^2 - 4 = 0$, where (x, y, z) is a point on the surface of the sphere. Implicit surface representation can become very complex in modeling free-form surfaces. In CAD and CAM systems, parametric representation is commonly used for modeling 3D surfaces.

In parametric representation, surfaces are described in the form of $X = S(u, v)$, where X is a point on the surface, and u and v are surface parameters. The u and v terms in this section should not be confused with the u and v of the camera's pixel coordinates. Some examples of parametric surfaces include Bezier surface [75], B-spline surface [76], and NURBS surface [38]. In this research, B-spline surfaces are used to model surfaces defined by the ROI and grid point clouds. Parametric surfaces can be obtained by taking a bidirectional net of control points [38]. In other words, parametric surfaces are simply the extension of parametric curves to a two-variable parametric domain u and v.

It is important to understand the basic formulation for B-spline curves before investigating B-spline surfaces. Equation 4.8 shows the parametric representation of a B-spline curve of degree p:

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i \tag{4.8}$$

where P_is are the control points and $N_{i,p}(u)$ terms are the B-spline basis functions defined on a knot vector with $m + 1$ knots:

$$Knot\ vector = \left\{ \underbrace{a, \dots, a}_{p+1}, u_{p+1}, \dots, u_{m-p-1}, \underbrace{b, \dots, b}_{p+1} \right\}, \tag{4.9}$$

where a and b are the knot repetition at the beginning and end of the curve. Most commonly, it is assumed that $a = 0$ and $b = 1$. The $N_{i,p}(u)$ basis function is defined by Cox-deBoor recursion algorithm [77]:

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p-1} - u_i} N_{i,p-1}(u) + \frac{u_{i+p} - u}{u_{i+p} - u_{i+1}} N_{i+1,p-1}(u) \tag{4.10}$$

$$N_{i,1}(u) = \begin{cases} 1 & if\ u_i \le u \le u_{i+1} \\ 0 & otherwise \end{cases} \tag{4.11}$$

Extending the B-spline curve in two dimensions generates a B-spline surface with surface parameters u and v.

The range of surface parameters u and v are kept from 0 to 1. The equation for a B-spline surface can be formulated by taking the tensor product of B-spline curves in two directions of the surface:

$$X = S(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,k}(u) N_{j,l}(v) p_{i,j}, \qquad u, v \in [0,1] \tag{4.12}$$

where, k and j are the degrees of curves along the u and v direction of the surface respectively. n and m represent the number of control points, and $N_{i,k}(u)$ and $N_{j,l}(v)$ are the univariate basis functions along the u and v directions, respectively. The u and v parameters make a 2D rectangular graph (Figure 4.21). Plugging in the values of u and v into Eq. 4.12 gives the corresponding point on the B-spline surface.

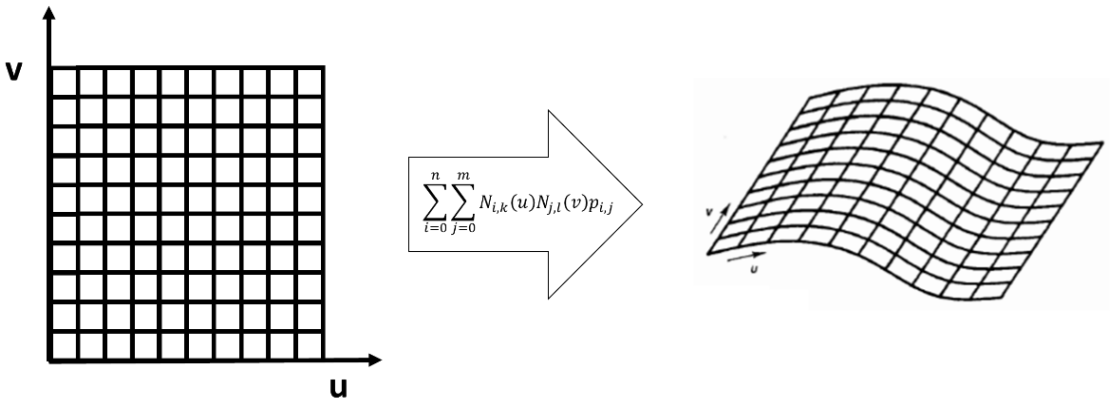


Figure 4.21: B-spline surface representation using *u* and *v* parameters

Geomdl python library

This thesis implements the B-Spline surface generation method using the geomdl library [78] in python. A surface, defined as an object in python, can be generated by using the class 'B-spline' of the geomdl library. The object takes in the input of control points for generating the B-spline surface. In this work, the point clouds representing the ROI and each rectangular grid are considered as the control points for generating the B-spline surface objects.

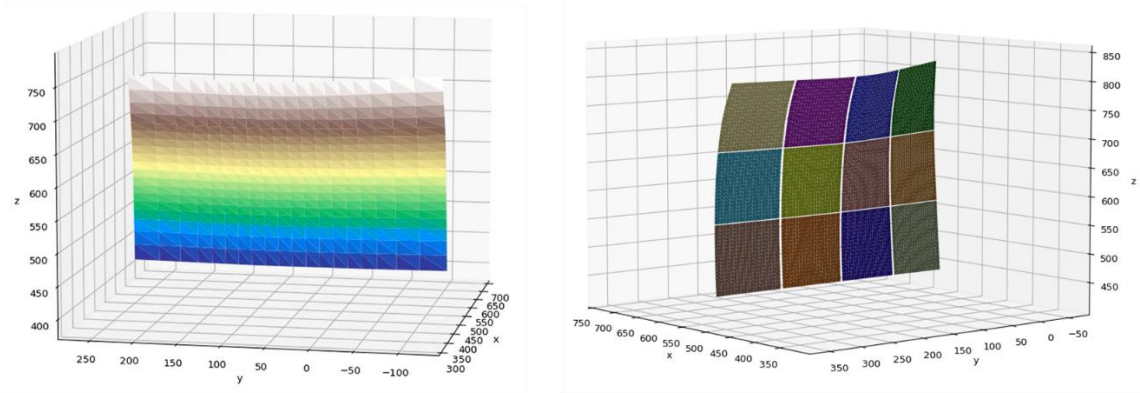


Figure 4.22: Generated B-spline surfaces for the ROI and grids

4.6 Surface derivatives and surface normal

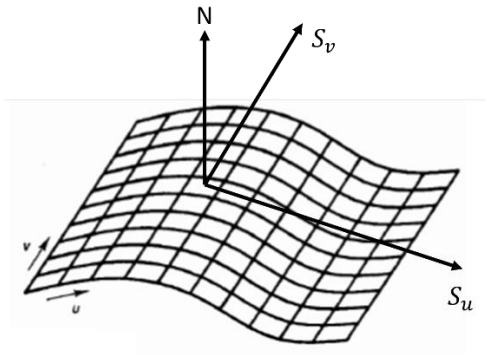


Figure 4.23: B-spline surface normal

The abrasion operation needs to be performed while keeping the tool normal to the surface at each part-tool contact point. This makes it important to identify the normal vectors for each point on the generated B-spline surfaces. Since a B-spline surface is a 2D parametrized entity, it can have two tangents along each parameter direction, i.e. a tangent along the u direction and a tangent along the v direction (Figure 4.23). The normal vector of the surface on a particular point can be obtained by taking the cross product of these two tangents at that particular point.

