Framework for vision-based robotic corner cleaning of window frames

by

Tzu-Jan Tung

A thesis submitted in partial fulfillment of the requirements of the degree of

Master of Science
in
Construction Engineering and Management

Department of Civil and Environmental Engineering
University of Alberta

Abstract

Corner cleaning is a critical step after welding of window frames to ensure aesthetic quality. Current methods to clean weld seams are limited in adaptability and quality, increasing rework, cost, and waste. This is largely attributable to the use of CNC cutting machines in combination with manual inspection and seam cleaning. Because the system relies on predefined window designs, cleaning processes are ineffective when dealing with manufacturing imperfections. However, as the blueprint of Industry 4.0 is becoming clear, the development of automation is proceeding at a rapid pace, and new technologies such as robots and sensors are playing a lead role in driving innovation within the manufacturing field. In this thesis, a vision-based robotics system is proposed that enhances adaptability to variability in weld cleaning while ensuring high quality and precision through an approach that combines robotic arms and computer vision in place of existing manual-based methods. The developed system uses edge detection to locate the window and identify its orientation, image segmentation techniques with a pre-trained Mask R-CNN model to detect the window weld seam, and a vision-guided robot manipulator to control the moving path for the robotic arm. The working process begins with the window location to obtain the rough position for the purpose of guiding the robot toward the window target, followed by image processing and detection in conjunction with instance segmentation techniques to segment the target area of the weld seam, and, finally, the generation of cleaning paths for further robot manipulation. The proposed robotic system is validated in a simulated environment as well as in a real-world scenario, with the results obtained demonstrating the effectiveness and adaptability of the proposed system.

Keywords: machine vision; window manufacturing; vision-guided robotics; deep learning.

Preface

This thesis is an original work by Tzu-Jan Tung. No part of this thesis has been previously published.

Acknowledgements

Throughout the writing of this thesis, I have received a great deal of support and assistance.

My sincere appreciation first goes to my supervisor Dr. Mohamed Al-Hussein for his continuous support, guidance, encouragement, and inspiration. His insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. I am highly grateful to Dr. Pablo Martinez for his valuable guidance throughout my academic study. I might not be able to complete it without his instruction and help.

I would also like extend my thanks to my friends and colleagues in the Construction Engineering and Management group as well as the Department of Civil and Environmental Engineering at the University of Alberta. To Jonathan Tomalty for his effort on editing and providing valuable suggestions especially on such tight timeline. To Cheng-Hsuan Yang for the knowledge support of robotics arms. To Nan Zhang and Anas Itani for providing knowledge support of window manufacturing in practice. To Bo Xiao and Yiheng Wang for the support of Python programming. Thanks also go to my peers with whom I have insightful and thoughtful discussions. All the suggestions and discussions mean a lot to me.

Finally, I would like to thank my family and friends for their support and sympathetic ear.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

This chapter presents an overview of the current practice for corner cleaning of welded window frames, followed by a definition of the problem motivating the research described in this thesis, and, finally, an outline of the thesis.

1.1. Background

With the emergence of the concept of Industry 4.0 and the ongoing development of related technologies, automation in the construction industry has drawn increasing attention in recent years. Successful applications in the manufacturing industry in particular have inspired the pursuit of further automated applications in the industrialized construction field. With the integration of novel technologies such as robotic arms, computer vision, and various real-time sensors, repetitive tasks can be executed more flexibly and accurately [1]. In the window manufacturing industry, for instance, a typical plastic framing production line may consist of a combination of semi-automated or automated processes: an initial cutting station where frames are cut to length using saws, a hot-plate welding station where individual frame elements are fused together to create the window frames, and a final corner-cleaning station where excess material on the corner of the joined window frame elements is removed [2]. This excess material is generated as pressure is applied to different individual elements to secure the frame during the welding process. As such, the material to be removed is always located on the exterior surface of the weld area, around the perimeter of the window frame. An example of the corner of a window frame with its main elements prior to cleaning can be seen in Figure 1.

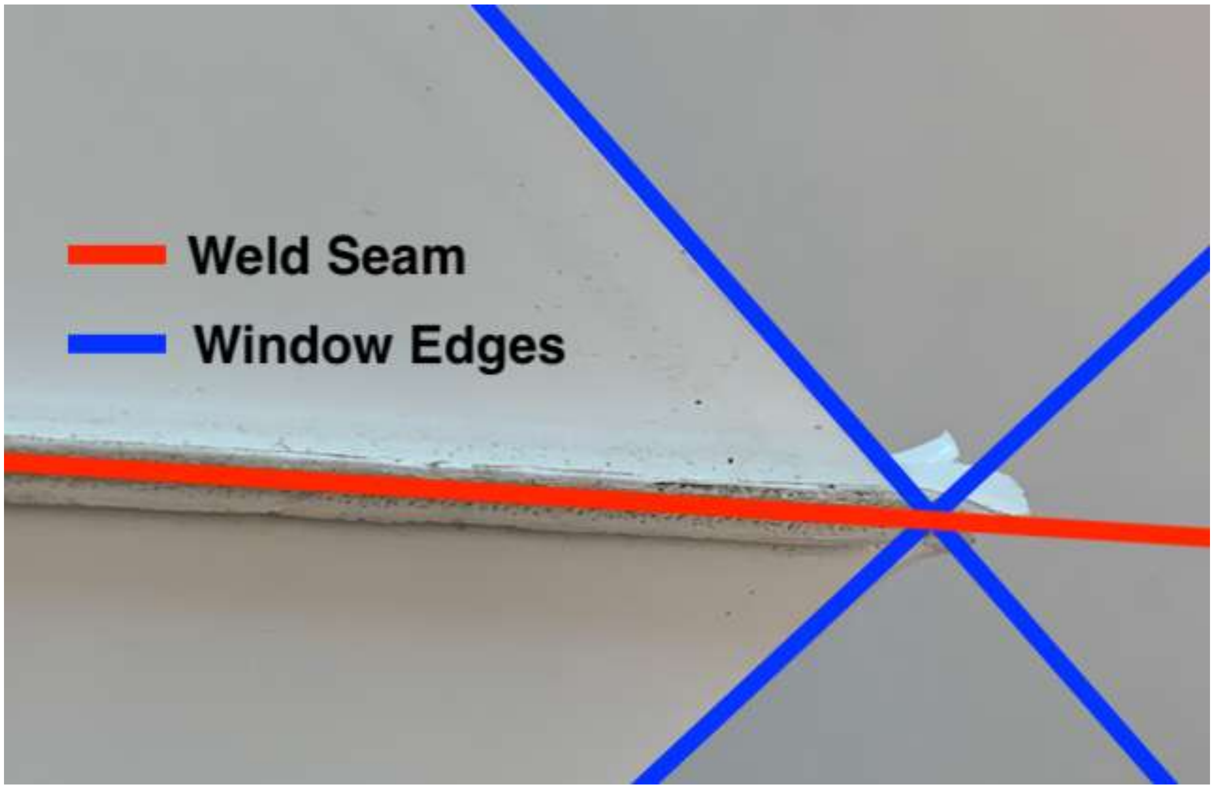


Figure 1. Example of the weld area of a window frame.

In this study, the window manufacturing procedure is based on the current practice implemented at All Weather Windows, which is Canada's largest privately-owned window and door manufacturer. Their current practice for removing excess material from window frame seams, which is assumed to be representative of the practice of other large-scale window and door manufacturers, proceeds in two sequential steps: (1) a series of knifing and milling tools, driven by linear actuators, are responsible for cleaning the window frame surface and leaving it as aesthetically pleasing as possible so that the weld seam is barely noticeable; and (2) a manual inspection ensures the quality of the cleaning process, and, if needed, a worker removes any residual seam materials manually. However, this procedure has crucial flaws, including low adaptability to variations in window placement and unpredictable quality due to the volume of weld seam material to be cleaned being unknown.

In the first part of the cleaning process, since the machine tasks are planned in advance based on idealistic CAD window models, the efficiency of the tool to clean the weld seam depends on the correct and accurate placement of the window, as well as on the geometric accuracy and overall quality of the anterior processes (cutting and welding operations). Window placement, which is in most cases a manual operation, becomes more complicated with larger window frames, as they are more difficult to handle. Furthermore, the weld seam characteristics, namely volume and shape, may change depending on the precision of the anterior hot-plate welding process. Under the current practice with its inflexible execution of the corner-cleaning process, even a minute variance between the design and the manufacturing output, such as 1 mm in seam thickness or 1-degree difference in weld angle, can lead to an unsatisfactory outcome of the cleaning process. These limitations challenge the reliability of current corner-cleaning machines to achieve the desired outcome and threaten product integrity. In fact, if product tolerances are considered, in introducing additional variability to the product status, the current approach to automated corner cleaning is rendered ineffective and incapable of ensuring product quality.

In the second part of the cleaning process, the quality of manually removing excess material highly depends on the skill of individual workers, so the quality is uncertain and difficult to control due to individual differences. In some instances, the frame may become damaged to the extent that it fails to meet the quality standards of a marketable product. Although workers exhibit better adaptability to varying and unpredictable product specifications than do machines, this manual process has the disadvantages of uncertain quality and high labour cost. Due to these inherent disadvantages, workers only play an assistive role in the cleaning procedure. However, the manual inspection of the window frames is still time consuming, tedious, and an inefficient use of labour resources. In summary, the current practice for clean weld seams in window manufacturing is suboptimal and needs to be improved.

1.2. Motivation and Objectives

Although the drawbacks of current practice are apparent, to the author's best knowledge, no previous study has investigated a dual approach combining computer vision and robots to clean window weld seams. As mentioned above, the placement difference can easily lead to irreversible damage to window frames, and non-standardized manually cleaning increases the uncertainty of the final quality. Moreover, rule-based machine operations and manual fine work that relies highly on worker skill and performance are limited in their capacity to accommodate complex variables within this manufacturing activity. In this context, the aim of the present research is to explore the prospect of applying computer vision and robotic manipulation to window manufacturing, particularly the weld-cleaning activity.

Automation in manufacturing has progressed significantly within the last decade, such that the use of robots and vision assistance on production lines can no longer be considered novel ideas. Combined with the rapid evolution of deep learning algorithms, which are now being widely applied in a number of research areas, manufacturing is gradually becoming more closely aligned with the vision of Industry 4.0, and several Industry 4.0 approaches have already achieved impressive outcomes in manufacturing [3]. Building on the success of these previous applications of Industry 4.0 principles, in the present research a framework of vision-based robotic window weld cleaning system is proposed to overcome the flaws in the current corner-cleaning procedure. To enhance its adaptability to accommodate variability and ensure cleaning quality, vision techniques and robotics are incorporated in the framework to identify and locate window frames, detect the weld seams on window frames, and generate moving paths for the robotic arm.

Specifically, the present study seeks to achieve the following two objectives:

1) To develop a working framework of a vision-based robotic system that guides the cleaning tool operation following window frame welding.
2) To investigate the feasibility of Mask R-CNN image segmentation for window weld seam identification.

1.3. Thesis Organization

This thesis consists of six chapters. The current chapter provides a summary of the current state of window cleaning, pointing out the flaws of current practice with regards to adaptability and quality. The motivation and objectives are also outlined. Chapter 2 reviews the literature on window manufacturing, identifying the segmentation techniques, algorithms (including Mask R-CNN, and vision-guided robot manipulation approaches already applied in this area. The methodology of the present study is presented in chapter 3. Chapter 4 describes the development of the framework for vision-guided robotic manipulation, including the details of each working module in the vision-based robotics system framework. The validation and results are presented in chapter 5. Serval experiments are carried out to validate the framework proposed. Chapter 6 summarizes the research, including the outcomes, suggest avenues of future work, and the limitations of the research are discussed.

Chapter 2 Literature Review

This chapter describes the two core techniques used in the thesis and reviews the related literature. First a summary of the evolution and current practice of window manufacturing is presented, where it is shown how several studies have explored the prospect of combining vision systems to guide robotic arms to make them better able to adapt to various situations with flexibility and accuracy. Among the advanced technologies that have been developed, vision-guided robot manipulation and image segmentation are the two core technologies employed in this study, and they are elaborated on as follows.

2.1. Window frame manufacturing

Windows are crucial components of a well-functioning building. The comfort level of the internal environment is largely governed by the relationship between openings and various physical variables, such as lighting, acoustics, flow of air, and thermal energy [4]. To adapt to various climatic conditions, strategies such as specialized window designs may be devised as passive design measures to better handle the outer environment [5]. For example, Takada et al. (2021) used various elemental technologies to improve the thermal insulation properties of windows [6]. In current practice, depending on various contributing factors, the materials used for window frame production may include wood, aluminum, wood-cladding, fibreglass, polyvinyl chloride (PVC), and other composite materials.

Due to the importance of windows in a building, the quality of the manufactured product must be taken seriously. In this regard, Kermani et al. (2015) proposed an application model for operational planning and order control in profile door and window manufacturing [7]. Another recent study investigated the cutting process for aluminum bars in door and window manufacturing aiming to reduce material waste [8]. Another recent study presented a Lean

Manufacturing 4.0 approach to aid decision-making in window manufacturing to optimize the manufacturing process, and, ultimately, the quality of the final product [9]. The same study also describes how, with the development of Industry 4.0, the window manufacturing process is continuing to mature and to gain momentum, particularly with the introduction of new technologies.

The methods used to connect junctions are also crucial. For PVC window frame production—i.e., the research target of the present study—hot-plate welding is widely used due to its simplicity. The quality of the window junction after hot-plate welding, of course, has a significant impact on the final product quality [2]. The welding process, it should be noted, consists of four stages: squaring, heating, welding, and cooling, and each stage has a direct impact on the function of the final product. Not only does the welding process itself require accurate execution, but the subsequent weld seam cleaning task must be carried out with precision. However, the current practice has two main drawbacks—low adaptability and unpredictable quality—that can have impact on the final product.

As an alternative to existing cleaning methods applied in window manufacturing, such as the use of CNC machine and manual removal, several studies have sought to address the problem of weld seam cleaning using different tools, including laser techniques, waterjet, and even pre-simulation for reducing contaminants in advance. The material under investigation in these studies has typically been steel or alloy. To reduce material waste and enhance porosity formation, AlShaer et al. researched the effects of short pulse laser surface cleaning for automotive component production, which includes the cleaning of the surface after welding as part of the production [10]. In another study, laser cleaning was applied as both a pre-treatment and a post-treatment for butt welding of HSLA steel plates [11]. Vu et al., meanwhile, conducted a study on the effect of pulsed fiber laser on the surface of both steel and aluminum alloy by estimating the roughness of the surface [12]. Jin et al. mentioned the application of fiber laser techniques in surface cleaning in their discussion of the development of fiber-laser-based photoacoustic microscopy (PAM) [13]. Applications of laser cleaning in manufacturing has also been identified as an important technique in the context of surface contamination and cleaning [14]. As mentioned above, waterjet has also been applied as a cleaning technique in research investigating its influence on welding quality [15]. Other researchers have sought to address root causes for surface contamination using simulation techniques. For instance, modelling has been applied to predict the impact of linear friction welding on the surface cleaning of a Ti-6AI-4V T-Joint [16]. Another study modelled the plastic flow of welding ends in order to optimize the self-cleaning effect of linear friction welding, which is an effective method for removing surface contaminants [17]. As demonstrated in the summary above, although weld seam cleaning has been the subject of a few research studies, the use of robotics for weld seam cleaning in window manufacturing has yet to be explored. Emerging technology applications with the potential to improve window manufacturing and thereby enhance the overall quality of PVC window manufacturing in accordance with the principles of Industry 4.0 are discussed in the next section.

2.2. Vision-guided robots in manufacturing

Within the Industry 4.0 paradigm, a wide range of vision techniques have been applied in inspection and quality control to enhance the positioning accuracy of robotic manipulation, including photogrammetry, stereo vision, structured light, time of flight, and laser triangulation [18]. Machine vision and robotics are at the forefront of innovation in manufacturing, and robot guidance applications have drawn increasing attention within the manufacturing field. The most intuitive and simple application of machine vision is to mount cameras to monitor a working area.

For example, Shah et al. (2017) proposed an approach for recognition of butt welding joints. They mounted a fixed camera at a working area to locate the position of the weld seam and create a path for the robot in 2D coordinates, thereby simplifying the coordinate transformation for situations in which the vertical distance between object and camera is fixed [19]. Abdelaal et al. (2019) conducted an experiment to control an industrial robotic arm to pick and place moving objects using a real-time computer vision system that detects the object's exact location on a moving conveyor belt. Their system overcomes the challenge of dynamic picking point estimation with a stereo vision system comprising two fixed cameras, thereby demonstrating the efficient use of multiple fixed cameras monitoring the working area to precisely locate positions in dynamic situations [20].

Kleppe et al. (2017) proposed an approach that uses a 3D camera and a 2D camera on a robotic arm to facilitate automated assembly tasks. The system first finds a rough estimate of the target object position using the 3D camera. A 2D camera on a robotic arm is then used to generate a fine estimate of the target. Their approach overcomes the viewing range limitation of robotic cameras by first estimating the rough location in order to guide the robotic arm to the working area. Their method also demonstrates a case of fixed cameras and robotic cameras operating cooperatively to position the target with high accuracy [21]. Levines et al. (2018) presented a deep learning-based approach to hand–eye coordination for robotic grasping. Since the target objects were scattered randomly in their study, optimal real-time control was achieved by implementing deep learning on the robotic arm grasping data. They also explored and validated the ability of deep learning and vision techniques to accommodate variability [22].