The derivative of a curve gives the tangent vector at each point along the curve. Similarly, the partial derivative of a surface with respect to the u parameter gives the surface tangent along the u direction, and the partial derivative of a surface with respect to the v parameter gives the surface tangent along the v direction. The surface normal can therefore be obtained as:

$$N = \frac{S_u(u,v) \times S_v(u,v)}{|S_u(u,v) \times S_v(u,v)|} \quad , \qquad 4.13$$

where N is the unit normal vector, $S_u(u,v)$ is the partial derivative of the surface $S(u,v)$ with respect to u, and $S_v(u,v)$ is the partial derivative of the surface $S(u,v)$ with respect to v. In this thesis, notation S_u is used for $S_u(u,v)$ and S_v for $S_v(u,v)$. The surface tangent along the u direction S_u is given by:

$$S_u = \frac{\partial}{\partial u} S(u,v) = \sum_{j=0}^{m} N_{j,l}(v) \left(\frac{\partial}{\partial u} \sum_{i=0}^{n} N_{i,k}(u) p_{i,j} \right) \tag{4.14}$$

$$= \sum_{i=0}^{n} N_{j,l}(v) \left(\frac{\partial}{\partial u} C_j(u) \right),$$

where

$$C_j(u) = \sum_{i=0}^{n} N_{i,p}(u) P_{i,j} \qquad j = 0, \dots, m. \tag{4.15}$$

Analogously, the surface tangent along the v direction S_v is obtained using similar formulation as Eq. 4.14 and 4.15.

4.7 Conversion of normal vector to Euler angles

In order to perform abrasion, the robot tool must be aligned with the surface normal obtained in the previous section. The Kuka robot uses Euler angle representation for defining the orientation of the end-effector with respect to the base coordinate system. Hence, a method for conversion from surface normal vector to robot's Euler angles must be obtained.

Figure 4.24 shows the coordinate system of the TCP, robot's base, and the surface normal at a particular point.

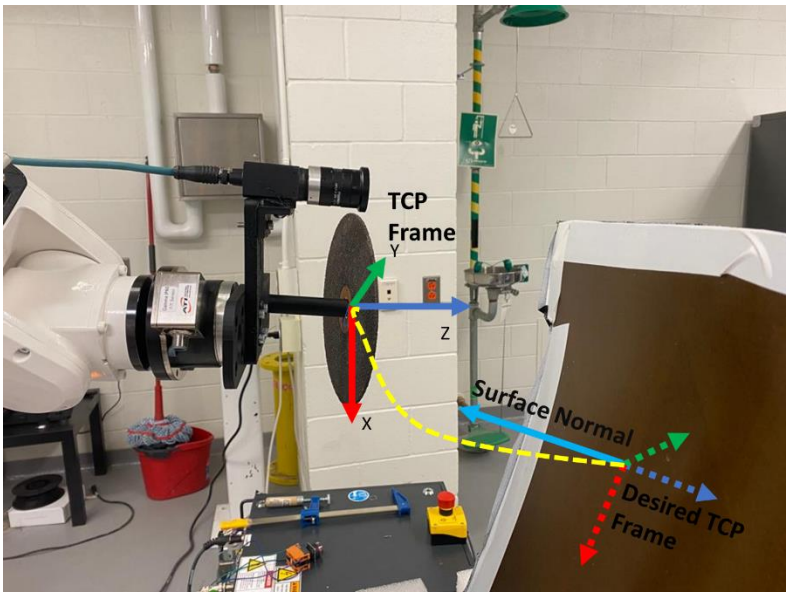


Figure 4.24: TCP coordinate system and surface normal

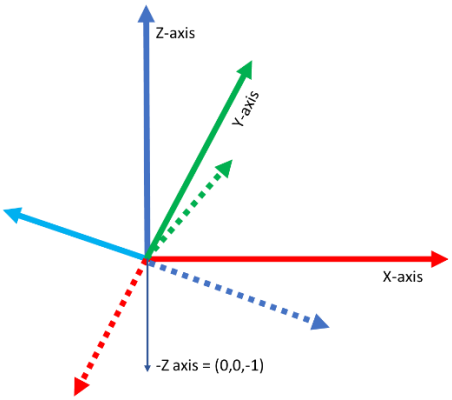
The cyan vector ($ai + bj + ck = 0$) in the image is the unit normal vector at a particular point on the curved surface. To make the end effector normal to the surface, we need to align the TCP's negative Z-axis (Blue-axis) with the cyan normal vector. For simplicity, the rotation matrix that can rotate the TCP coordinate system to align with the normal vector, all in the robot's base coordinate system, is first solved:

$$R_N^w = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{4.16}$$

The R_N^w is the rotation matrix that needs to be solved for finding the rotation between the robot's base (also, nominal workpiece frame) and the normal vector. Since we are aligning the negative Z component of the TCP axis to the normal vector, the Z column of the rotation matrix is replaced with the normal vector $(-a, -b, -c)$. Therefore, the rotation matrix R_N^w can be rewritten as:

$$R_N^w = \begin{bmatrix} R_{11} & R_{12} & -a \\ R_{21} & R_{22} & -b \\ R_{31} & R_{32} & -c \end{bmatrix}. \tag{4.17}$$

This fixes the rotation of the end effector in the Z direction. The transformation frame can have infinite rotation solutions if it is fixed only along one direction. However, the transformation can have a unique solution if two axes are fixed. Hence, the next step is to find the X column (R_{11}, R_{21}, R_{31}) and then the Y column (R_{12}, R_{22}, R_{32}). The cross product between any two vectors gives the perpendicular vector. For finding the Y column (new TCP Y axis), the cross product between the the Z component and the negative Z-axis vector $(0,0,-1)$ is taken.



	Base Frame
	Surface Normal
	Desired TCP Z axis = - Surface Normal
	Desired TCP Frame Y axis = Desired TCP Z axis X -Z axis
	Desired TCP Frame X axis = Desired TCP Y axis X Desired TCP Z axis

Figure 4.25: Desired TCP coordinate system

Hence, the new TCP Y axis (ay, by, zy) is found. The rotation matrix can be rewritten as:

$$R_N^W = \begin{bmatrix} R_{11} & ay & -a \\ R_{21} & ay & -b \\ R_{31} & ay & -c \end{bmatrix}. \qquad 4.18$$

For finding the X column (new TCP X axis (ax, bx, cx)) of the rotation matrix, the cross product between the Y column and the Z column is taken. Finally, the rotation matrix is obtained as:

$$R_N^w = \begin{bmatrix} ax & ay & -a \\ bx & by & -b \\ cx & cy & -c \end{bmatrix}. \quad 4.19$$

This rotation matrix is then converted into the Euler angles of sequence ZYX using Eq. 3.3.

The point clouds for each grid are converted into a smooth B-spline surface and the surface normal for each point on the B-spline surface can be determined. Along with that, the Euler angles for the robot's TCP for each point on the surface can also be determined. Hence, for each parametric point (u, v) on the B-spline surface, the Euler angles (A, B, C) for keeping the end-effector normal to the surface are known. The next step is to plan the area coverage tool path for the robot to perform the abrasion operation on the failed grids (water-broken regions) of the curved panel.

4.8 Abrasion tool path generation

In this section, the framework for generating an abrasion tool path for each grid is discussed. The abrasion operation can be performed using area clearance methods. As shown in Figure 4.26, there are many types of tool path clearance methods such as raster, spiral, offset, offset-spiral, etc.

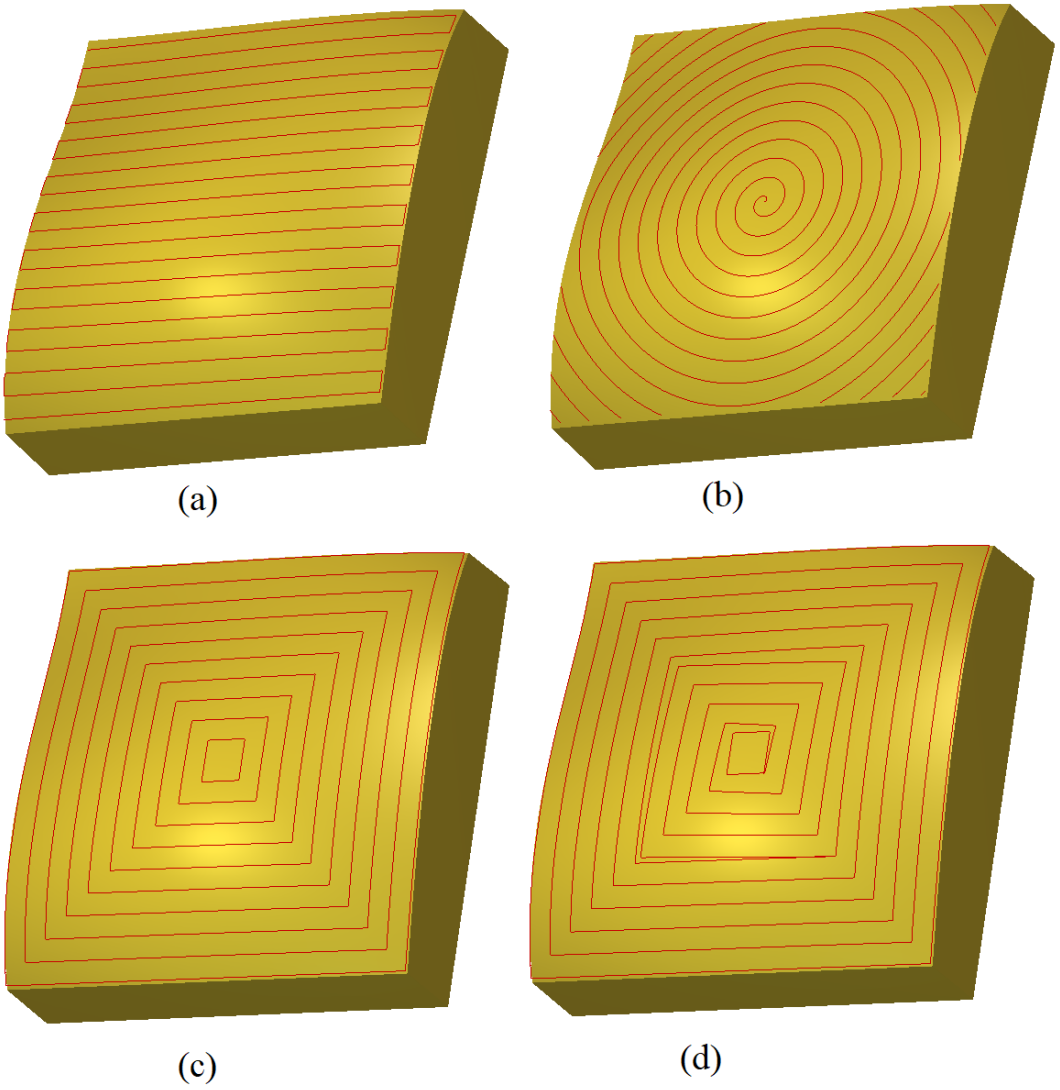


Figure 4.26: Area clearance tool path strategies, (a) Raster tool path, (b) Spiral tool path, (c) Offset tool path, (d) Offset-Spiral tool path

In Figure 4.26, the red lines represent the tool path. This research employs the raster area clearance tool path method (Figure 4.26.a) and develops algorithms to autonomously perform abrasion on the free form surface of each grid of the composite panel.