In recent decades, the advantages of using robotics for welding procedures, including quality improvement and worker safety enhancement, have been demonstrated. Not only has vision-aided robotic welding been applied, but several other applications in welding-related scenarios have also been explored [23]. For instance, the use of supporting sensor technologies, such as vision-sensing and self-learning neural control, have been deployed in robotic welding [24]. Moreover, advanced vision techniques have been implemented in the control process for robotic welding to facilitate seam tracking and identification, which is an important task in efforts to automatically direct the robots along the correct welding path [25].

With the emergence of Industry 4.0, robotics and deep learning now play a central role in the ongoing evolution of manufacturing, with their application in seam identification to realize intelligent welding being an example of particular relevance to the present study. Dinham & Fang (2013), for instance, found that the combination of computer vision and a welding robot aided by camera monitoring provided highly accurate weld seam identification [26]. Along with the ongoing advancement of deep learning algorithms, image-processing techniques have improved, as has the accuracy of welding seam recognition. As recent studies have noted, these developments underscore the potential of computer vision to accommodate varying and dynamic environments based on their capacity to identify multiple types of seams [19, 20]. Meanwhile, as Rout et al. (2019) have noted, novel algorithms such as neural networks and fuzzy approaches, generally applied in the control design, are now bringing promising results in terms of improved accuracy, and this trend is expected to continue as new soft computing techniques are introduced [21, 22]. Although these previous applications have performed well in welding-related cases, weld seam corner-cleaning in particular, which is the target of the present research, has received little attention in the literature.

2.3. Image segmentation

For accurate detection of seams on window frames, high-precision object-detection techniques are essential. Image segmentation is a general designation of techniques used in digital image processing to define objects and boundaries by dividing the image into meaningful segments. The recent success of image segmentation has largely been a function of the rapid development of deep learning models, which has led to significant improvements in feature extraction and accuracy. Deep learning is generally proficient in tackling the issue of feature learning, and this allows it to easily overcome the domain knowledge requirement [27]. Among the notable algorithms for image segmentation, region-based convolutional networks models, such as Mask R-CNN, are widely deployed for instance segmentation, and they have exhibited high accuracy in object detection and image segmentation applications. Mask R-CNN was first proposed in 2017, building on the existing Faster R-CNN model by adding a third branch to generate an object mask output along with the original outputs, as well as a bounding box and a class label, and replacing the region of interest (ROI) pooling module with an ROI Align module. This algorithm uses a selective search technique to analyze characters on images, such as colour or texture, in order to locate possible object regions based on the analysis results.

Applications of the Mask R-CNN model in manufacturing are numerous, but the primary use has been for inspection. Due to the crucial role of assembly in automated manufacturing, the advantages of Mask R-CNN, including its ability to perform multi-object classification and identification, can simplify the inspection process by easily finding defects [28]. A similar application has been in vehicle appearance component inspection [29]. Moreover, for objects with specific rather than generic shapes, Mask R-CNN is capable of yielding highly accurate results.

With regard to applications in weld seam identification, although in the case of weld seams the shape is usually not as specific as in the examples discussed above, the Mask R-CNN algorithm has still been shown to be highly effective. For instance, Xia et al. (2020) proposed a Mask R-CNN-based method to detect the melt pool area within a visual sensing system for a robotic wire arc additive manufacturing system. Their method demonstrated the potential of detecting objects with a non-specific shape [30]. Another recent study proposed a micro-gap weld detection method to overcome the difficulty of recognizing welds with widths less than 40 μm with the utilization of Mask R-CNN [31]. Jin et al. (2020) also proposed a method based on Mask R-CNN to recognize small batches, and their findings demonstrated the potential for more complex welding seam location, given the pixel-level accuracy of their results [32]. Yang et al. (2020) used identification and classification techniques in a trajectory planning method to obtain the pixel-wise coordinate data for the small target objects to be scanned. This data was then extracted using a cloud-point method based on Mask R-CNN [33]. Overall, these examples show that deep learning algorithms have performed well in vision-based weld identification. For the present study, Mask R-CNN is applied to weld seam identification in window manufacturing.

2.4. Summary and Research Gap

Industry 4.0 expands the scope of manufacturing innovation, particularly in the form of automated manufacturing solutions. Moreover, recent technological advancements provide the opportunity to integrate vision techniques and robot manipulation for tackling the dynamic operations and working conditions characteristic of automated manufacturing. Existing literature in this area touches on current practice of window manufacturing, vision-guided robotics applications in manufacturing (i.e., welding-related cases), and applications of the Mask R-CNN deep learning model for image segmentation. The literature review above reveals a research gap, which is investigated in this thesis. Although several welding-related applications of vision-guided robots have been explored, to the author's knowledge the welding seam cleaning application has yet to be investigated. As mentioned in Chapter 1, the current practice has notable flaws in terms of low adaptability and unpredictable quality, so innovation of the cleaning process is needed in order to ensure a high-quality final product. In this context, to provide an adaptive solution for welding seam cleaning, a vision-based robotic system integrating the use of robotic arms and computer vision for image segmentation is a promising solution to fill the gap.

Chapter 3 Methodology

This chapter describes the methodology used in this research. A design science research (DSR) methodology is adopted to develop a framework of vision-based robotic system capable of locating, identifying, and cleaning the weld seams at window corners. The goal of DSR is to develop an artifact—something that has a meaningful use and improves the understanding of a problem at the identified research gaps. The process of developing an artifact includes a rigorous procedure of literature review, then developing the artifact in accordance with predefined evaluation methods in a structured and replicable manner while clearly communicating its outputs [34]. The artifact developed in this research is a vision-based framework to enable robotic corner cleaning of window frames. Following development of the artifact, an alternative method to the industrial machinery currently employed for corner cleaning in window manufacturing is devised that increases the flexibility and reliability of the operation. The methods applied in this research are divided into four stages as illustrated in Figure 2.

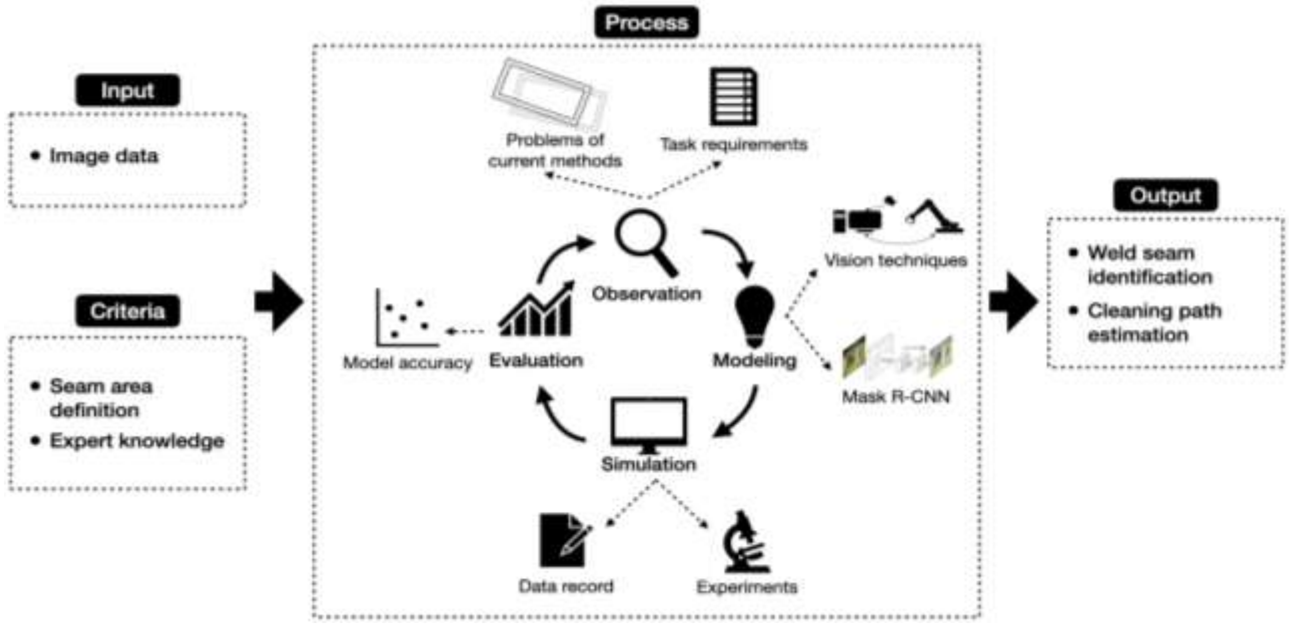


Figure 2. Overview of the proposed research methods.

In the **observation** stage, the purpose is to develop a concise understanding of PVC window manufacturing by investigating the welding process and identifying issues with regard to current window cleaning practice and task requirements. From a “big picture” perspective, these flaws associated with the current practice provide highly relevant information that can be used as the basis for selecting a suitable technique to address the problems. The **modelling** stage involves model building using Mask R-CNN and vision techniques. Based on the information obtained at the observation stage, the structure of a vision-based robotic system is developed using computer vision algorithms for image segmentation, and using vision techniques to monitor and control the working environment. The **simulation** stage includes several experiments for validating the proposed approach. The data generated through the validation process in this stage is recorded and analyzed. In the **evaluation** stage, the proposed approach is evaluated and tested in a real-world scenario. The resulting performance is compared with the hypothesis to evaluate the efficiency of the proposed approach, as well as to identify any limitations and potential measures to improve it. Definition of seam area and expert knowledge are required inputs for the model training and operation involved, and the final output is an assessment of the performance of the weld seam identification and cleaning path estimation.

Chapter 4 Vision-based Robotic Corner-cleaning of Window frames

This chapter describes the framework proposed in this research based on the combination of image segmentation and vision-guided robot manipulation. The vision-based robotic framework consists of three modules—window and location identification, weld seam detection, and cleaning path generation.

4.1. Overview of framework

The purpose of the framework presented in this study is to guide a robotic arm in executing weld cleaning on window frames with the assistance of computer vision. As shown in Figure 3 and as mentioned above, the proposed framework comprises three consecutive modules: (1) window and location identification; (2) weld seam detection; and (3) cleaning path generation. The first module uses the Hough transform technique to perform accurate edge detection from the image captured from a fixed camera in order to define the location of the target. In the second module, the camera attached to the robotic arm captures a close-range image of the target. The framework employs a pre-trained Mask R-CNN model to process the image for the purpose of seam detection, and generates a mask to quantify the area of weld seam where there is material to be removed. In the last module, based on the information obtained in the previous stage, the framework calculates a cleaning path for the robotic arm and transforms the coordinates from the 2D image to 3D (real world). The three modules are described in greater detail in the following sections. The Python scripts used for training the Mask R-CNN model, it should be noted, are developed based on the scripts from Matterport available on GitHub [35]. The complete scripts can be found in Appendix A.

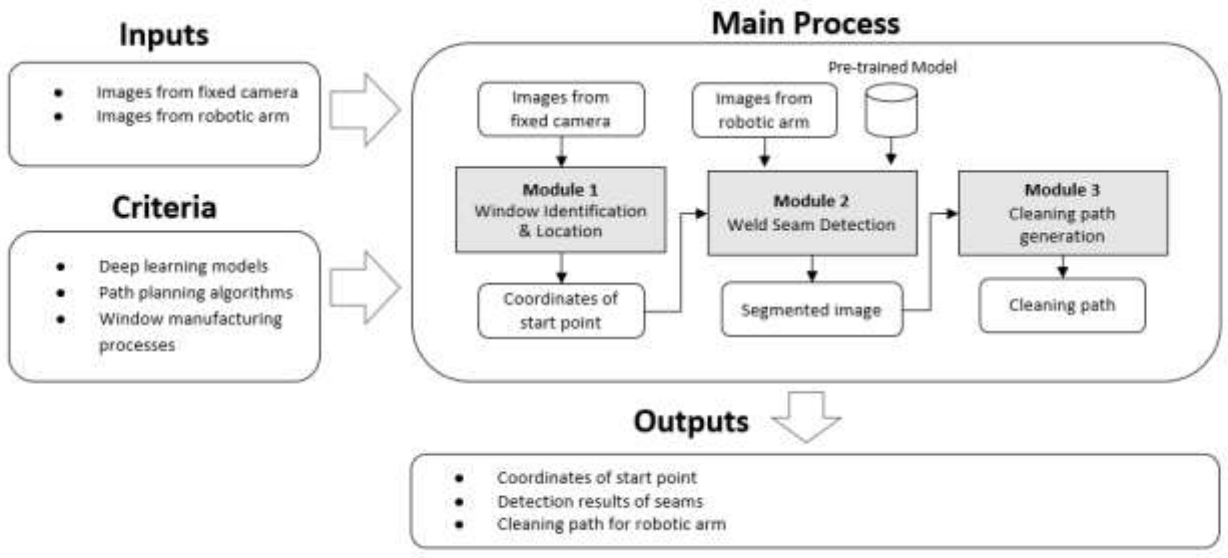

Inputs
Images from fixed camera
Images from robotic arm
Criteria
Deep learning models
Path planning algorithms
Window manufacturing processes
Main Process
Pre-trained Model
Images from fixed camera
Images from robotic arm
Module 1 Window Identification & Location
Module 2 Weld Seam Detection
Module 3 Cleaning path generation
Coordinates of start point
Segmented image
Cleaning path
Outputs
Coordinates of start point
Detection results of seams
Cleaning path for robotic arm

Figure 3. Proposed vision-based window corner-cleaning framework.

The framework's process model and notation showing the process of one working cycle, as well as the interactions among the three main elements—the fixed camera, the robotic arm, and the control system—are illustrated in Figure 4. The working cycle begins when a task is assigned, at which point an order is placed in the framework. The fixed camera is activated and captures an image for the purpose of locating the corner of the window frame. Next, the control system detects the location of the window frame, calculates the coordinates of the edge points, and sends this information to the robotic arm to serve as the start-point for the robot's path. Once the robotic arm has moved to the location above the start-point, the camera attached to the robot begins capturing images and the system detects whether there is any seam in the image based on the pre-trained image seam-detection model. If no seam is detected, the order will be completed, whereas, if a seam is detected, a cleaning path will be generated and sent to the robotic arm. The robotic arm then executes the cleaning task and returns to the start-point to repeat the cycle. The cycle keeps repeating until there is no seam captured by the moving camera.

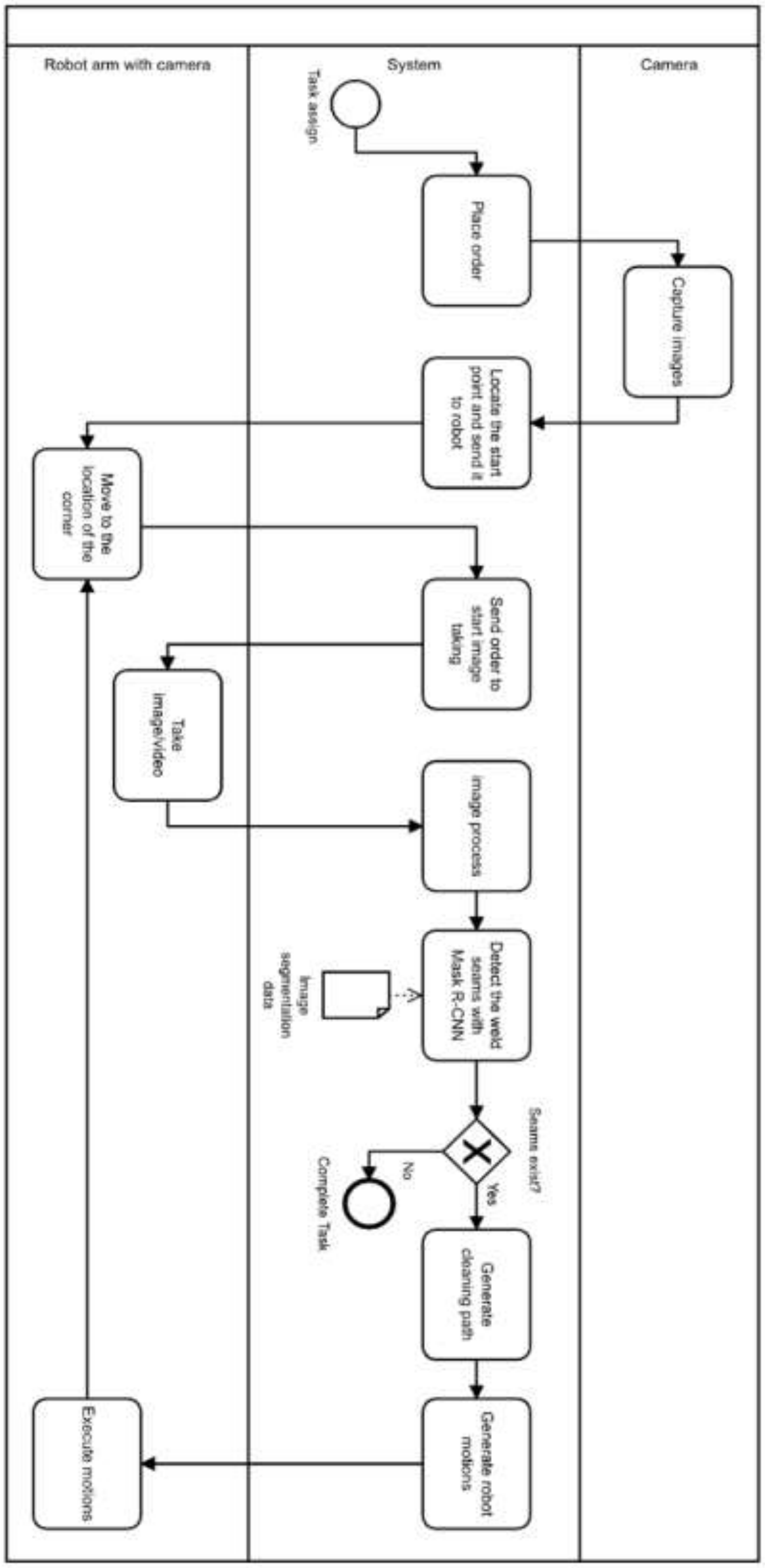


Figure 4. Proposed vision-based window corner-cleaning process model and notation.

4.2. Module 1: Window location and identification

In this module, images from the fixed camera are processed in order to obtain the current location and orientation of the window. This information supports an initial approximation of the robot towards its target. Considering that the vision range of the camera attached to the robotic arm is highly dependent on the position of the robot, a fixed camera overlooking the corner-cleaning workstation is used to initially guide the robot towards the target and perform a controlled approach towards the weld seam. Because two window profiles are forced together to produce the weld seam, the only possible location for the generated weld seam on the window frame is along the intersection of the two profiles. The illustration is shown in Figure 5.

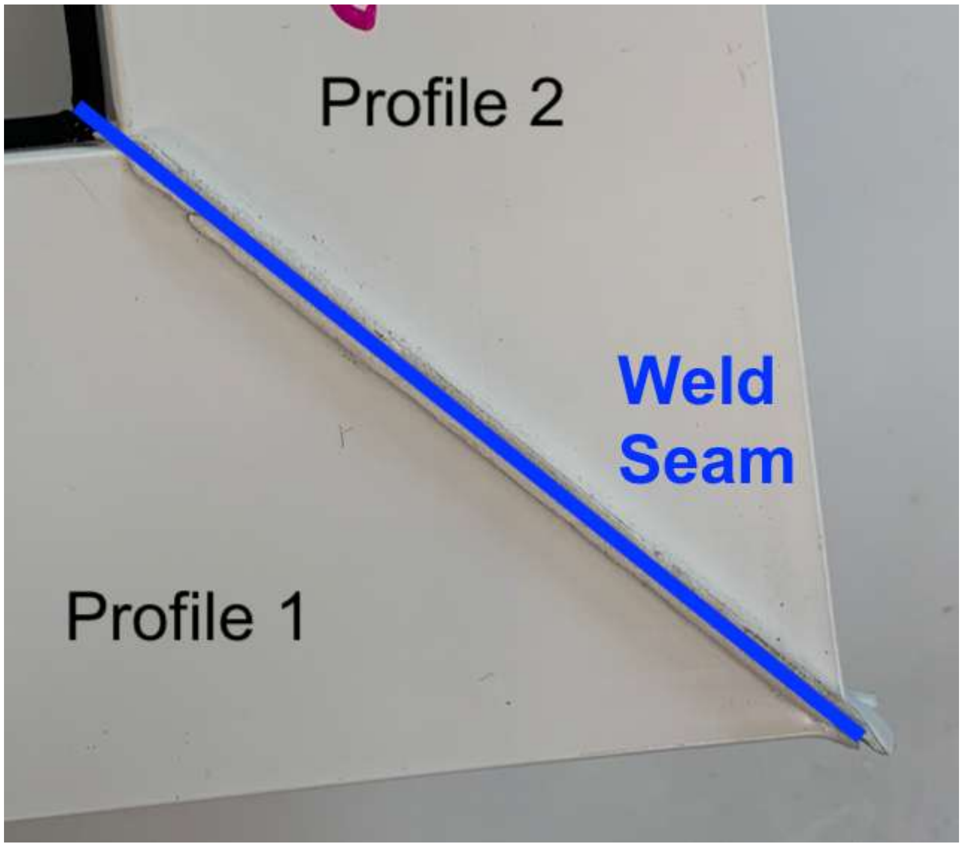


Figure 5. Window profiles and weld seam.

To be more specific, the only possible seam location is along the intersecting line between the edge points on the corner of the window, and with an identical inclined angle between both edges of the window (assuming an orthogonal window box, which is the most common design). A summary of the operations performed in this module is shown in Figure 6.

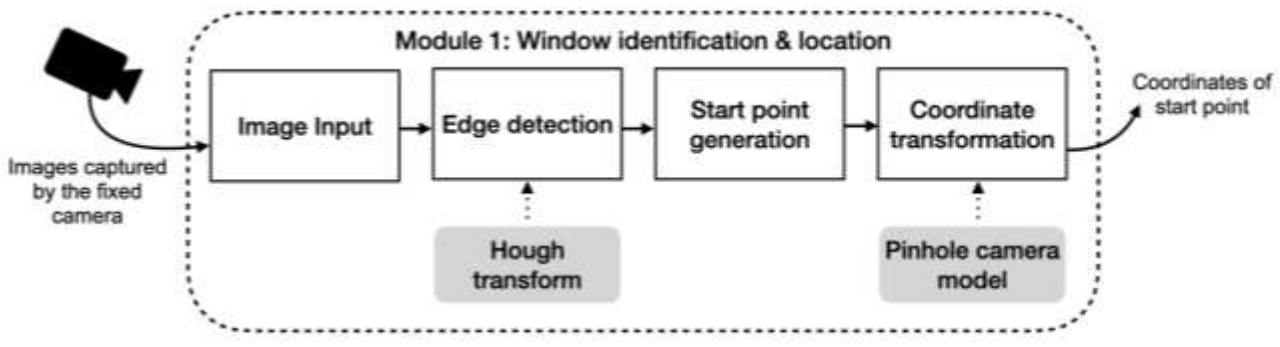

Module 1: Window identification & location
Images captured by the fixed camera
Image Input
Edge detection
Start point generation
Coordinate transformation
Coordinates of start point
Hough transform
Pinhole camera model

Figure 6. Proposed window identification and location system.

To detect the window edge coordinates, the same principle underlying the vision-based inspection method proposed by Martinez et al. (2020) is followed. In terms of hardware, the fixed camera is connected by USB cable the desktop running the python script, while Basler Pylon camera software is used for running the image capture. The images taken are stored in a folder that is then accessed by module 1 in order to initiate the subsequent tasks. The Hough transform, it should be noted, is a feature extraction technique widely used in image analysis to detect lines, among other geometrical shapes. It was introduced to detect complex patterns of points in binary image data [36]. In the present research, this technique is applied on the image captured by the fixed camera to obtain the parametric lines that are recognized as the edge lines in accordance with Equation (1):

$$\boldsymbol{L(\rho,\theta)} \coloneqq \begin{cases} \boldsymbol{\rho} = \sqrt{\boldsymbol{x^2+y^2}} \\ \boldsymbol{\theta} = \mathbf{tan^{-1}}(\frac{\boldsymbol{y}}{\boldsymbol{x}}) \end{cases} \quad (1)$$

where $L\ (\rho, \theta)$is the set of lines defined by the Hough transform, and (x, y) are the coordinates of the points of each edge as defined based on the image. Sample image results for edge detection using the Hough transform technique are presented in Figure 7. As can be seen, three main orientations of these lines, namely, the edges of both individual frame elements and the weld seam, are consistently detected. Thus, the detected lines can be used to generate rough start-points by calculating their points of intersection in the next step.

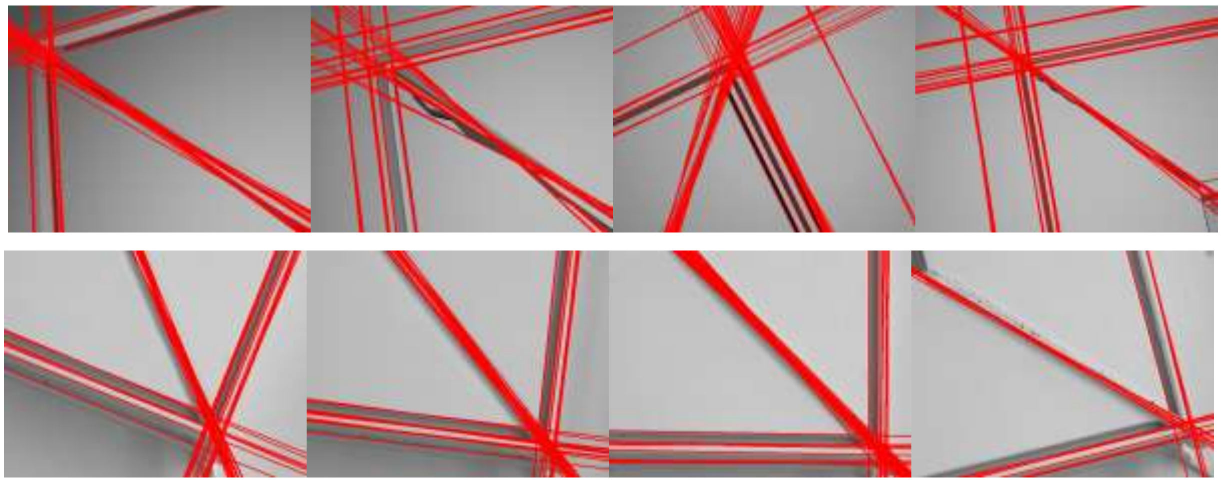


Figure 7. Examples of edge detection results in a window corner. *Top*: image samples from a virtual environment. *Bottom*: image samples from real environment.

The lines detected are categorized into three clusters based on slope: (1) the edges from the vertical frame element; (2) the weld seam; and (3) the edges from the horizontal frame element. This is done by using *k*-means clustering around the value of (θ_{cj}) of each line. The line clusters identified are illustrated in Figure 8. To simplify the calculation of the intersection point, each cluster is integrated into an average line, $C(\rho_c, \theta_c)$, to represent the cluster. The term $C(\rho_c, \theta_c)$, is defined in Equation (2) below:

$$\boldsymbol{C_j(\rho_{cj}, \theta_{cj}) \coloneqq \begin{cases} \rho_{cj} = \frac{1}{n}\sum_{i=0}^{n} \rho_i \\ \theta_{cj} = \frac{1}{n}\sum_{i=0}^{n} \theta_i \end{cases}} \tag{2}$$

where (C_1), (C_2), and (C_3) are the average lines from each cluster, satisfying $\theta_{c1} < \theta_{c2} < \theta_{c3}$. Next, considering that the three lines cannot intersect at any one point, three intersection points (P_i) of any two of the three lines are calculated as defined in Equation (3), where the average of the three intersection points calculated serves as the final start-point.

$$\boldsymbol{P} = \frac{\begin{vmatrix} \begin{vmatrix} \boldsymbol{a_1} & \boldsymbol{a_2} \\ \boldsymbol{b_1} & \boldsymbol{b_2} \end{vmatrix} & \boldsymbol{a_1 - a_2} \\ & \boldsymbol{b_1 - b_2} \end{vmatrix}}{|\boldsymbol{a_1 - a_2} \quad \boldsymbol{b_1 - b_2}|} \tag{3}$$

where any two lines of different clusters are defined by two sets of points, $(a_1,\ a_2)$ and $(b_1,\ b_2)$, and $a_1,\ a_2,\ b_1$ and b_2 are the endpoints of a line.

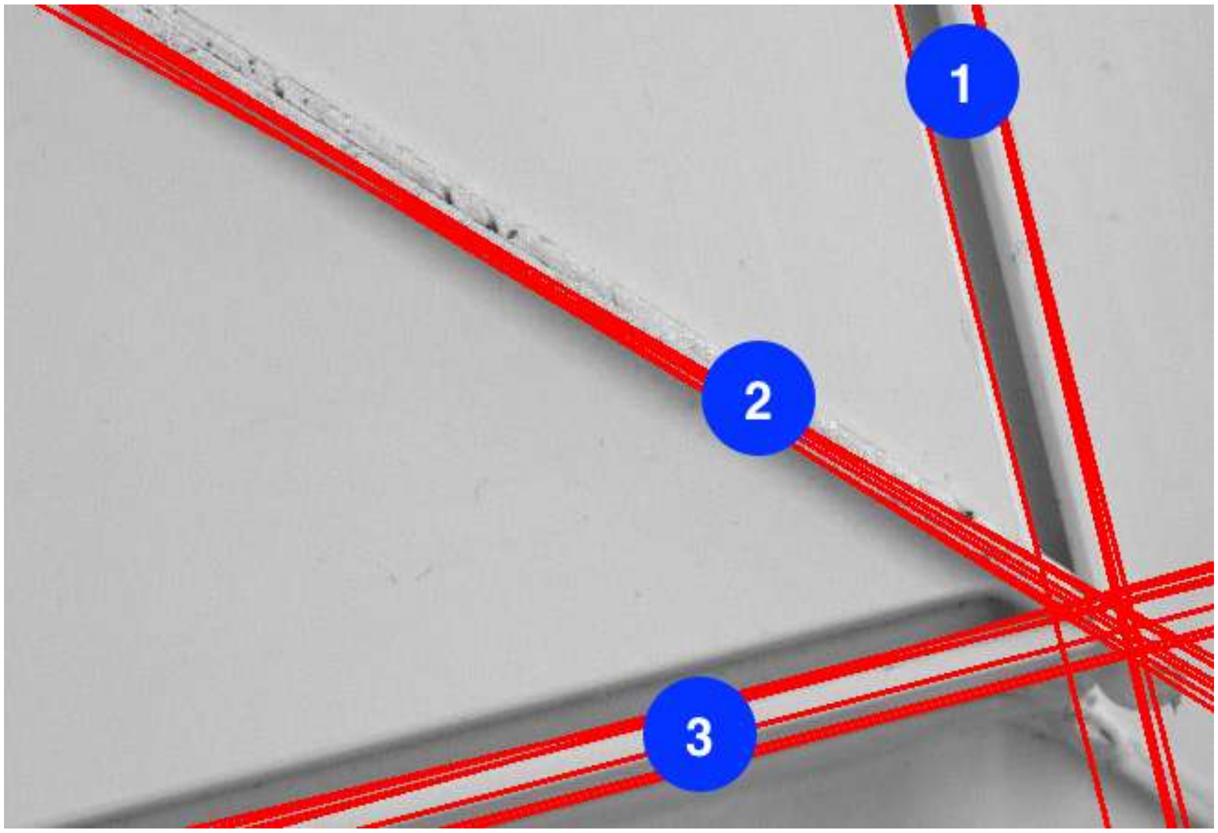


Figure 8. Clusters of detected lines based on the Hough transform angle.

The start-point on the image having been defined, the pinhole model is used to transform the coordinates from the 2D image into 3D space in accordance with Equation (3) and as illustrated in Figure 9 below.

$$\boldsymbol{Z} = \boldsymbol{d}\,\frac{\boldsymbol{f}}{\boldsymbol{l}} = \boldsymbol{d}\,\frac{\boldsymbol{f}}{\sqrt{\boldsymbol{f}^2 + \boldsymbol{x}^2 + \boldsymbol{x}^2}} \qquad (3)$$

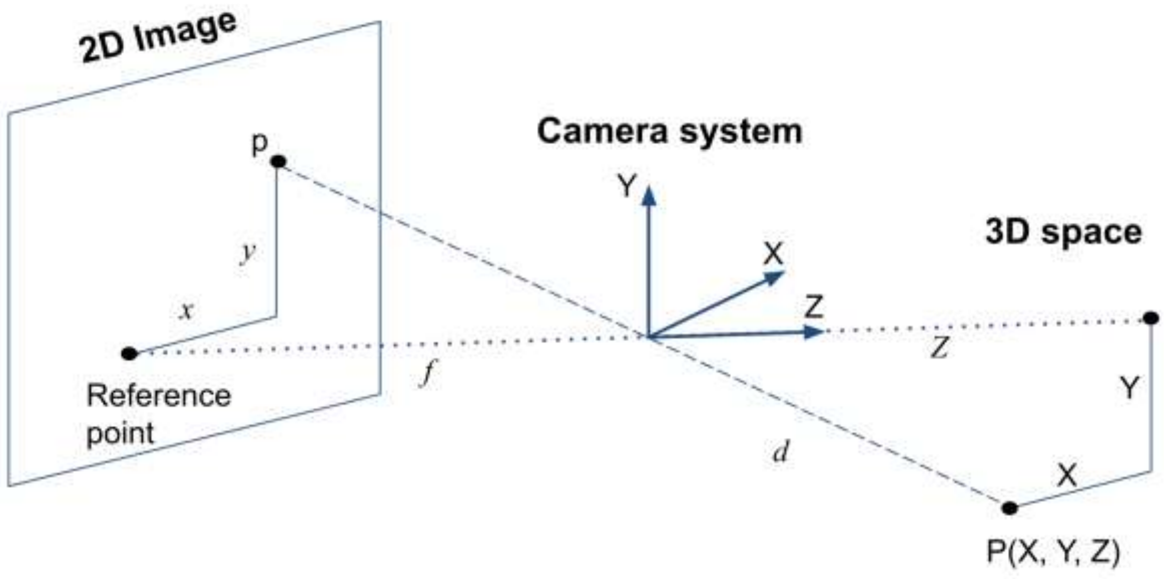


Figure 9. Pinhole camera model illustration.

where (*Z*) is the distance between the reference point and the camera, (*d*) is the distance between the start-point and the camera, (*l*) is the distance between the camera and the reference point projected on the image, and (*f*) represents the focal length of the camera. The height of the start-point is fixed in order to keep the robotic arm at an adequate distance to observe the target. The setup is shown in Figure 10, where (r_0) and (r) are the reference points in the 3D space and (s_0) and (s) are the start-points on the 2D image.