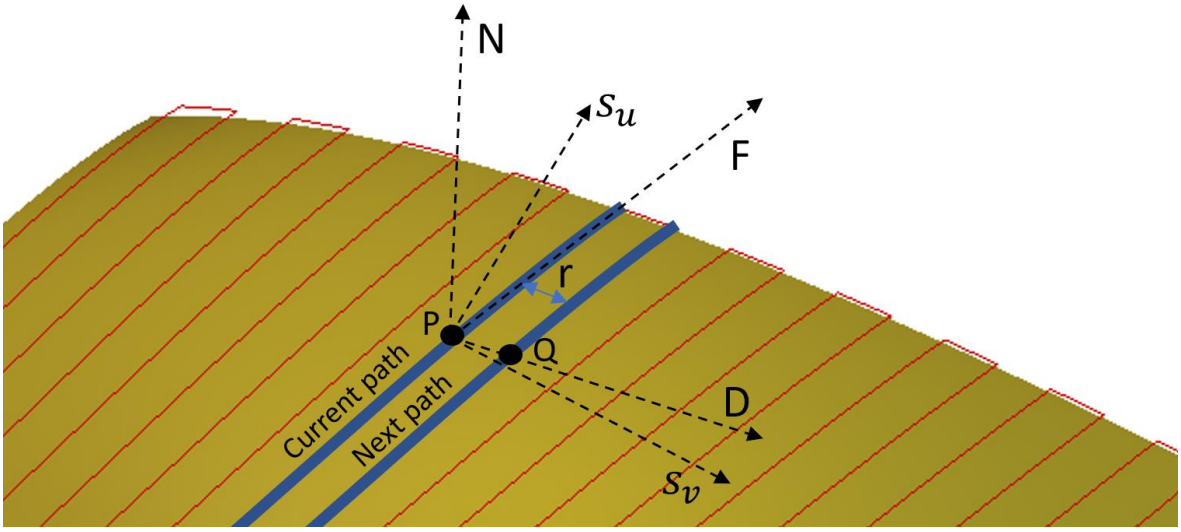


Figure 4.27: Local coordinates at a point P on the surface

As discussed in the previous section, a B-spline surface is represented in a parametric form, i.e. in terms of surface parameters u and v, while the tool path is defined in terms of an array of points in the Euclidean space (X, Y, Z). Hence, it is important to determine a relation between the Euclidean space and the parametric space. As presented in Figure 4.27, the relation between the Euclidean and parametric spaces can be used to find the change in surface parameters $(\Delta u, \Delta v)$ for the fixed distance r between two adjacent paths. This fixed distance *r* represents the step-over of the tool path.

Figure 4.27 shows a local coordinate system that can be generated on any point of the surface. This local coordinate system is created to find Δu and Δv. For the case shown in Figure 4.27, the local coordinate system is generated on a point P. Assume that points $P(S(u_p, v_p))$ and $Q(S(u_q, v_q))$ are on adjacent paths. F denotes the unit vector in the feed direction, and N is the unit normal vector at point P. Using the right-hand rule, unit vector D can be found by the cross product of vectors N and F. From surface differential geometry, we have:

$$Q - P \approx S_u \Delta u + S_v \Delta v. \qquad 4.20$$

Multiplying Eq. 4.20 by vector D and F yields:

$$\begin{bmatrix}(Q-P).D\\(Q-P).F\end{bmatrix} = \begin{bmatrix}S_u\Delta u.D + S_v\Delta v.D\\S_u\Delta u.F + S_v\Delta v.F\end{bmatrix}. \quad 4.21$$

Since point Q is in the direction of D and $\lfloor Q-P \rfloor = r$, the above equation can be rewritten as:

$$\begin{bmatrix}S_u.D + S_v.D\\S_u.F + S_v.F\end{bmatrix}\begin{bmatrix}\Delta u\\\Delta v\end{bmatrix} = \begin{bmatrix}r\\0\end{bmatrix}. \quad 4.22$$

Using 4.22,

$$\begin{bmatrix}\Delta u\\\Delta v\end{bmatrix} = \begin{bmatrix}S_u.D + S_v.D\\S_u.F + S_v.F\end{bmatrix}^{-1}\begin{bmatrix}r\\0\end{bmatrix} = \begin{bmatrix}\dfrac{r(S_v.F)}{(S_u.D)(S_v.F)-(S_v.D)(S_u.F)}\\\dfrac{-r(S_u.F)}{(S_u.D)(S_v.F)-(S_v.D)(S_u.F)}\end{bmatrix}. \quad 4.23$$

Using Eq. 4.23, the surface parameters for point Q are found as:

$$\begin{bmatrix}u_q\\v_q\end{bmatrix} = \begin{bmatrix}u_p + \Delta u\\v_p + \Delta v\end{bmatrix} \quad 4.24$$

The approximated change in the surface parameters can be found in any direction using the above equations. In this thesis, the first path is taken for $v = 0$; all adjacent paths are then found using the previous paths. Adding all of them together in a raster form generates an area coverage tool path for surface abrasion. This technique is used for generating the tool paths for all the B-spline surfaces of grids. Hence, the tool path data is known in terms of u and v. Using the tool path's u, v data, the pose data (X, Y, Z, A, B, C) for robot waypoints is constructed.

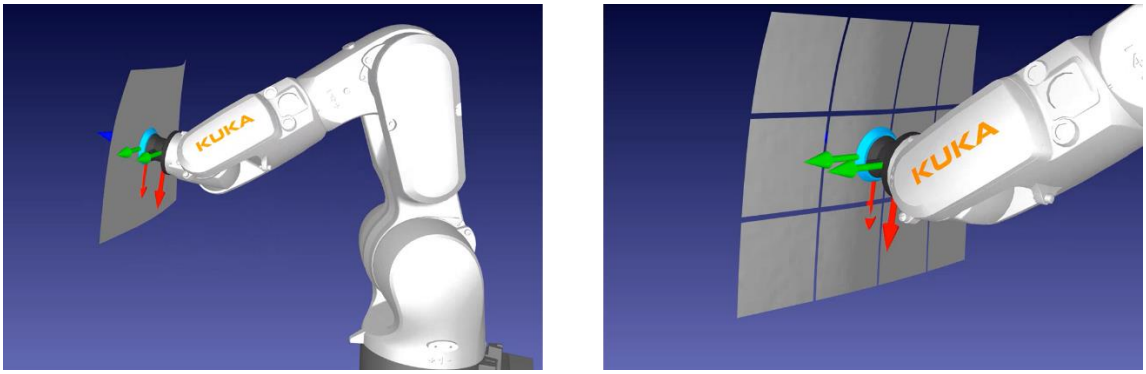


Figure 4.28: Visualization of robotic abrasion of each grid

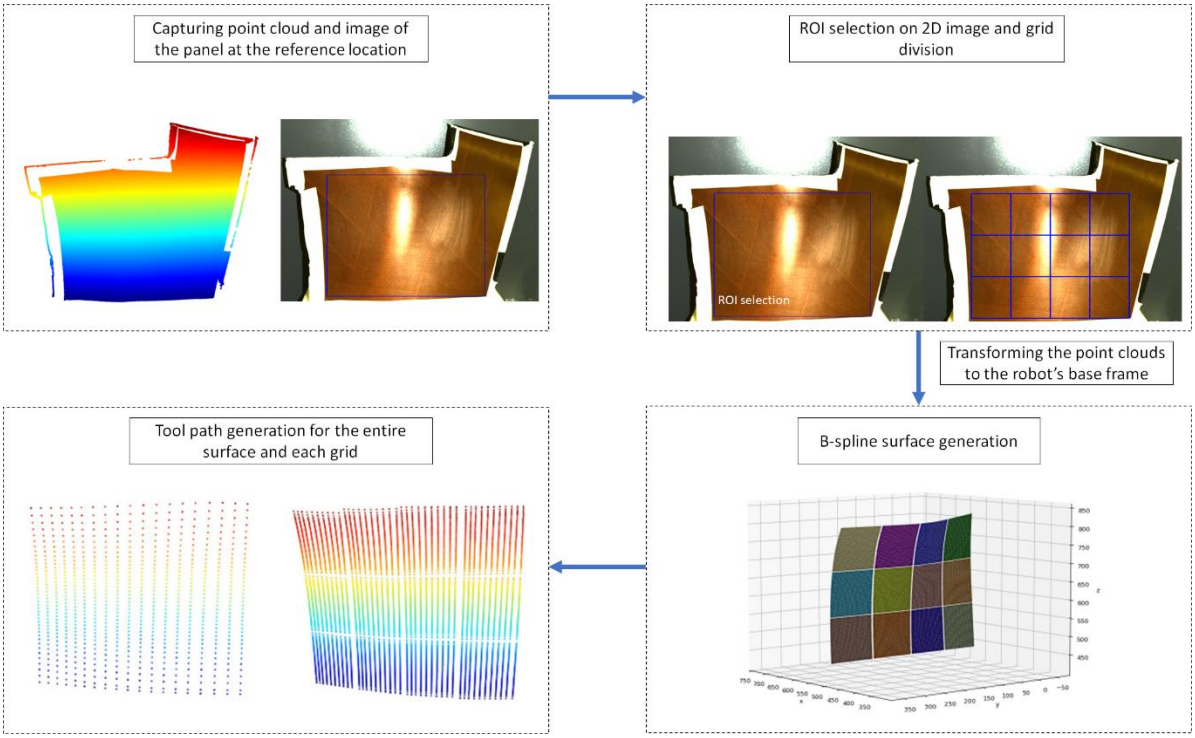


Figure 4.29 Summarizing the tool path generation process

To conclude, Figure 4.29 summarizes the procedure for tool path generation. The tool path robot poses are with respect to the nominal workpiece coordinate system 'w', which is at the robot's base coordinate system 'b'. Therefore, it is referred to as the nominal tool path pose (${}^{b}P_{nominal}$ or ${}^{w}P_{nominal}$). After each robot cycle when a new workpiece is arbitrarily placed in the cell, the workpiece's coordinate system is found using the developed part localization method (i.e. transformation ${}^{b}T_{w_{refined}}$ in Section 3.8). Once the part is localized, the next step is to correct tool paths for the abrasion operation. This can be achieved by setting the new base coordinate system in the Kuka program as the pose format of the transformation ${}^{b}T_{w_{refined}}$. Hence, the contaminated portion of the composite panel can be abraded.

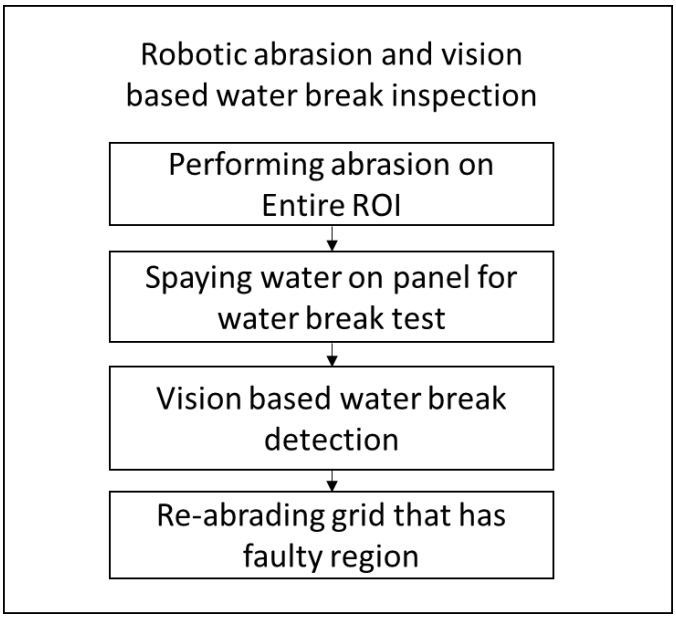


Figure 4.30: Robot cycle for water-break inspection and robotic abrasion

Putting everything together, the sequence of a robot cycle for water-break test and abrasion can be summarized as presented in Figure 4.30.

5 Conclusions and Future Work

Surface treatment processes such as priming, coating, and painting are commonly performed on free-form composite panels to improve their functional and aesthetic characteristics. Peel plies, release films, and mold release agents are often used in molding of free-form surfaces. These organic contaminations can stay on surfaces after the part is de-molded. The presence of contamination can weaken the adhesive bond of surface treatment processes. Therefore, a framework for autonomous surface contamination detection and robotic abrasion on free-form composite panels is developed in this thesis.

5.1 Summary of completed work and contributions

The fundamentals of the pin-hole camera model are used for the intrinsic calibration of a 2D Triton camera mounted onboard of the robot. To find the positioning relation between the robotic arm and the mounted 2D and 3D cameras, a three-point hand-eye calibration method is developed. The developed hand-eye calibration method is used to calibrate the cameras in both configurations, i.e., 'eye-in-hand' and 'eye-to-hand'. A python API for controlling the KUKA robot is developed using KUKA-EKI module. The API supports several basic robot control functions such as linear motion, PTP motion, setting base, setting TCP, getting current robot state, etc.

The ICP point cloud registration method is used for initial localization of the part. It was found that ICP registration method alone may not provide sufficient accuracy because of camera measurement errors and point registration inaccuracies. Therefore, a 2D vision-based method is also presented to further improve the accuracy of part localization. An image processing algorithm for automated detection of water-broken regions in the water-break test is developed. Since, lighting conditions can critically impact the robustness of the image processing algorithm, different types of lighting conditions were first tested. It was found that a Compact Fluorescent Lamp (CFL) with a diffuser produced uniform lighting

conditions for the test workpiece, hence it was used for the proposed image processing algorithm. The developed vision system can successfully detect the presence of hydrophobic contamination on the hydrophilic free-form surface of composite panels.

The hydrophobic contamination layer on the surface of the workpiece can be in any shape. Creating a robotic abrasion tool path directly based on the infinite possibilities in the shape and location of the detected layer of the contamination is not reliable. To facilitate robot tool path generation for abrasion, the composite panel's surface is divided into multiple grids. A B-spline representation of all grids is constructed using the 3D scanned point cloud of the free-form surface. The organized point clouds are used as the control points of the B-spline surfaces. Finally, a tool path generation method is developed to generate area scanning abrasion tool paths for abrading the composite panel's surface.