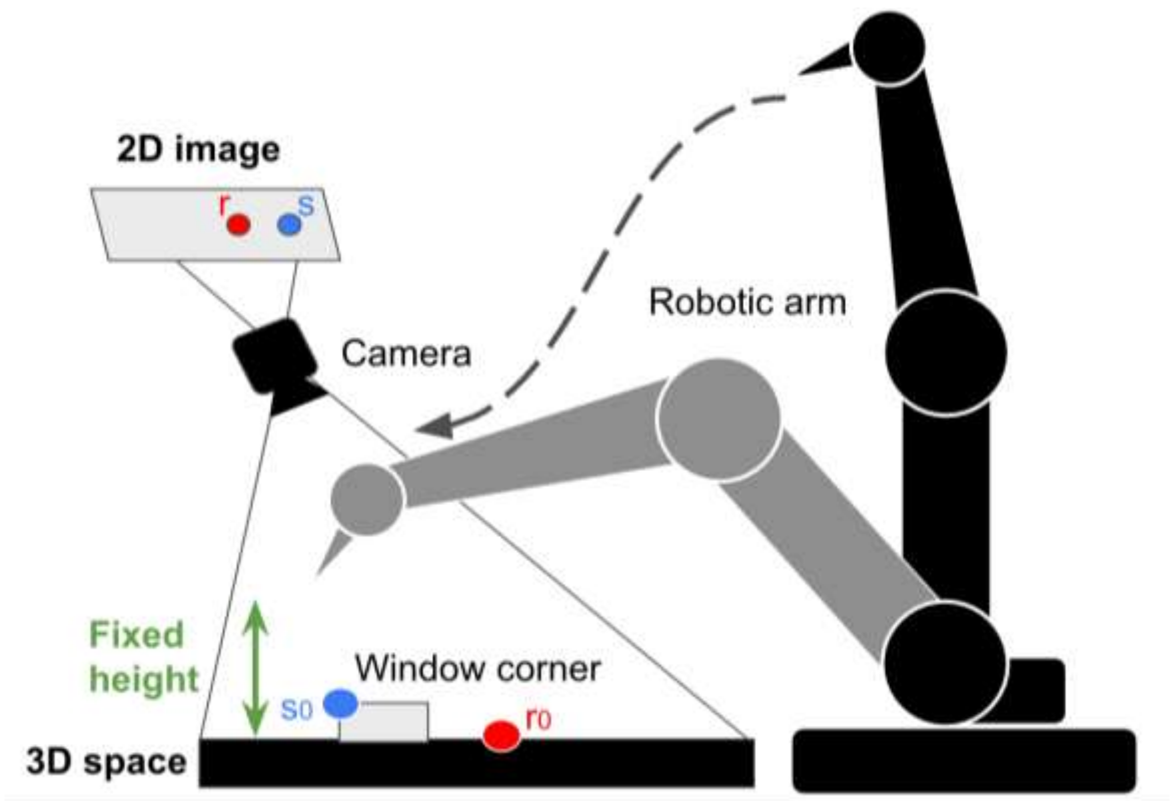


Figure 10. Application of the pinhole model in the workstation under investigation.

4.3. Module 2: Weld seam detection

In this module, the goal is to determine the area of weld seam to be cleaned in order to guide the corner-cleaning process accordingly. A summary of the operations performed in this module is shown in Figure 11.

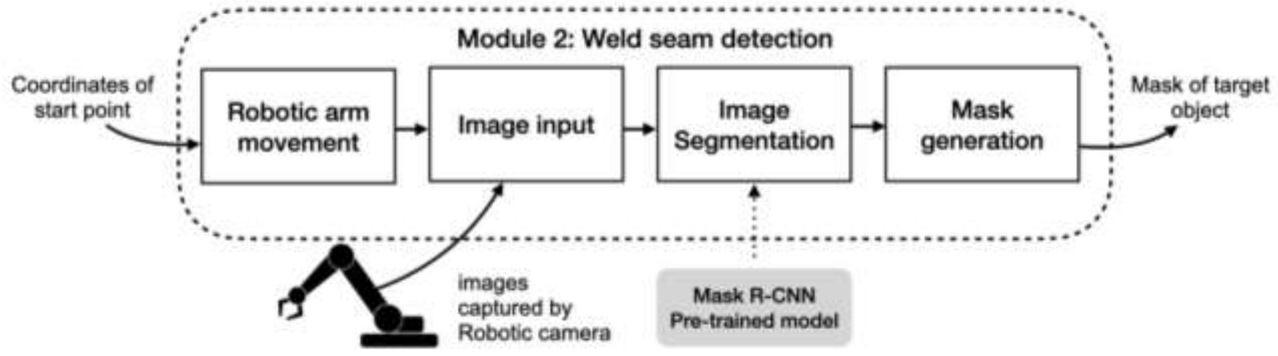
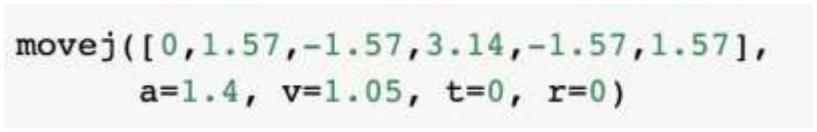

Module 2: Weld seam detection
Coordinates of start point
Robotic arm movement
Image input
Image Segmentation
Mask generation
Mask of target object
images captured by Robotic camera
Mask R-CNN Pre-trained model

Figure 11. Proposed weld seam detection system.

The coordinates that are the final output of module 1 need to be transformed into a script file in order for the robotic arm to perform a motion. The transformation script is generated using RoboDK motion planning. Example motion script can be found in Figure 12.



```
movej([0,1.57,-1.57,3.14,-1.57,1.57],
      a=1.4, v=1.05, t=0, r=0)
```

Figure 12. Example script for robot motion

where moveJ represents a point-to-point motion, which is a most used command in a robot script. A typical format of a moveJ command is: $movej(q, a = 1.4, v = 1.05, t = 0, r = 0)$, where q represents joint positions, a is the joint acceleration of the leading axis, v is the joint speed of the leading axis, t is time, and r is the blend radius [37].

Weld seams are not generated as any particular geometrical shape, due to the nature of the process where force is applied to the thermoplastic material from both sides in a non-uniform manner. This makes it difficult to locate the exact area of seams to be cleaned, as simple feature recognition image-processing techniques are not directly applicable to such situations. However, image segmentation techniques can be used to define objects and boundaries by separating the image into several segments. For the purpose of segmenting the weld seam and accurately determining the boundaries of the area to be cleaned, a Mask R-CNN network is created. An example of the desired detection result and the Mask R-CNN architecture are shown in Figure 13.

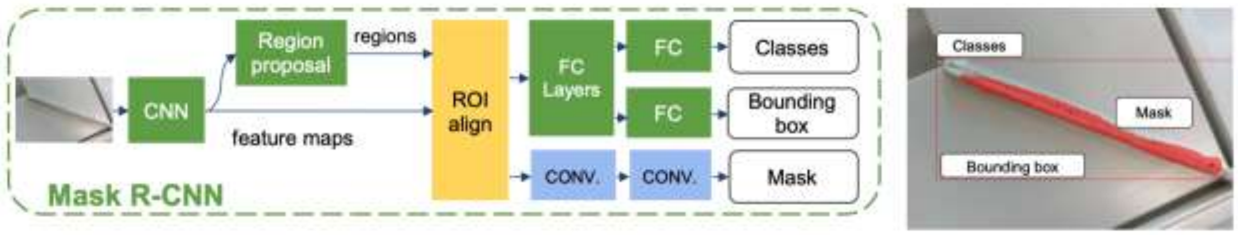


Figure 13. Image segmentation using Mask R-CNN.

To train a weld seam detection model that can identify the boundaries of the area to be cleaned correctly, several steps are required. Step (1): images of uncleaned corners of window frames are collected as training material. Step (2): to teach the model to recognize the weld seam, each image is manually labelled, and the labelled images are then divided into two datasets—a training dataset and a validation dataset—at a proportion of 3:2. Step (3): the model is trained using the training dataset, and then tested using the validation dataset. These steps are described in greater detail as follows. First, the selected model is trained using a pre-labelled dataset containing 817 images in total, 396 of them being images of weld seams in actual windows, and 421 of them being representations of weld seams from modelling software. The image dataset is constructed using images obtained using a Robotiq wrist camera affixed to the robot arm. All images in the database are manually labelled and segmented. The software used, it should be noted, is VGG Image Annotator (VIA), which is an online manual annotation software developed by the Visual Geometry Group at the University of Oxford [38]. A few examples of images from the training database are shown in Figure 14.

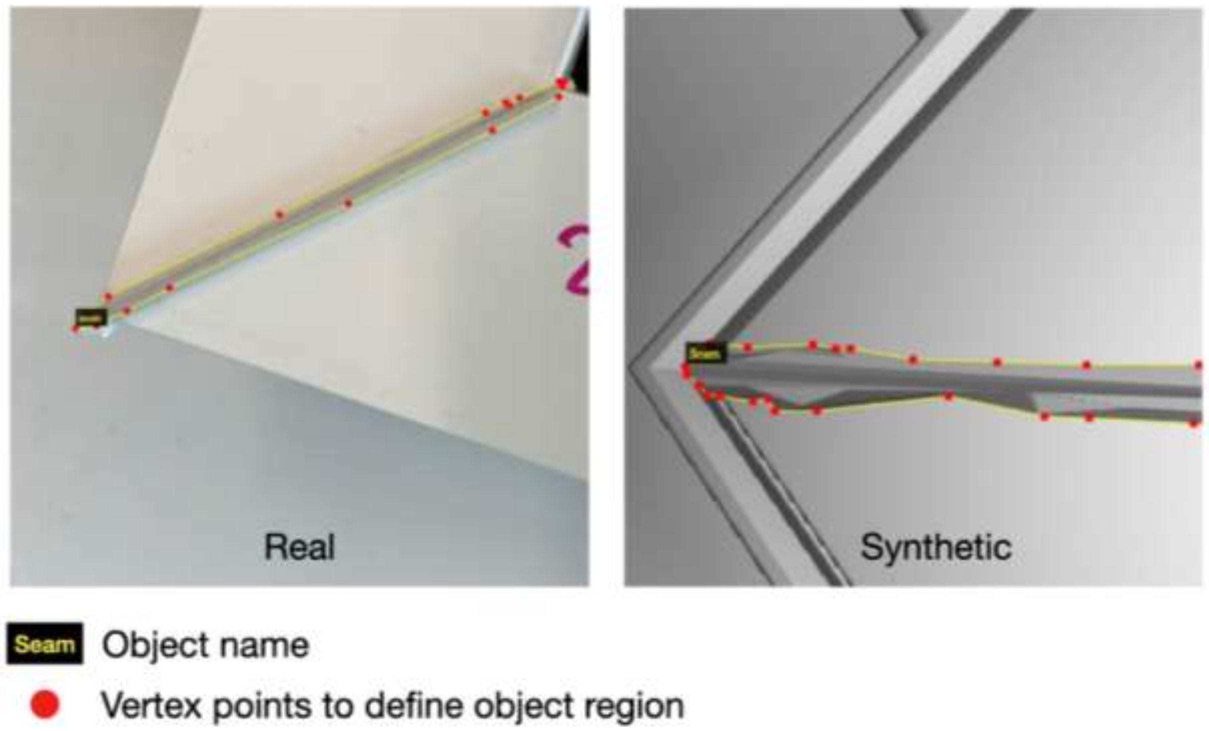
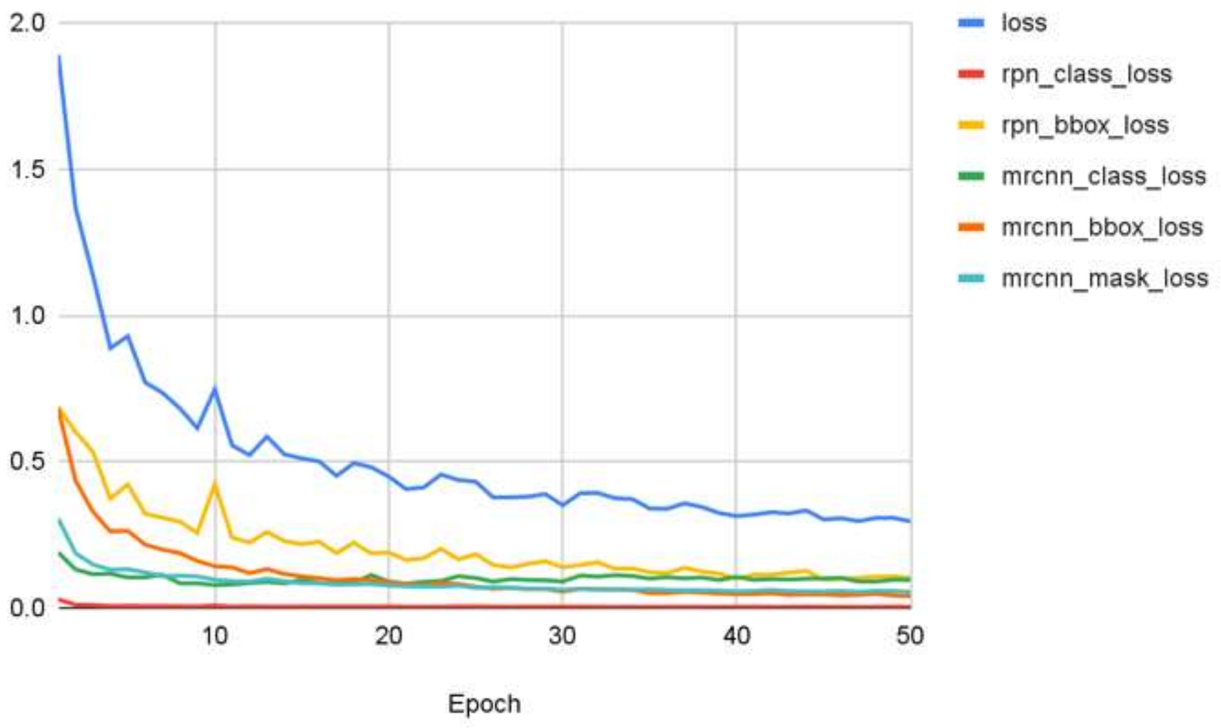


Figure 14. Sample images with labels from the training dataset.

The model is trained within Google Colaboratory with GPU boosting and 12 GB of RAM, with Tensorflow (version 1.15.0) and Keras (version 2.1.5) installed. The COCO dataset, which is a large-scale object detection, segmentation, and captioning dataset, is used as the initial training weights on the pre-trained R-CNN model. The training process takes 3.5 hr to complete 50 epochs of model training.

The model has a final training loss of 0.2959 and a final validation loss of 1.0228, as shown in Figure 15. As expected, the model's loss (error) gradually decreases as the number of training epochs (runs) increases. In the figure, "loss" indicates the overall model loss during the training and validation steps; the other indices in the figure, i.e., "rpn_class_loss" and "rpn_bbox_loss", represent the region proposal losses for the classification and localization (bounding box); and "mrcnn_class_loss", "mrcnn_bbox_loss", and "mrcnn_mask_loss" represent the classification, localization (bounding box), and segmentation (mask) losses during the Mask R-CNN training and validation steps. It is important to note that, in this particular application of this neural network, it is paramount that the final value for “mrcnn_mask_loss” is as small as possible. With the dataset used, the final segmentation loss achieved is found to be 0.0542 for the training and 0.0976 for the validation.