Pre-existing low-level libraries such as OpenCV, Open3D, and NURBS-Python are employed for performing tasks such as image processing, point cloud registration, and surface construction. Leveraging the usage of those low-level libraries, the techniques and algorithms for application-level tasks of autonomous part localization, water-break test, and abrasion path generation are developed in this thesis.

The developed framework including automated part localization, water-break inspection, and robotic abrasion allows for fully autonomous surface preparation of composite panels prior to priming and coating operations.

5.2 Future work and research directions

Future research can be pursued in two main areas, i.e. part localization and real-time compliance control for robotic abrasion. Since the part localization is performed using Iterative Closest Point (ICP) registration technique, it can converge to local minima. To solve the local minimum problem, the point clouds should be globally registered. Investigation of the effect of global point cloud registration algorithms is warranted. Once the point clouds are registered globally, the ICP algorithm can be used for the final refinement of the point cloud registration.

This thesis provides the methodology for abrasion tool path generation. However, an actual abrasion process needs active compliance control for ensuring uniform abrasion of the composite panel. Excessive contact stress between the abrasion tool and the part surface may lead to over-polishing; not sufficient contact stress, on the other hand, can lead to under-polishing. There are two methods that can be used for active compliance control: 1) through-the-arm system and 2) around-the-arm system. 'Through-the-arm' active compliance can be achieved by constrained motion of each robot joint. 'Around-the-arm' active compliance can be achieved by employing external end-effectors with active compliance control. Further research could explore these active compliance systems to ensure uniform abrasion.

6 Bibliography

[1] "Is It Clean? - Oil and Hydrophobic Films - Water Break Test - CTG Technical Blog," Aug. 2011. https://techblog.ctgclean.com/2011/08/is-it-clean-oil-and-hydrophobic-films-water-break-test/ (accessed Jun. 23, 2022).

[2] "Triton Industrial GigE Camera | LUCID Vision Labs." https://thinklucid.com/triton-gige-machine-vision/ (accessed Jun. 23, 2022).

[3] "See more. Do more. Zivid Two industrial 3D camera - Zivid." https://www.zivid.com/zivid-two (accessed Jun. 23, 2022).

[4] A. Joubair and I. A. Bonev, "Non-kinematic calibration of a six-axis serial robot using planar constraints," *Precision Engineering*, vol. 40, pp. 325–333, 2015, doi: 10.1016/j.precisioneng.2014.12.002.

[5] "Probing systems and software." https://www.renishaw.com/en/probing-systems-and-software--12466 (accessed May 09, 2022).

[6] "Technologies -- Touch Probes | Marposs." https://www.marposs.com/eng/application/technologies-touch-probes (accessed May 09, 2022).

[7] "Touch probes and vision systems from HEIDENHAIN." https://www.heidenhain.com/products/touch-probes-and-vision-systems (accessed May 09, 2022).

[8] "Tactile Probe Systems - Mitutoyo." https://www.mitutoyo.com/products/coordinate-measuring-machines/cmm-probe-and-change-rack-options/tactile-probe-systems/ (accessed May 09, 2022).

[9] H. Srinivasan, O. L. A. Harrysson, and R. A. Wysk, "Automatic part localization in a CNC machine coordinate system by means of 3D scans," *The International Journal of Advanced Manufacturing Technology 2015 81:5*, vol. 81, no. 5, pp. 1127–1138, May 2015, doi: 10.1007/S00170-015-7178-Z.

[10] Z. Zheng, Y. Ma, H. Zheng, Y. Gu, and M. Lin, "Industrial part localization and grasping using a robotic arm guided by 2D monocular vision," *Industrial Robot*, vol. 45, no. 6, pp. 794–804, Dec. 2018, doi: 10.1108/IR-06-2018-0128/FULL/PDF.

[11] Ø. Skotheim, M. Lind, P. Ystgaard, and S. A. Fjerdingen, "A flexible 3D object localization system for industrial part handling," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3326–3333, 2012, doi: 10.1109/IROS.2012.6385508.

[12] M. Rajaraman, M. Dawson-Haggerty, K. Shimada, and D. Bourne, "Automated workpiece localization for robotic welding," *IEEE International Conference on Automation Science and Engineering*, pp. 681–686, 2013, doi: 10.1109/COASE.2013.6654062.

[13] K. Okarma and M. Grudziński, "The 3D scanning system for the machine vision based positioning of workpieces on the CNC machine tools," *2012 17th International Conference on Methods and Models in Automation and Robotics, MMAR 2012*, pp. 85–90, 2012, doi: 10.1109/MMAR.2012.6347906.

[14] A. Hernandez, A. Maghami, and M. Khoshdarregi, "A Machine Vision Framework for Autonomous Inspection of Drilled Holes in CFRP Panels," in *2020 6th International Conference on Control, Automation and Robotics, ICCAR 2020*, Apr. 2020, pp. 669–675. doi: 10.1109/ICCAR49639.2020.9108000.

[15] J. Fan, L. Ma, and Z. Zou, "A registration method of point cloud to CAD model based on edge matching," *Optik (Stuttg)*, vol. 219, p. 165223, Oct. 2020, doi: 10.1016/J.IJLEO.2020.165223.

[16] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992, doi: 10.1109/34.121791.

[17] A. Purabgola, S. Rastogi, G. Sharma, and B. Kandasubramanian, "Surface Preparation for Structural Adhesive Joints," in *Structural Adhesive Joints: Design, Analysis, and Testing*, K. L. Mittal and S. K. Panigrahi, Eds. John Wiley & Sons, Ltd, 2020, pp. 1–34. doi: 10.1002/9781119737322.CH1.

[18] Hugh R. Gregg and Michael P. Meltzer, "Contamination analysis unit," US5521381A, 1996 Accessed: May 06, 2022. [Online]. Available: https://patents.google.com/patent/US5521381A/en

[19] "ASTM F22-21 Standard Test Method for Hydrophobic Surface Films by the Water-Break Test." Accessed: May 06, 2022. [Online]. Available: https://www.astm.org/f0022-21.html

[20] B. Ellis, “The water break test,” *Circuit World*, vol. 31, no. 4, 2005, doi: 10.1108/03056120510603143/FULL/PDF.

[21] B. F. Monzyk *et al.*, “Concept Evaluation of Visual Cleaning Performance Indicators (VCPI) For Real Time Cleaning Verification (PP-1117) to,” Mar. 2002.

[22] I. Kolenov, A. Galuza, A. Belyaeva, S. Mizrakhy, P. Nesterov, and A. Savchenko, “Influence of Contamination with Silicone Release Agent on the Ellipsometric Parameters of CFRP Surface in the Sub-THz Range,” in *2021 IEEE 3rd Ukraine Conference on Electrical and Computer Engineering, UKRCON 2021 - Proceedings*, Aug. 2021, pp. 56–59. doi: 10.1109/UKRCON53503.2021.9575665.

[23] R. Ecault, M. Boustie, and L. Berthe, “EXTENDED NDT FOR THE QUALITY ASSESSMENT OF ADHESIVE BONDED CFRP STRUCTURES,” Nov. 2011, Accessed: May 06, 2022. [Online]. Available: https://www.researchgate.net/publication/267364769

[24] A. R. G. Jay M. Amos, “Surface contamination detection method and apparatus,” US7126123B1, 2006 Accessed: May 06, 2022. [Online]. Available: https://patents.google.com/patent/US7126123B1/en

[25] M. T. Traband and D. J. Medeiros, “CAD-directed programming of a vision-based inspection system,” *Journal of Manufacturing Systems*, vol. 8, no. 3, pp. 215–223, Jan. 1989, doi: 10.1016/0278-6125(89)90043-5.

[26] J.-M. Chen and J. A. Ventura, “Vision-based shape recognition and analysis of machined parts,” *http://dx.doi.org.uml.idm.oclc.org/10.1080/00207549508930140*, vol. 33, no. 1, pp. 101–135, 2007, doi: 10.1080/00207549508930140.

[27] C. Wang and C.-H. Yu, “Machine Vision Based Inspection of Textile Fabrics*,” *IAPR Workshop on Machine Vision Applications*, Oct. 1994.

[28] C. Bradley and S. Kurada, “Industrial inspection employing a three dimensional vision system and a neural network classifier,” *IEEE Pacific RIM Conference on Communications, Computers, and Signal Processing - Proceedings*, pp. 505–508, 1995, doi: 10.1109/PACRIM.1995.519580.

[29] C. Koch, S. Paal, A. Rashidi, Z. Zhu, M. König, and I. Brilakis, “Achievements and Challenges in Machine Vision-Based Inspection of Large Concrete Structures:,” *http://dx.doi.org.uml.idm.oclc.org/10.1260/1369-4332.17.3.303*, vol. 17, no. 3, pp. 303–318, Nov. 2016, doi: 10.1260/1369-4332.17.3.303.

[30] D. Li *et al.*, “Automatic defect detection of metro tunnel surfaces using a vision-based inspection system,” *Advanced Engineering Informatics*, vol. 47, p. 101206, Jan. 2021, doi: 10.1016/J.AEI.2020.101206.

[31] O. A. Menendez, M. Perez, and F. A. A. Cheein, “Vision based inspection of transmission lines using unmanned aerial vehicles,” *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, vol. 0, pp. 412–417, Jul. 2016, doi: 10.1109/MFI.2016.7849523.

[32] H. Q. Bong, Q. B. Truong, H. C. Nguyen, and M. T. Nguyen, “Vision-based Inspection System for Leather Surface Defect Detection and Classification,” *NICS 2018 - Proceedings of 2018 5th NAFOSTED Conference on Information and Computer Science*, pp. 300–304, Jan. 2019, doi: 10.1109/NICS.2018.8606836.

[33] A. Maghami, M. Salehi, and M. Khoshdarregi, “Automated vision-based inspection of drilled CFRP composites using multi-light imaging and deep learning,” *CIRP Journal of Manufacturing Science and Technology*, vol. 35, pp. 441–453, Nov. 2021, doi: 10.1016/J.CIRPJ.2021.07.015.

[34] Vinesh. Raja and K. J. (Kiran J. Fernandes, *Reverse engineering : an industrial perspective*. Springer, 2008.

[35] R. J. Abella, J. M. Daschbach, and R. J. McNichols, “Reverse engineering industrial applications,” *Computers & Industrial Engineering*, vol. 26, no. 2, pp. 381–385, Apr. 1994, doi: 10.1016/0360-8352(94)90071-X.

[36] Y. Qie, S. Bickel, S. Wartzack, B. Schleich, and N. Anwer, “A function-oriented surface reconstruction framework for reverse engineering,” *CIRP Annals*, vol. 70, no. 1, pp. 135–138, 2021, doi: 10.1016/J.CIRP.2021.04.016.

[37] T. Várady, R. R. Martin, and J. Cox, “Reverse engineering of geometric models—an introduction,” *Computer-Aided Design*, vol. 29, no. 4, pp. 255–268, Apr. 1997, doi: 10.1016/S0010-4485(96)00054-1.

[38] L. Piegl and W. Tiller, *The NURBS Book*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. doi: 10.1007/978-3-642-97385-7.

[39] N. Leal, E. Leal, and J. W. Branch, “Simple method for constructing NURBS surfaces from unorganized points,” *Proceedings of the 19th International Meshing Roundtable, IMR 2010*, pp. 161–175, 2010, doi: 10.1007/978-3-642-15414-0_10.

[40] W. Wu, Y. Hu, and Y. Lu, "Parametric Surface Modelling for Tea Leaf Point Cloud Based on Non-Uniform Rational Basis Spline Technique," *Sensors 2021, Vol. 21, Page 1304*, vol. 21, no. 4, p. 1304, Feb. 2021, doi: 10.3390/S21041304.

[41] A. Gálvez, A. Iglesias, and J. Puig-Pey, "Iterative two-step genetic-algorithm-based method for efficient polynomial B-spline surface reconstruction," *Information Sciences*, vol. 182, no. 1, pp. 56–76, Jan. 2012, doi: 10.1016/J.INS.2010.09.031.

[42] X. Liu, Y. Li, S. Ma, and C. H. Lee, "A tool path generation method for freeform surface machining by introducing the tensor property of machining strip width," *Computer-Aided Design*, vol. 66, pp. 1–13, Sep. 2015, doi: 10.1016/J.CAD.2015.03.003.

[43] G. Elber and E. Cohen, "Toolpath generation for freeform surface models," *Computer-Aided Design*, vol. 26, no. 6, pp. 490–496, Jun. 1994, doi: 10.1016/0010-4485(94)90070-1.

[44] A. Can and A. Ünüvar, "A novel iso-scallop tool-path generation for efficient five-axis machining of free-form surfaces," *The International Journal of Advanced Manufacturing Technology 2010 51:9*, vol. 51, no. 9, pp. 1083–1098, May 2010, doi: 10.1007/S00170-010-2698-Z.