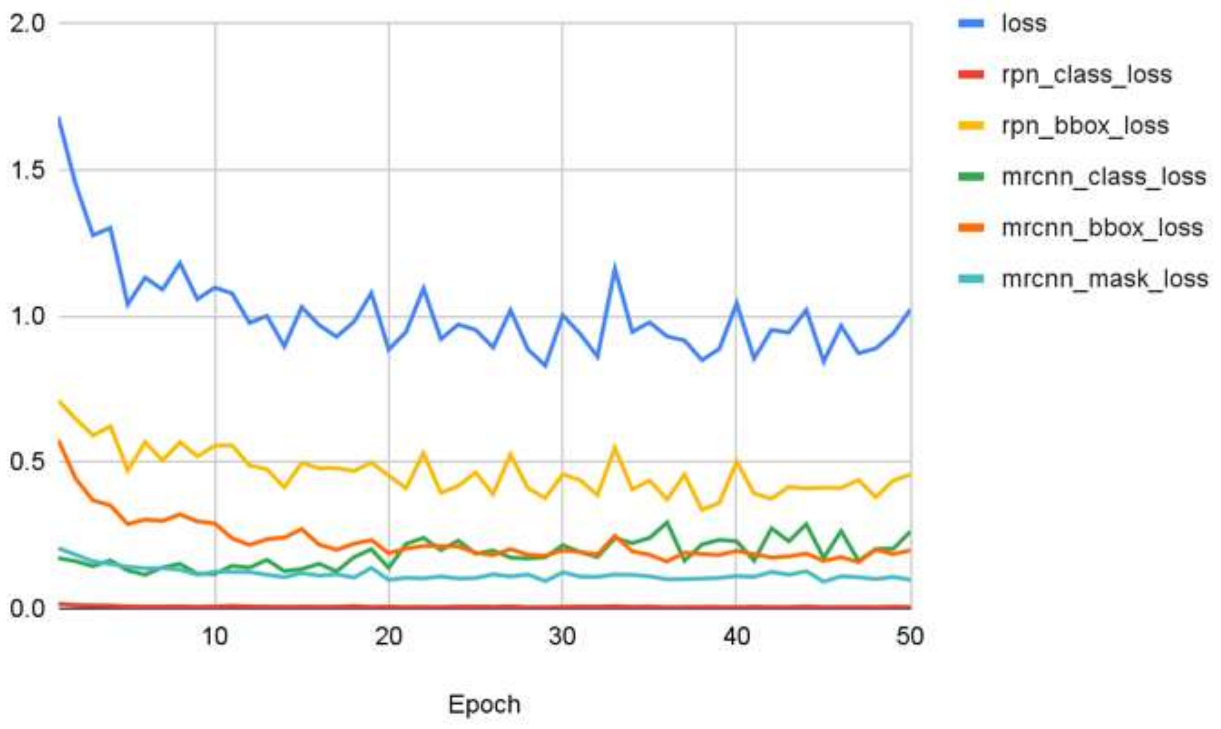


Figure 15. Model training results. *Top:* training data. *Bottom*: validation data.

4.4. Module 3: Cleaning path generation

In this module, the cleaning path is generated based on the object mask obtained in the preceding module. The overall module procedure is illustrated in Figure 16. The weld seams are typically of a strip-like shape due to the manner in which the frame elements are forced together during the weld operation, so the cleaning path is an almost linear path between the start- and end-points of the seam. Since the width of the seam is usually approximately 2 mm to 5 mm, the most efficient cleaning method is to go travel along the centerline of the seam with a tool that is sufficiently wide to span the full width of the seam, similar to the current practice for automated corner-cleaning procedure. Accordingly, the objective of this module is to estimate the centerline of the weld seam. Given that weld seams are irregular in shape due to non-uniform forces applied during the hot-plate welding process, the path generated must be able to adapt to the weld's shape.

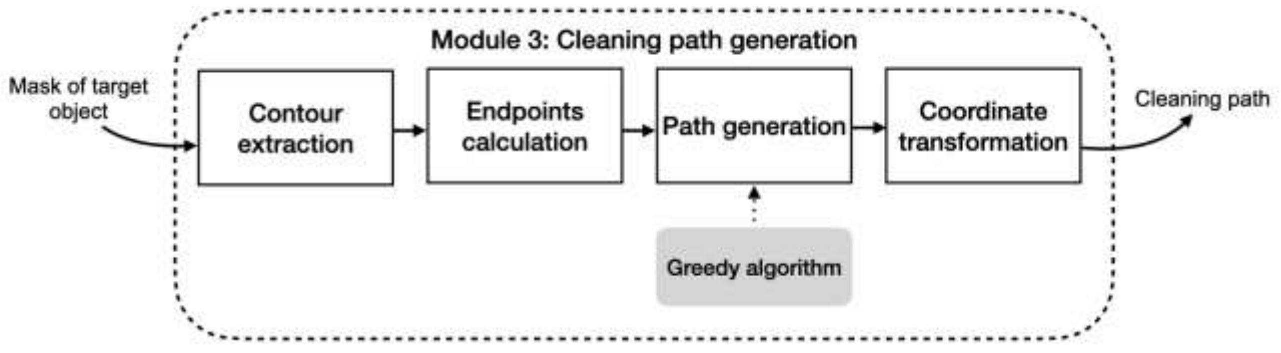


Figure 16. Proposed cleaning path generation system.

The output of the preceding module having been a strip-like mask that defines the seam area, the first step in this module is to extract a contour from the generated mask. This contour defines, using as many points as necessary, the mask obtained by the Mask R-CNN model. Each contour point is stored into a list and assigned an index according to its location on the contour. Due to the strip-like shape of the area, by comparing the distances between any two points on the contour, the two points with the longest distance are chosen as start- and end-points for the robot path. Both end-points satisfy Equation (3):

$$(\boldsymbol{E_1}, \boldsymbol{E_2}) = \{(\boldsymbol{p}, \boldsymbol{q}) | \boldsymbol{max}(\boldsymbol{d}(\boldsymbol{p}, \boldsymbol{q})\} \boldsymbol{where}\ \boldsymbol{d}(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{(\boldsymbol{qx} - \boldsymbol{px})^2 + (\boldsymbol{qy} - \boldsymbol{py})^2}\} \quad (3)$$

where (E_1) and (E_2) represent the start- and end-points for the robot path and $d(p, q)$ is the Euclidean distance between two cartesian points $p(px, py)$ and $q(qx, qy)$. From the remaining contour points, the robot path is determined using the Greedy algorithm. The Greedy algorithm, it should be noted, is typically applied when obtaining an optimal result is not feasible in a single solution. It divides the optimization problem—in this case, a path planning problem—into several minimization problems in order to yield locally optimal solutions at every iteration and eventually reach an optimal solution. Thus, the cleaning path generation is calculated by dividing the whole area into several sections and gathering the local center points from each section to define a final path. As shown in Figure 16, (E_1) and (E_2) represent the start- and end-points, respectively, on the contour. To ensure efficient weld cleaning, the path should track along the center of the seam, so the best local solution for the division, where the contour is divided into many parts by each pair of counterpart contour points, is determined as the midpoint of each two counterpart points. All (P_i) points are defined while extracting the contour. Thus, if (E_1) is the start-point of the path, the second point for the robot path is defined as the midpoint, (CP_1), of the next two points, represented by (P_1) and (P_1') in Figure 17. Consequently, the third point, (CP_2), is the midpoint of (P_2) and (P_2'), and so on. The set of points generating the robot path, (CP_i), is defined by Equation (4).

$$\boldsymbol{CP_i(x_{ci}, y_{ci})} = \left(\frac{X_i+x_i'}{2}, \frac{y_i+y_i'}{2}\right) \tag{4}$$

where P_i (x_i, y_i) and $P_i{}'$ (x_i', y_i') are the corresponding points on the contour. Given that every point in the contour list is assigned an index, if we let the index of (E_1) be j, then the indices of (P_i) and $(P_i{}')$ will be j+1 and j−1. Taking the figure below as an example, (E_1) in the figure is (P_{45}) in the contour list, and the corresponding points for calculating (CP_1) are (P_{44}) and (P_{46}). A step-by-step overview of the path generation results is provided in Figure 18.

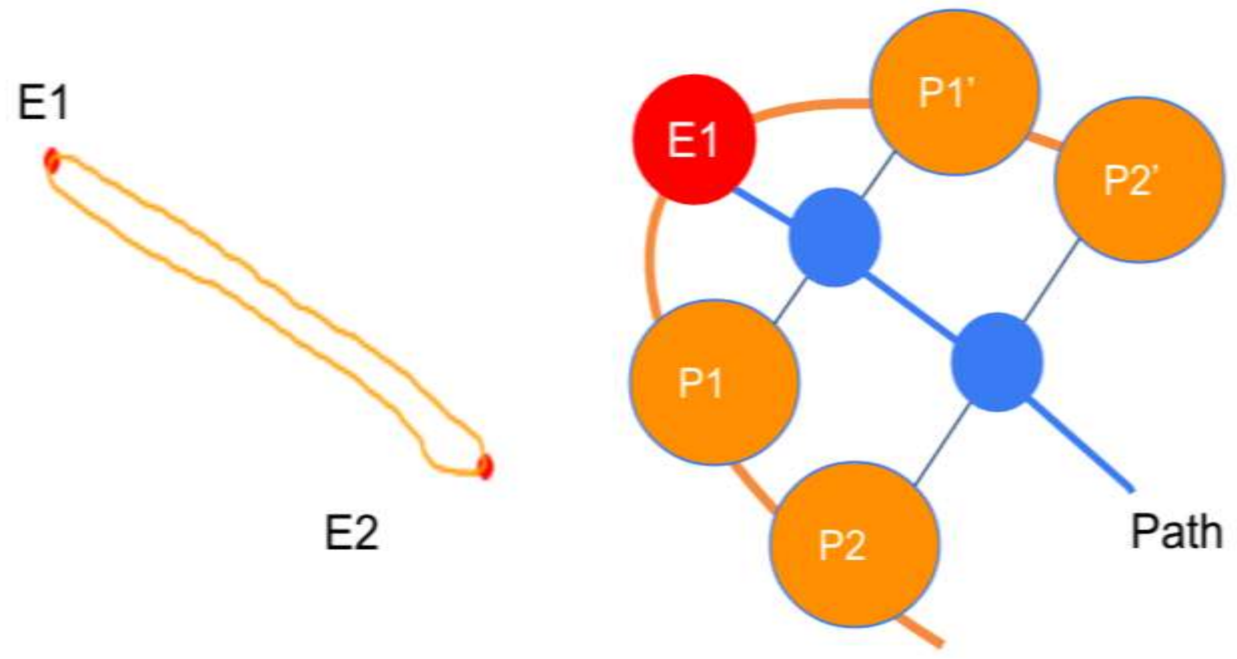
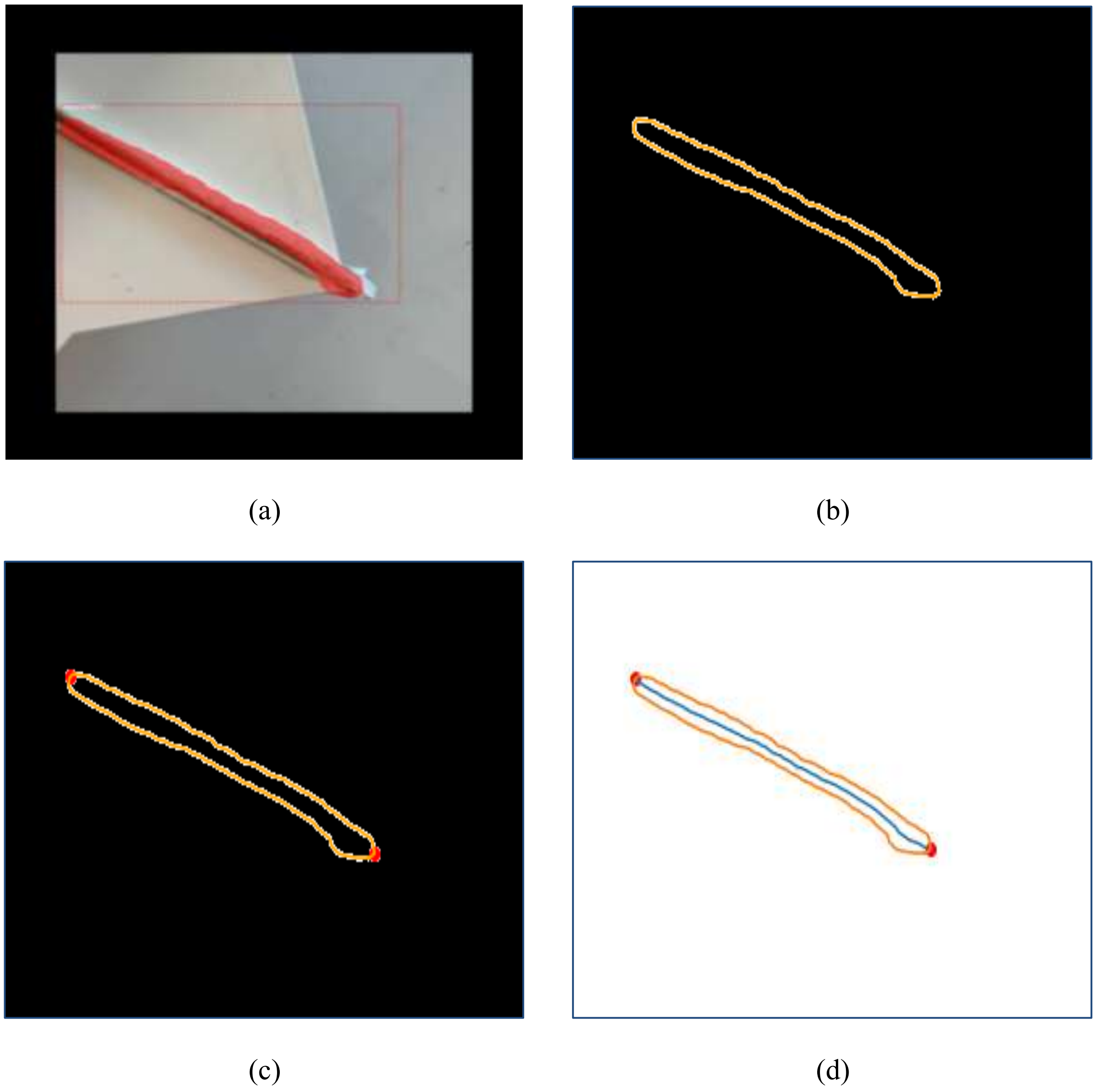


Figure 17. Application of greedy algorithm in path planning.



(a) (b) (c) (d)

Figure 18. Steps to generate the corner-cleaning path alongside the detected weld seam:(a) Mask R-CNN mask generated; (b) contour extracted from the mask; (c) estimated start- and end-points for the corner-cleaning operation; (d) cleaning path obtained using the Greedy algorithm.

Chapter 5 Results

This chapter presents the validation of the proposed vision-based robotics framework in two different environments: a simulation environment and the real-world environment. In the following sections, the experimental setup is first defined, then the results and analysis are presented, followed by a discussion of the limitations of the study.

5.1. Environmental setup

For the validation of the proposed approach to corner cleaning in window manufacturing, two cameras are used—a Basler ACE camera with an Optron 35 mm lens monitoring the working area, and a Robotiq wrist camera attached to a Universal Robots UR5e. The virtual environment is built within the RoboDK platform, providing an accurate representation of the real environment. In the robotic reference frame, the origin is defined as the center base of the robot, so the coordinates of the camera position are (500, −500, 600), where the unit is mm. Both experimental setups in their environments are shown in Figure 19.

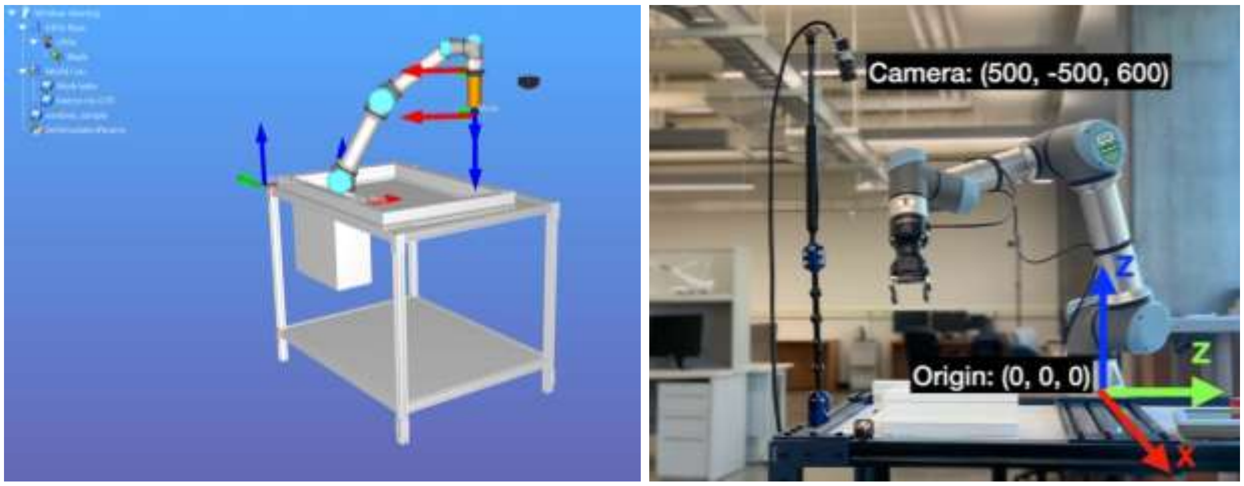


Figure 19. Validation environments: Vision-guided robot system. *Left:* virtual environment in RoboDK software. *Right:* Real world environment.