[45] S. Singh Randhawa and J. Singh Saini, "Tool path generation for free-form surfaces using B-spline surface," *Lecture Notes in Mechanical Engineering*, vol. 15, pp. 57–71, Apr. 2014, doi: 10.1007/978-81-322-1859-3_6/TABLES/1.

[46] L. Zhang, Y. Han, C. Fan, Y. Tang, and X. Song, "Polishing path planning for physically uniform overlap of polishing ribbons on freeform surface," *The International Journal of Advanced Manufacturing Technology 2017 92:9*, vol. 92, no. 9, pp. 4525–4541, May 2017, doi: 10.1007/S00170-017-0466-Z.

[47] Y. Han, L. Zhang, M. Guo, C. Fan, and F. Liang, "Tool paths generation strategy for polishing of freeform surface with physically uniform coverage," *The International Journal of Advanced Manufacturing Technology 2017 95:5*, vol. 95, pp. 2125–2144, Nov. 2017, doi: 10.1007/S00170-017-1281-2.

[48] "OpenCV: Camera Calibration and 3D Reconstruction." https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html (accessed Jun. 23, 2022).

[49] "Lens Distortion: What Every Photographer Should Know." https://clickitupanotch.com/lens-distortion/ (accessed Jun. 23, 2022).

[50] A. E. Conrady, “Decentred Lens-Systems,” *Monthly Notices of the Royal Astronomical Society*, vol. 79, no. 5, pp. 384–390, Mar. 1919, doi: 10.1093/MNRAS/79.5.384.

[51] “OpenCV.” https://opencv.org/ (accessed Jun. 23, 2022).

[52] “OpenCV: Camera Calibration.” https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html (accessed Jun. 23, 2022).

[53] W. Jakob, “Calibration Patterns Explained – calib.io,” Nov. 15, 2018. https://calib.io/blogs/knowledge-base/calibration-patterns-explained (accessed Aug. 17, 2022).

[54] “OpenCV: Perspective-n-Point (PnP) pose computation.” https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html (accessed Jun. 23, 2022).

[55] G. Godin, M. Rioux, and R. Baribeau, “Three-dimensional registration using range and intensity information,” *https://doi.org/10.1117/12.189139*, vol. 2350, pp. 279–290, Oct. 1994, doi: 10.1117/12.189139.

[56] K. Pulli, “Multiview registration for large data sets,” *Proceedings - 2nd International Conference on 3-D Digital Imaging and Modeling, 3DIM 1999*, pp. 160–168, 1999, doi: 10.1109/IM.1999.805346.

[57] Y. Chen and G. Medioni, “Object modeling by registration of multiple range images,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2724–2729, 1991, doi: 10.1109/ROBOT.1991.132043.

[58] G. Blais and M. D. Levine, “Registering Multiview Range Data to Create 3D Computer Objects,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 820–824, 1995, doi: 10.1109/34.400574.

[59] P. J. Neugebauer, “Geometrical cloning of 3D objects via simultaneous registration of multiple range images,” *Proceedings - 1997 International Conference on Shape Modeling and Applications, SMI 1997*, pp. 130–139, 1997, doi: 10.1109/SMA.1997.634890.

[60] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A Modern Library for 3D Data Processing,” Jan. 2018, Accessed: Jun. 23, 2022. [Online]. Available: http://arxiv.org/abs/1801.09847

[61] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Commun ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975, doi: 10.1145/361002.361007.

[62] S. Umeyama, “Least-Squares Estimation of Transformation Parameters Between Two Point Patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991, doi: 10.1109/34.88573.

[63] M. Sezgin and B. Sankur, “Survey over image thresholding techniques and quantitative performance evaluation,” *https://doi.org/10.1117/1.1631315*, vol. 13, no. 1, pp. 146–165, Jan. 2004, doi: 10.1117/1.1631315.

[64] “OpenCV: Smoothing Images.” https://docs.opencv.org/3.4/d4/d13/tutorial_py_filtering.html (accessed Aug. 17, 2022).

[65] “OpenCV: Arithmetic Operations on Images.” https://docs.opencv.org/3.4/dd/d4d/tutorial_js_image_arithmetics.html (accessed Aug. 17, 2022).

[66] W. Jasim and R. Mohammed, “A Survey on Segmentation Techniques for Image Processing,” *Iraqi Journal for Electrical and Electronic Engineering*, vol. 17, no. 2, pp. 73–93, Dec. 2021, doi: 10.37917/IJEEE.17.2.10.

[67] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986, doi: 10.1109/TPAMI.1986.4767851.

[68] “OpenCV: Morphological Transformations.” https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html (accessed Aug. 17, 2022).

[69] “OpenCV: Geometric Transformations of Images.” https://docs.opencv.org/3.4/da/d6e/tutorial_py_geometric_transformations.html (accessed Aug. 17, 2022).

[70] D. Martin, “A Practical Guide to Machine Vision Lighting,” 2013.

[71] H. M. Subhash, “Full-field and single-shot full-field optical coherence tomography: A novel technique for biomedical imaging applications,” *Advances in Optical Technologies*, 2012, doi: 10.1155/2012/435408.

[72] Y. Dattner and O. Yadid-Pecht, “Low light cmos contact imager with an integrated poly-acrylic emission filter for fluorescence detection,” *Sensors*, vol. 10, no. 5, pp. 5014–5027, May 2010, doi: 10.3390/S100505014.

[73] H. A. Afify, “Evaluation of change detection techniques for monitoring land-cover changes: A case study in new Burg El-Arab area,” *Alexandria Engineering Journal*, vol. 50, no. 2, pp. 187–195, Jun. 2011, doi: 10.1016/J.AEJ.2011.06.001.

[74] S. Suzuki and K. A. be, “Topological structural analysis of digitized binary images by border following,” *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, Apr. 1985, doi: 10.1016/0734-189X(85)90016-7.

[75] “Bézier surface - Wikipedia.” https://en.wikipedia.org/wiki/B%C3%A9zier_surface (accessed Aug. 18, 2022).

[76] “B-spline - Wikipedia.” https://en.wikipedia.org/wiki/B-spline (accessed Aug. 18, 2022).

[77] C. de Boor, “On Calculating with B-Splines,” *JOURNAL OF APPROXIMATION THEORY*, vol. 6, pp. 50–62, 1972.

[78] O. R. Bingol and A. Krishnamurthy, “NURBS-Python: An open-source object-oriented NURBS modeling framework in Python,” *SoftwareX*, vol. 9, pp. 85–94, Jan. 2019, doi: 10.1016/J.SOFTX.2018.12.005.