The window frames used in the real environment are made of polyvinyl chloride (PVC), and their geometry is modelled in a CAD software before being transferred to the virtual environment in STL format. The height of the selected window profiles is 3.260 $\pm$ 0.15 inches (82.804 $\pm$ 3.81 mm) and the width is 4.31 $\pm$ 0.20 inches (109.474 $\pm$ 5.08 mm). Examples of the windows used in both environments are shown in Figure 20.

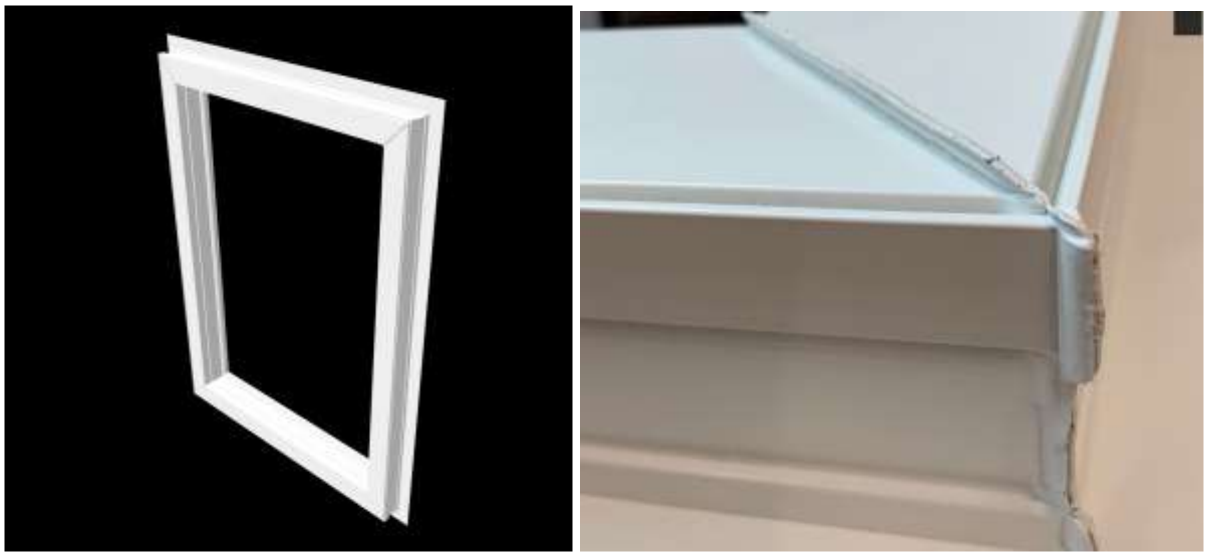


Figure 20. Window samples used in this study. *Left:* STL model used in the virtual environment. *Right:* PVC window used in the real-world environment.

The wrist camera attached to the UR5e robot arm has a field of view (FOV) ranging from 100 mm × 75 mm to 640 mm × 480 mm, and the focus ranges from 70 mm to infinity. Based on the specifications and the height of the window under investigation in this study, a required minimum FOV to fully cover the corner area is around 110 mm × 110 mm. However, considering that the placement angle of the window frame may vary, the narrowest FOV should not be less than 156 mm, which is the maximum length of the seam (as shown in Figure 21). If the shorter width is approximately 156 mm, then the working distance from the camera should be approximately 146 mm above the object (see Figure 21). The height of the window frame (83 mm) must be taken into consideration as well. Therefore, the vertical distance from the camera to the working table should be not less than 229 mm, which equals the height of the window plus the working distance. Finally, to reserve a sufficient amount of space in which for the end-effector (cleaning tool) to operate, the fixed distance along the *z*-axis is ultimately set to 300 mm. Since the camera distance to the workstation is fixed, as is the height of the robotic arm at the initial position, the height measurements (*z*-axis) are kept as a constant true value, rather than being estimated, in order to simplify the system's operations.

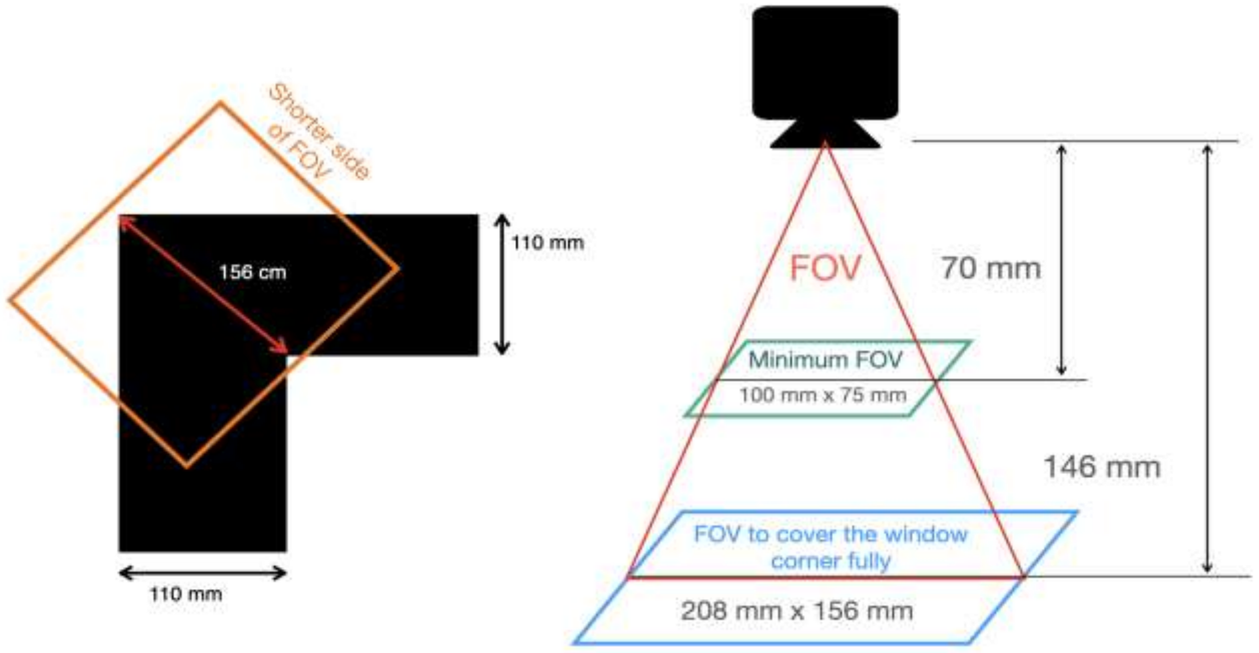


Figure 21. Camera placement considerations. *Left*: Illustration of the decision for the minimum width of field of view. *Right*: Illustration of the calculation for the minimum working distance needed to completely cover the window corner area.

5.2. Validated results

All three modules the proposed framework comprises are engaged during the process in both the virtual and real environments. The results of these experiments are analyzed in order to validate the framework performance.

5.2.1. Module 1 validation

The first module, window identification and location, locates the rough position of the start-point to enable the robot to perform an initial approach towards the corner of the window frame. To validate the coordinate estimation process, several window corners, are placed one-by-one at random positions within the working area. The system is then run to obtain the location estimate. The actual location is measured as well using a manual tape measure accurate to within 1 mm. The error is calculated following Equation (5), where (x_A), (y_A) are the coordinates along the x- and y-axes of the actual position of the start-point, and (x_E), (y_E) are the coordinates along the x- and y-axes of the estimated position of the start-point. The results for locating the window corner area are presented in Table 1.

$$\boldsymbol{Error} = \sqrt{(\boldsymbol{x_A} - \boldsymbol{x_E})^2 + (\boldsymbol{y_A} - \boldsymbol{y_E})^2} \tag{5}$$

Table 1. List of results obtained from Module 1.

	Coordinates of start-point (x, y)									
	Virtual environment					**Real environment**				
#	**Estimated**		**Actual**		**Error**	**Estimated**		**Actual**		**Error**
	x[mm]	**y[mm]**	**x[mm]**	**y[mm]**	**[mm]**	**x[mm]**	**y[mm]**	**x[mm]**	**y[mm]**	**[mm]**
1	494	−497	496	−499	2.828	486	−510	489	−506	5.000
2	499	−490	501	−503	13.153	470	−488	467	−479	9.487
3	488	−504	492	−510	7.211	478	−502	481	−496	6.708
4	512	−508	510	−516	8.246	478.0	−497	475	−499	3.606
5	502	−509	507	−501	9.434	472	−500	471	−502	2.236
6	469	−405	476	−401	8.062	496	−408	500	−401	8.062
7	490	−388	497	−391	7.616	512	−388	500	−390	12.166

8	482	−393	480	−405	12.166	488	−420	494	−415	7.810
9	485	−402	480	−398	6.403	496	−420	500	−412	8.944
10	483	−401	481	−405	4.472	497	−413	498	−400	13.038
11	589	−499	593	−501	4.472	602	−498	600	−500	2.828
12	586	−502	592	−508	8.486	596	−480	597	−483	3.162
13	605	−509	608	−505	5.000	604	−518	601	−515	4.242
14	601	−505	602	−503	2.236	598	−483	600	−481	2.828
15	600	−487	601	−480	7.071	601	−495	610	−496	9.055
	Average [mm]				7.123735848	**Average [mm]**				6.611634874
	Standard Deviation [mm]				2.975296215	**Standard Deviation [mm]**				3.389142118

Based on the results in Table 1, the performance of the identification algorithm can be statistically analyzed. The average measurement error (mean ± standard deviation) of the window location is 7.23 ± 3.21 mm and 6.10 ± 3.55 mm for the virtual environment and the real environment, respectively. As per Figure 21, the dimensions of the window corner are 110 mm × 110 mm and the FOV of the set height is approximately 232 mm × 309 mm. The FOV at the set height is kept at least 86 mm wider than the required FOV in order to fully cover both sides of the corner. Given these specifications, a 6 to 7 mm measurement error represents a 5% error relative to the window corner area, and this is considered acceptable considering the purpose of this particular algorithm (i.e., approximation).

5.2.2. Module 2 validation

The main function of the second module is to detect the weld seam using the pre-trained image segmentation model. The Mask R-CNN applied dataset, which consists of 78 new images from the virtual and real environments, is used for validation of the model. An example of the weld seam classification, location, and segmentation results is presented in Figure 22.

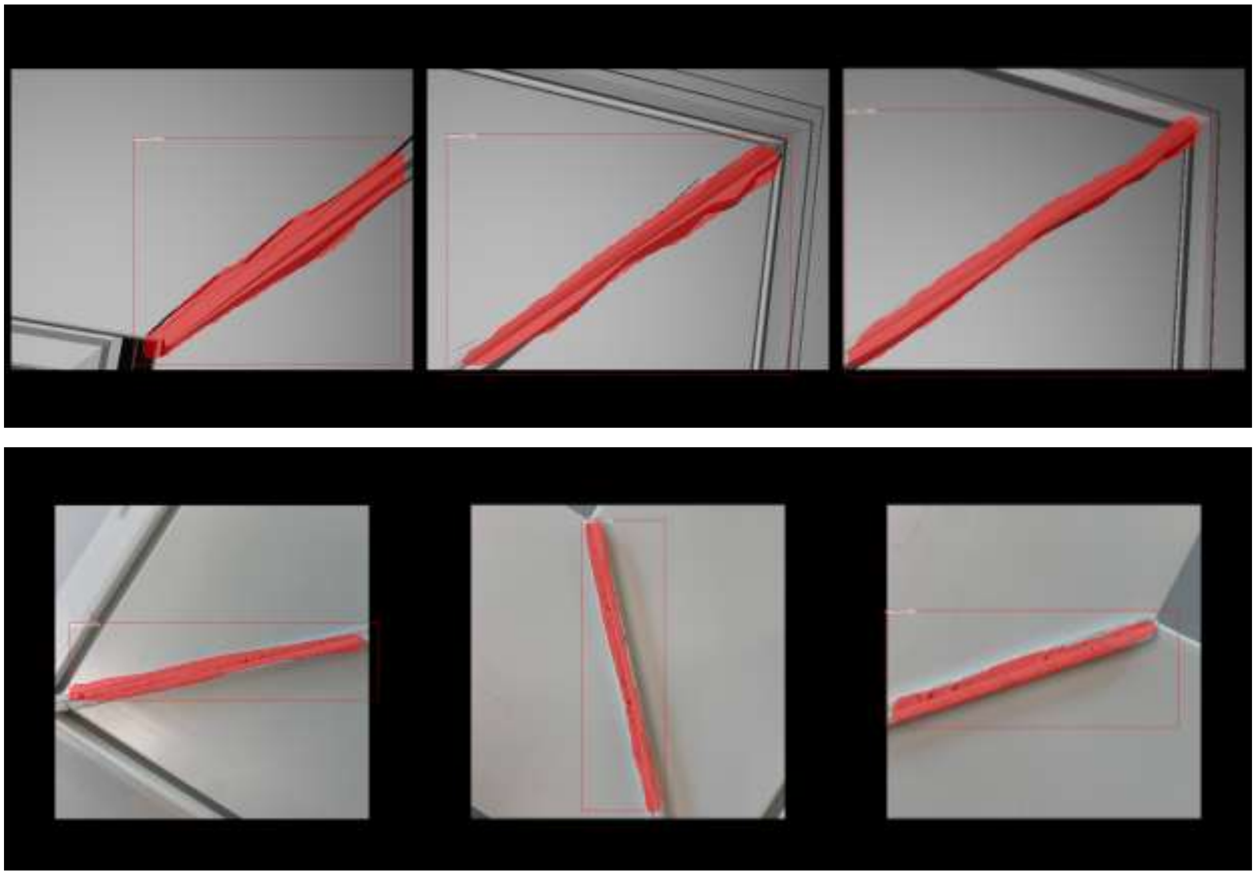


Figure 22. Results of the weld seam segmentation using the Mask R-CNN model on images of the test dataset. *Top*: Images of the simulated environment. *Bottom*: Images of the real environment.

Table 2 shows the seam-detection model's performance for the intersection over union (IoU) at various values. IoU, it should be noted, indicates the area of overlap over the area of union, as expressed in Equation (6).

$$\boldsymbol{IoU} = \frac{\boldsymbol{Area\ of\ Overlap}}{\boldsymbol{Area\ of\ Union}} \tag{6}$$

IoU is often used as a threshold to classify a prediction as either a true positive or a false positive. Given that the mask prediction accuracy is the key performance factor in this research, the IoU here is calculated by dividing the overlap of the predicted and ground-truth masks over the area of union. Additionally, the mean average precision (mAP) is commonly used to evaluate object detection models. Since, in this particular case, there is only one type of object to predict, mAP represents the average precision (AP), which is the area under the precision-recall curve. Precision, recall, AP, and mAP, defined in Equations (7) to (10), respectively, are used as the performance metrics for the Mask R-CNN model.

$$\boldsymbol{Precision} = \frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ Positives + False\ Positives}} \tag{7}$$

$$\boldsymbol{Recall} = \frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ Positives + False\ Negatives}} \tag{8}$$

$$\boldsymbol{AP@n} = \frac{\boldsymbol{1}}{\boldsymbol{total\ number\ of\ ground\ TF}} \sum_{\boldsymbol{k}}^{\boldsymbol{n}} \boldsymbol{Precision@k} \times \boldsymbol{Recall@k} \tag{9}$$

$$\boldsymbol{mAP} = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{AP_i} \tag{10}$$

Table 2. Seam-detection performance using the Mask R-CNN model with an IoU = 0.5, 0.75, 0.8 and 0.9.

IoU	mAP
0.50	0.953
0.60	0.851
0.75	0.755
0.80	0.707
0.90	0.511

The mAP results in Table 2 are the calculated results corresponding to 20 randomly chosen images from the test dataset, where the IoU is defined as falling within the range from 0 (no union between the ground truth and detection) to 1 (detection and ground truth are identical). Different IoU thresholds are applied to describe the accuracy of the model in correctly identifying the boundaries of the weld seam. As the threshold value increases, the determination of what is considered a true positive becomes stricter. For this study, IoU is tested starting at 0.5 (commonly used as the minimum acceptable IoU) until it reaches 0.9. Based on the results, the model performance is found to decrease gradually as the IoU value increases, having a mAP of 0.95 when the IoU is set at 0.5, but then decreasing to a mAP as low as 0.5 when the IoU is set at 0.9. Thus, the model performance in this study is found to be comparable with that of similar applications described in the literature [39]. As observed, the performance of the pre-trained model is found to be sufficient to proceed to the next module.

5.2.3. Module 3 validation

The final output of module 3 is a list of coordinates or target points in the 3D space representing the cleaning path for the robot arm. Aside from the accuracy of path location, which is directly related to the accuracy obtained from the image segmentation process discussed above, the evaluation of this module also encompasses the force control and the accuracy capacity of the robotic arm executing the task. The force control process is similar to position control, where each joint is adjusted to match the required output, but the force control process is more complex than position control due to the fact that both the trajectory and the exact force need to be satisfied simultaneously. Since the aim here is simply to demonstrate the preliminary motion planning of the cleaning task, only the position control is discussed, but this also constitutes a study limitation with respect to the validation of this module as no force analysis is provided. According to current industrial practice, it is assumed to be possible to follow the generated path. Nonetheless, a full validation of the task execution for the generated cleaning path can only be achieved by combining the force control with the position control as discussed above. Thus, the results here are limited to visualization results (see Figure 23). Moreover, the force control varies for different end-effectors in the execution of the robotic arm motions, and the impact of this aspect on the cleaning performance requires further research.

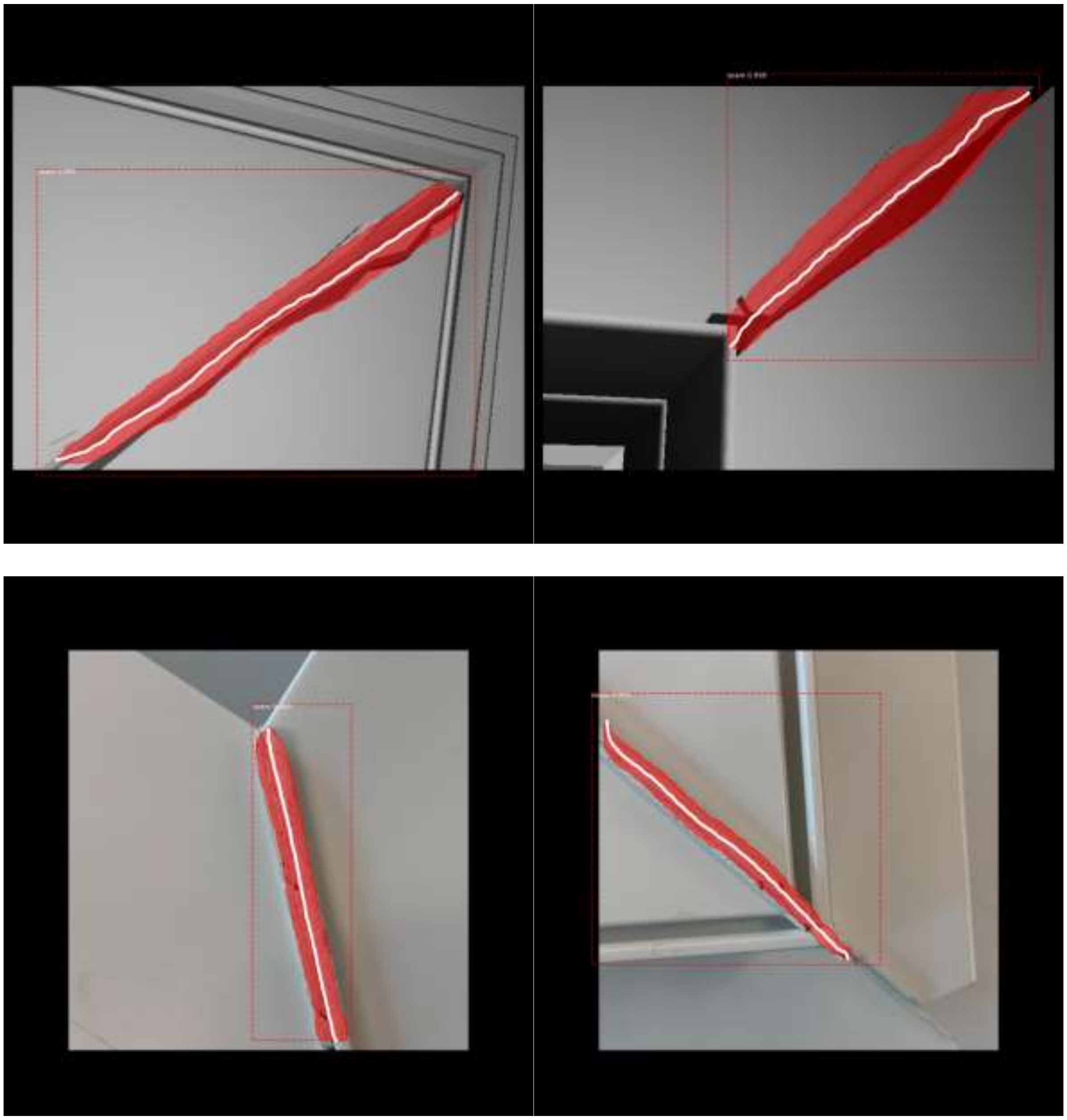


Figure 23. Results of the cleaning path planning (white lines) on a detected weld seam. *Top*: Images of the simulated environment. *Bottom*: Images of the real environment.

Chapter 6 Conclusions

6.1. Summary and conclusions

Hot-plate welding is a commonly used welding technique that plays an important role in the current practice of PVC window frame manufacturing. Weld seams are an unwanted result of the welding process that must be cleaned to ensure a marketable window frame. When considering the quality of weld seam cleaning, current methods lack the ability to adapt to dynamic situations while ensuring consistent quality. This leads to unpredictability of the cleaning process and a reliance on manual operator capacity to perform any required adjustments or rework.

In light of these challenges, this study proposes a vision-based robotic corner-cleaning framework to enhance the adaptability of weld seam cleaning in window frame production with the aid of vision systems and robotic arms to achieve precise cleaning task execution. The proposed method relies on novel image-processing tools to locate, identify, and quantify the area of weld seam that requires cleaning. The location of the corner area and weld seam are determined based on the locations of the edges of the frame, identified using the Hough transform technique, and this enables the robot arm to perform a guided approximation that does not rely on predefined window positioning or orientation. A Mask R-CNN model is then trained to locate and segment the weld seam. Finally, the mask obtained is transformed using the greedy algorithm to generate, in an automated manner, the robot path required in order to clean the weld seam.

Two scenarios are used to validate the proposed framework—a virtual simulation, and a real scenario involving a UR5e robot furnished with a built-in wrist camera and a Basler ACE camera overlooking the workstation. The proposed approach results in less than 1 cm error for the window location and robot approximation and a 0.95 mean average precision for weld seam detection and segmentation. In future research, given the limitations in terms of accommodating different types of end-effectors and concerns related to generic applicability, solutions will be developed to adapt to various blade types and enlarge the sample database in order to optimize the vision-based robotic corner-cleaning framework.

6.2. Discussion and Limitations

The proposed framework of vision-based robotic corner-cleaning system is found to be capable of identifying and locating window frames, detecting weld seams, and ultimately generating the cleaning path that guides the robotic arm. However, there are two limitations in the execution of the research and the analysis of the results. These limitations, as well as related topics warranting further research, are discussed below.

First, the choice of end-effector can have significant implications for path planning. The types of industrial blades used in manufacturing include circular saws, straight knives, and deburring blades. The end-effector considered in this study is a deburring blade, which is the most commonly used end-effector for edge deburring, chamfering, countersinking, and scraping of material in window manufacturing. However, given the varying designs encountered in window manufacturing, the path planning developed in this research is not a universal solution for all kinds of end-effectors. Accordingly, potential solutions to adapt to different end-effectors warrant further investigation in future research.

Second, although the results of the neural network model show adequate performance, there are concerns of overfitting and a lack of generalization in the dataset. Due to the limited amount of window samples, the dataset used to train the image segmentation model is somewhat homogeneous, as the images are of windows from the same production line. To increase its generic applicability to different window types and materials, the model will need to be retrained and validated using a larger and more diverse dataset.

Although robotic arms are selected for this study because of their precision in executing tasks, a notable limitation of this study is that it only considers the position control, and not the force control, of the robotic arm. Since force control and position control normally function together in practice and are thus calculated and planned together to ensure the precision of the robotic arm, future work in this area should take into consideration the force control aspect. This aspect is crucial when it comes to executing cleaning tasks in practice, including taking into account friction and force/torque sensors. In other words, since most of the weld seams extend horizontally, the force applied to remove the residual material includes a horizontal force as well as the downward force. Both the friction between the window frame and the working table and the horizontal force applied by the robot need to be taken into consideration when determining the amount of force to be applied by the end-effector on a window frame in order to prevent displacement of the window frame during execution of the cleaning task. Furthermore, although for the purpose of the study the workstation area is assumed to be an ideal working environment, some additional support and position-fixed equipment (e.g., clamps, magnetic holders) may be required in practice in order to decrease the possibility of displacement.

Modern industrial robot arms are equipped with sensors to control force and torque, because the force needs to be controlled accordingly to match target objects. Considering the diverse position and orientation of window frames in the study, the optimal sensor controlling force and torque is another key problem to be addressed. For example, the precise amount of force needed to clean the weld seam without causing deformation or other damage to the frames needs to be calculated.

References

[1] S. Mittal, M. A. Khan, D. Romero and T. Wuest, "Smart manufacturing: Characteristics, technologies and enabling factors," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* vol. 233, pp. 1342-1361, 2019.

[2] P. Martinez, M. Al-Hussein and R. Ahmad, "Online vision-based inspection system for thermoplastic hot plate welding in window frame manufacturing," *Procedia CIRP,* vol. 93, pp. 1316-1321, 2020.

[3] S. Narula, S. Prakash, M. Dwivedy, V. Talwar and S. P. Tiwari, "Industry 4.0 adoption key factors: An empirical study on manufacturing industry," *Journal of Advances in Management Research,* vol. 17, no. 5, pp. 697-725, 2020.

[4] Q. Roslan, S. H. Ibrahim, R. Affandi, M. N. M. Nawi and A. Baharun, "A literature review on the improvement strategies of passive design for the roofing system of the modern house in a hot and humid climate region," *Frontiers of Architectural Research,* vol. 5, no. 1, pp. 126-133, 2016.

[5] A. Aldawoud, "Windows design for maximum cross-ventilation in buildings," *Advances in building energy research,* vol. 11, p. 67–86, 2017.

[6] K. Takada, H. Hayama, T. Mori and K. Kikuta, "Thermal insulated PVC windows for residential buildings: feasibility of insulation performance improvement by various elemental technologies," *Journal of Asian Architecture and Building Engineering,* vol. 20, no. 3, pp. 340-355, 2021.

[7] A. P. Kermani, L. Baslu and S. R. Alizadehshahi, "A Model to Plan the Operations and Control the Orders (The Case Study: Profile Door and Window Manufacturing (UPVC))," *Biomedical & Pharmacology Journal,* vol. 8, no. 1, pp. 219-229, 2015.

[8] A. A. Machado, J. C. Zayatz, . M. M. da Silva, G. M. Neto, G. C. L. Leal and R. H. P. Lima, "Aluminum bar cutting optimization for door and window manufacturing," *Dyna,* vol. 87, no. 212, pp. 155-162, 2020.

[9] A. Itani, M. Alghamdy, H. Nazir, S. Sharma and R. Ahmad, "A Decision-Making Tool to Integrate Lean 4.0 in Windows Manufacturing Using Simulation and Optimization Models," in *32nd European Modelling and Simulation Symposium*, 2020.

[10] A. W. AlShaer, L. Li and A. Mistry, "The effects of short pulse laser surface cleaning on porosity formation and reduction in laser welding of aluminium alloy for automotive component manufacture," *Optics & Laser Technology,* vol. 64, pp. 162-171, 2014.

[11] L. Zhu, B. Sun, Z. Li, X. Pan, Y. Chen and Y. Cao, "The weld quality improvement via laser cleaning pre-treatment for laser butt welding of the HSLA steel plates," *Welding in the World,* vol. 64, no. 10, pp. 1715-1723, 2020.

[12] T. T. Vu and H. H. Hoang, "Investigating the Effect of Pulsed Fiber Laser Parameters on the Roughness of Aluminium Alloy and Steel Surfaces in Cleaning Processes," *Lasers in Manufacturing and Materials Processing,* pp. 1-12, 2021.

[13] L. Jin and Y. Liang, "Fiber laser technologies for photoacoustic microscopy," *Visual Computing for Industry, Biomedicine, and Art,* vol. 4, no. 1, pp. 1-13, 2021.

[14] S. Marimuthu, H. K. Sezer and A. M. Kamara, "Applications of laser cleaning process in high value manufacturing industries," in *Developments in Surface Contamination and Cleaning: Applications of Cleaning Techniques*, Elsevier, 2019, pp. 251-288.

[15] A. Gabdrakhmanov and T. Gabdrakhmanova, "Investigation of the effect of waterjet cleaning on the quality of welding," *Materials Today: Proceedings,* vol. 38, pp. 1968-1970, 2021.

[16] A. R. McAndrew, P. A. Colegrove, . A. C. Addison, B. C. Flipo and M. J. Russell, "Modelling the influence of the process inputs on the removal of surface contaminants from Ti–6Al–4V linear friction welds.," *Materials & Design ,* vol. 66, pp. 183-195, 2015.

[17] P. Geng, G. Qin, . H. Ma, J. Zhou, C. Zhang and N. Ma, "Numerical modelling on the plastic flow and interfacial self-cleaning in linear friction welding of superalloys.," *Journal of Materials Processing Technology,* vol. 296, pp. 117-198, 2021.

[18] L. Pérez, Í. Rodríguez, N. Rodríguez, R. Usamentiaga and D. F. García, "Robot guidance using machine vision techniques in industrial environments: A comparative review," *Sensors,* vol. 16, no. 3, p. 335, 2016.

[19] H. N. M. Shah, M. Sulaiman, A. Z. Shukor and M. Z. A. Rashid, "Recognition of butt welding joints using background subtraction seam path approach for welding robot," *International Journal of Mechanical & Mechatronics Engineering,* vol. 17, no. 01, pp. 57-62, 2017.

[20] M. Abdelaal, A. Bahgat, H. M. Emara and A. El-Dessouki, "Real-time external control coupled with vision-based control for an industrial robotic arm to pick and place moving object on a conveyor belt," *International Journal of Mechanical and Mechatronics Engineering,* vol. 19, no. 1, pp. 123-131, 2019.

[21] A. L. Kleppe, A. Bjørkedal, K. Larsen and O. Egeland, "Automated assembly using 3D and 2D cameras," *Robotics,* vol. 6, no. 3, p. 14, 2017.

[22] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research,* vol. 37, pp. 421-436, 2018.

[23] T. Lei, Y. Rong, H. Wang, Y. Huang and M. Li, "A review of vision-aided robotic welding," *Computers in Industry,* vol. 123, p. 103326, 2020.

[24] S.-B. Chen, Y. Zhang, T. Qiu and T. Lin, "Robotic welding systems with vision-sensing and self-learning neuron control of arc welding dynamic process," *Journal of Intelligent and Robotic Systems,* vol. 36, no. 2, pp. 191-208, 2003.

[25] K. Micallef, G. Fang and M. Dinham, "Automatic seam detection and path planning in robotic welding," in *Robotic Welding, Intelligence and Automation*, Springer, 2011, pp. 23-32.

[26] M. Dinham and G. Fang, "Autonomous weld seam identification and localisation using eye-in-hand stereo vision for robotic arc welding," *Robotics and Computer-Integrated Manufacturing,* vol. 29, no. 5, pp. 288-301, 2013.

[27] S. Minaee, Y. Y. Boykov, . F. Porikli, . A. J. Plaza, . N. Kehtarnavaz and . D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2021.

[28] . H. Huang, . Z. Wei and L. Yao, "A novel approach to component assembly inspection based on mask R-CNN and support vector machines," *Information,* vol. 10, no. 9, p. 282, 2019.

[29] Q. Zhu, S. Liu and W. Guo, "Research on Vehicle Appearance Component Recognition Based on Mask R-CNN," in *Journal of Physics: Conference Series*, vol. 1335, IOP Publishing, 2019, p. 012026.

[30] . C. Xia, . Z. Pan, S. Zhang, J. Polden, H. Li, Y. Xu and . S. Chen, "Mask R-CNN-Based Welding Image Object Detection and Dynamic Modelling for WAAM," in *Transactions on Intelligent Welding Manufacturing*, Springer, 2020, pp. 57-73.

[31] F. He, X. Sun, Y. Wang, S. Rong and Y. Hu, "Research on Weld Recognition Method Based on Mask R-CNN," in *2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, IEEE, 2021, pp. 545-551.

[32] X. Jin, L. Lv, C. Chen, . F. Yang and T. Chen, "A New Welding Seam Recognition Methodology Based on Deep Learning Model MRCNN," in *2020 7th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*, IEEE, 2020, pp. 767-771.

[33] . Y. Yang, Z. Li, X. Yu, . Z. Li and . H. Gao, "A trajectory planning method for robot scanning system uuuusing mask R-CNN for scanning objects with unknown model," *Neurocomputing,* vol. 404, pp. 329-339, 2020.

[34] A. R. Hevner, S. T. March, J. Park and S. Ram, "Design science in information systems research," *MIS quarterly,* pp. 75-105, 2004.

[35] A. Waleed , "Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow," 2017. [Online]. Available: https://github.com/matterport/Mask_RCNN.

[36] J. Illingworth and . J. Kittler, "A survey of the Hough transform," *Computer vision, graphics, and image processing,* vol. 44, no. 1, pp. 87-116, 1988.

[37] "The URScript Programming Language," 31 May 2019. [Online]. Available: https://www.siemens-pro.ru/docs/ur/scriptManual.pdf.

[38] A. Dutta and A. Zisserman, "The VIA Annotation Software for Images, Audio and Video," in *Proceedings of the 27th ACM international conference on multimedia*, 2019, pp. 2276-2279.

[39] S. Wang, G. Sun, B. Zheng and Y. Du, "A Crop Image Segmentation and Extraction Algorithm Based on Mask RCNN," *Entropy,* vol. 23, no. 9, p. 1160, 2021.

[40] Y. Duan, "Welding Seam Recognition Robots Based on Edge Computing," in *2020 International Conference on Computing and Data Science (CDS)*, 2020, pp. 27-30.

[41] Y. Tian, H. Liu, L. Li, G. Yuan, J. Feng, Y. Chen and W. Wang, "Automatic Identification of Multi-Type Weld Seam Based on Vision Sensor With Silhouette-Mapping," *IEEE Sensors Journal,* vol. 21, no. 4, pp. 5402-5412, 2020.

[42] A. Rout, B. Deepak and B. Biswal, "Advances in weld seam tracking techniques for robotic welding: A review," *robotics and computer-integrated manufacturing,* vol. 56, pp. 12-37, 2019.

[43] J. Xu, G. Zhang, Z. Hou, J. Wang, J. Liang, X. Bao, W. Yang and W. Wang, "Advances in multi-robotic welding techniques: A review," *Int. J. Mech. Eng. Robot. Res,* vol. 9, pp. 421-328, 2020.

[44] "Wrist Camera - Robotiq," [Online]. Available: https://robotiq.com/products/wrist-camera. [Accessed October 2021].

[45] . L. A. Lee, A. McAndrew, C. Buhr, P. A. Colegrove and K. Beamish, "2D Linear Friction Weld Modelling of a Ti-6Al-4V T-Joint.," *Journal of Engineering Science &*

Technology Review, vol. 8, no. 6, 2015.

Appendix A : Python scripts for the cleaning process

Table of Contents

1. Window location and identification

1.1. Edge detection

```
import os
```

```
os.environ["CUDA_VISIBLE_DEVICES"] = "1"
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2 as cv
import numpy as np

#detect the line using SHT
def line_detection(image):
    img = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    #load camera parameters
    camMtx =
np.genfromtxt("/content/drive/MyDrive/Research/Thesis/Module1/camera/intri
nsics.csv", delimiter = ",")
    dist =
np.genfromtxt("/content/drive/MyDrive/Research/Thesis/Module1/camera/disto
rtion.csv", delimiter = ",")

    #image processing
    img = cv.undistort(img, camMtx, dist, None)
    img = cv.GaussianBlur(img,(3,3),0)
    nimg = cv.bitwise_not(img)

    # compute the median of the single channel pixel intensities
    v = np.median(img)
    nv = np.median(nimg)
    sigma = 0.33

    #Canny edge detection
    lower = int(max(0, (1.0 - sigma) * v))
    upper = int(min(255, (1.0 + sigma) * v))
    edges = cv.Canny(img, lower, upper)
    nlower = int(max(0, (1.0 - sigma) * nv))
    nupper = int(min(255, (1.0 + sigma) * nv))
    nedges = cv.Canny(nimg, nlower, nupper)

    cv.imwrite("edges.png",edges)
    cv.imwrite("negative_edges.png",nedges)

    # run Hough on edge detected image
    if nedges.mean() > edges.mean():
        lines = cv.HoughLines(nedges,1,np.pi/180,135)
    else:
        lines = cv.HoughLines(edges,1,np.pi/180,135)
    # clean noisy lines due to distortion correction
    height, width = img.shape
    alpha = 0.05
    c = 0
    for i,line in enumerate(lines):
        if (abs(line[0][0]) > (1-alpha)*width) | (abs(line[0][0]) <
alpha*height) | ((abs(line[0][0]) > (1-alpha)*height) & (abs(line[0][0]) <
height) & (abs(line[0][1]) > np.pi/4)):
            lines = np.delete(lines,i-c,axis=0)
            c = c+1
    res_img = img.copy()


    #plot the lines
    for line in lines:
        for rho,theta in line:
          #rho,theta=line[0]
          a=np.cos(theta)
          b=np.sin(theta)
          x0=a*rho
          y0=b*rho
          x1=int(x0+100000*(-b))
          y1=int(y0+100000*a)
          x2=int(x0-100000*(-b))
          y2 = int(y0 - 100000 * a)
          cv.line(image,(x1,y1),(x2,y2),(0,0,255),2)
          print("Line", ":[", x1,",",y1,"],[",x2,", ",y2,"]")
       #   ("Line",  ":(", x1,y1,"),(",x2,y2,")")


    cv.imwrite("img lines.png",image)
    plt.show(image)
    images = np.hstack((img,nimg,edges,nedges,res_img))

    cv.namedWindow("image", cv.WINDOW_AUTOSIZE)
    cv.startWindowThread()

    # separate clusters in arrays
    for i in range(len(segmented)):
        for j,line in enumerate(segmented[i]):
            for rho,theta in line:
                if rho < 0:
                    segmented[i][j][0][0] = abs(rho)
                    segmented[i][j][0][1] = theta - 2*(theta-np.pi/2)

    dist_cluster0 = np.array([line[0][0] for line in segmented[0]])
```

```
dist_cluster1 = np.array([line[0][0] for line in segmented[1]])
dist_cluster2 = np.array([line[0][0] for line in segmented[2]])
angles_cluster0 = np.array([line[0][1] for line in segmented[0]])
angles_cluster1 = np.array([line[0][1] for line in segmented[1]])
angles_cluster2 = np.array([line[0][1] for line in segmented[2]])

# plot results
fig = plt.figure()
ax = fig.add_subplot(111, polar=True)
ax.scatter(angles_cluster0,abs(dist_cluster0), s= 100, color='green')
ax.scatter(angles_cluster1,abs(dist_cluster1), s= 100, color='blue')
ax.scatter(angles_cluster2,abs(dist_cluster2), s= 100, color='red')
ax.set_theta_zero_location('S')
plt.show()

fig.savefig('polar_results.png',bbox_inches='tight')

# determine each cluster origins
m_cluster0 = np.mean(angles_cluster0)*180/np.pi
m_cluster1 = np.mean(angles_cluster1)*180/np.pi
m_cluster2 = np.mean(angles_cluster2)*180/np.pi
if (m_cluster0 > m_cluster1) & (m_cluster0 > m_cluster2):
    vert_angles = angles_cluster0
    if m_cluster1 > m_cluster2:
        diag_angles = angles_cluster1
        horz_angles = angles_cluster2
    else:
        diag_angles = angles_cluster2
        horz_angles = angles_cluster1
elif (m_cluster1 > m_cluster0) & (m_cluster1 > m_cluster2):
    vert_angles = angles_cluster1
    if m_cluster0 > m_cluster2:
        diag_angles = angles_cluster0
        horz_angles = angles_cluster2
    else:
        diag_angles = angles_cluster2
        horz_angles = angles_cluster0
else:
    vert_angles = angles_cluster2
    if m_cluster0 > m_cluster1:
        diag_angles = angles_cluster0
        horz_angles = angles_cluster1
    else:
        diag_angles = angles_cluster1
        horz_angles = angles_cluster0
```

1.2. Intersection point

```
import cv2 as cv
import numpy as np

def line_intersection(line1, line2):
    xdiff = (line1[0][0] - line1[1][0], line2[0][0] - line2[1][0])
    ydiff = (line1[0][1] - line1[1][1], line2[0][1] - line2[1][1])

    def det(a, b):
        return a[0] * b[1] - a[1] * b[0]

    div = det(xdiff, ydiff)
    if div == 0:
        raise Exception('lines do not intersect')

    d = (det(*line1), det(*line2))
    x = det(d, xdiff) / div
    y = det(d, ydiff) / div
    return x, y


[a,b] = line_intersection(lineA, lineB)
x = np.round(a).astype("int")
y = np.round(b).astype("int")
print(x,y)
```

1.3. coordinate transformation

```
import cv2 as cv
import numpy as np

# project the point on image to 3d camera space
def pinhole_model(x, y):

    # location of camera, unit: mm, origin point
    L = [0, 0, 0]

    # reference point in camera system, unit: mm, fixed value
    r0 = [500,500,-500]
    # reference point: on image, unit: pixel, fixed value
    r = [900,900]

    # focalLength
    f = 250

    # Calculate the real coordinate using pinhole model
    # W = (DxP)/F
    x_real = (L[2]-r0[2])*(x-r[0])/f
    y_real = (L[2]-r0[2])*(y-r[1])/f

    position = [x_real, y_real]
    return position

#projecct the point from camera into robot coordinate system
def locate_start_point(x,y):

    # Define the origin point in robot space
    RL = [0,0,0]

    #Assume the distance between the two origin point
    LinR = [500,500,600]

    #calculate the start point
    SPoint = [x+LinR[0], y+LinR[1], 300]]

    return SPoint
```

2. Weld seam detection

2.1. Model training of Mask R-CNN

2.1.1. Data augmentation

```
from numpy import expand_dims
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import save_img
from matplotlib import pyplot

#load image
img = load_img("/content/drive/MyDrive/Research/Thesis/Training/sample/1 -
45.png")
data = img_to_array(img)
# expand dimension to one sample
samples = expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(rotation_range=90)
```

```
# prepare iterator
it = datagen.flow(samples,
batch_size=1,save_to_dir="/content/drive/MyDrive/Research/Thesis/Training/
training photos")
# generate samples and plot
for i in range(9):
  # define subplot
  pyplot.subplot(330 + 1 + i)
  # generate batch of images
  batch = it.next()
  # convert to unsigned integers for viewing
  image = batch[0].astype('uint8')
  # plot raw pixel data
  pyplot.imshow(image)
# show the figure
pyplot.show()
```

2.1.2. Environmental setup and model training

```
!pip install git+https://github.com/minetorch/minetorch.git
!pip install tensorflow==1.15.0
# to solve AttributeError: module 'tensorflow_core.compat.v2' has no
attribute '__internal__'
!pip uninstall keras-nightly
!pip install h5py==2.10.0
!pip list | grep tf
!pip install 'h5py==2.10.0' --force-reinstall
!pip install tensorflow-estimator==1.15.1

# downgrage keras to solve ModuleNotFoundError: No module named
'keras.saving'
!pip install keras==2.1.5 --force-reinstall --no-deps --no-cache-dir

import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import keras

# unzip the image dataset
!mkdir ./data
!cp /content/Datasets.zip ./data/
!cd ./data && unzip Datasets.zip

# model training with COCO
```

```
%cd
/content/drive/MyDrive/Research/Thesis/Training/Project/main/Mask_RCNN-TF2
!python3 train.py train --dataset=/content/data/Datasets --weights=coco
```

2.2. Mask Prediction

2.2.1. Environmental setup

```
# environmental setting
!pip install tensorflow==1.15.0
!pip install keras==2.0.8 --force-reinstall --no-deps --no-cache-dir

# to solve AttributeError: module 'tensorflow_core.compat.v2' has no
attribute '__internal__'
!pip uninstall keras-nightly
!pip install h5py==2.10.0

# to solve error: ModuleNotFoundError: No module named 'mrcnn'
%cd
/content/drive/MyDrive/Research/Thesis/Training/Project/main/Mask_RCNN-TF2
!python setup.py install

# to solve error: ModuleNotFoundError: No module named
'keras.utils.generic_utils'
!pip list | grep tf
!pip install 'h5py==2.10.0' --force-reinstall

import tensorflow as tf
print(tf.__version__)
import keras
!pip install tensorflow-estimator==1.15.1
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

# to solve ModuleNotFoundError: No module named
'keras.utils.training_utils'`
!pip uninstall keras
!pip install keras==2.2.4


import os
import sys
import random
import math
```

```python
import re
import time
import numpy as np
import tensorflow as tf
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.patches as patches

# Root directory of the project
ROOT_DIR = os.path.abspath("/content/drive/MyDrive/My
Drive/Research/Thesis/Training/Project/main")

# Import Mask RCNN
sys.path.append(ROOT_DIR)  # To find local version of the library
from mrcnn import utils
from mrcnn import visualize
from mrcnn.visualize import display_images
import mrcnn.model as modellib
from mrcnn.model import log

%matplotlib inline

# Directory to save logs and trained model
MODEL_DIR = os.path.join(ROOT_DIR, "logs")

# Local path to trained weights file
COCO_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")
# Download COCO trained weights from Releases if needed
if not os.path.exists(COCO_MODEL_PATH):
    utils.download_trained_weights(COCO_MODEL_PATH)

# Path to Shapes trained weights
SHAPES_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_object_0050.h5")
```

2.2.2. Configurations

```python
import sys
import os
import json
import datetime
import numpy as np
import skimage.draw
import cv2
from mrcnn.visualize import display_instances
from mrcnn.visualize import draw_rois
from mrcnn.visualize import display_contours
```

```
import matplotlib.pyplot as plt

# Root directory of the project
ROOT_DIR = os.path.abspath("/content/drive/MyDrive/My
Drive/Research/Thesis/Training/Project/main")

# Import Mask RCNN
sys.path.append(ROOT_DIR)  # To find local version of the library
from mrcnn.config import Config
from mrcnn import model as modellib, utils


# Run one of the code blocks

# Shapes toy dataset
#import object
config = CustomConfig()
CUSTOM_DIR = os.path.join(ROOT_DIR, "mask_rcnn_object_0010.h5")


from mrcnn.config import Config
# Override the training configurations with a few
# changes for inferencing.
class InferenceConfig(config.__class__):
    # Run detection on one image at a time
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1

config = InferenceConfig()
config.display()
```

2.2.3. Notebook Preferences

```
# Device to load the neural network on.
# Useful if you're training a model on the same
# machine, in which case use CPU and leave the
# GPU for training.
DEVICE = "/cpu:0"  # /cpu:0 or /gpu:0

# Inspect the model in training or inference modes
# values: 'inference' or 'training'
# TODO: code for 'training' test mode not ready yet
TEST_MODE = "inference"
```

```
def get_ax(rows=1, cols=1, size=16):
    """Return a Matplotlib Axes array to be used in
    all visualizations in the notebook. Provide a
    central point to control graph sizes.

    Adjust the size attribute to control how big to render images
    """
    _, ax = plt.subplots(rows, cols, figsize=(size*cols, size*rows))
    return ax
```

2.2.4. Load Validation Dataset

```
# Build validation dataset
if config.NAME == "object":
  CUSTOM_DIR =
"/content/drive/MyDrive/Research/Thesis/Training/Project/main/Mask_RCNN-
TF2/data/testDatasets"
  dataset = CustomDataset()
  dataset.load_custom(CUSTOM_DIR, "val")
elif config.NAME == "coco":
    dataset = coco.CocoDataset()
    dataset.load_coco(COCO_DIR, "minival")
print(config.NAME)
# Must call before using the dataset
dataset.prepare()
print("Images: {}\nClasses: {}".format(len(dataset.image_ids),
dataset.class_names))
```

2.2.5. Load Model and run detection

```
# Create model in inference mode
with tf.device(DEVICE):
    model = modellib.MaskRCNN(mode="inference", model_dir=MODEL_DIR,
                              config=config)

# Set weights file path
if config.NAME == "object":
    weights_path = SHAPES_MODEL_PATH
elif config.NAME == "COCO":
    weights_path = COCO_MODEL_PATH
# Or, uncomment to load the last model you trained
# weights_path = model.find_last()
# Load weights
print("Loading weights ", weights_path)
model.load_weights(weights_path, by_name=True)


# Run detection
image_id = random.choice(dataset.image_ids)
image, image_meta, gt_class_id, gt_bbox, gt_mask =\
    modellib.load_image_gt(dataset, config, image_id, use_mini_mask=False)
info = dataset.image_info[image_id]
print("image ID: {}.{} ({}) {}".format(info["source"], info["id"],
image_id,
                                       dataset.image_reference(image_id)))
# Run object detection
results = model.detect([image], verbose=1)

# Display results
ax = get_ax(1)
r = results[0]
visualize.display_instances(image, r['rois'], r['masks'], r['class_ids'],
                            dataset.class_names, r['scores'], ax=ax,
                            title="Predictions")
log("gt_class_id", gt_class_id)
log("gt_bbox", gt_bbox)
log("gt_mask", gt_mask)
```

3. Cleaning Path Generation

3.1. Extraction of Contours

```
# extract the contour through the display_contours function from
visualize.py file
c = visualize.display_contours(image, r['rois'], r['masks'], r['class_ids'
dataset.class_names, r['scores'], ax=ax,title="Predictions")

# store contour into an a x b matrix
a = 2
b = int(np.size(c)/2)
mask = r['masks']
mask = mask.astype(int)
y_coo = mask.shape[1]
print(y_coo)
# empty array to store data
```

```
mask_polygon = np.empty(shape=[0, a])
# store all the coordinates into a list mask_polygon
for j in range(b):
    mask_polygon = np.append(mask_polygon, [[c[0][0][j][1], y_coo-
c[0][0][j][0]]], axis=0)


# visulize the contour
from shapely.geometry import Polygon
from matplotlib import pyplot as plt
from matplotlib.pyplot import MultipleLocator

ax = plt.gca()
x_major_locator = MultipleLocator(200)
y_major_locator = MultipleLocator(200)
ax.xaxis.set_major_locator(x_major_locator)
ax.yaxis.set_major_locator(y_major_locator)
plt.xlim(0,1024)
plt.ylim(0,1024)

# plot the contour
poly = Polygon(mask_polygon)
x, y = poly.exterior.xy
plt.plot(x,y,color='orange')
plt.show
```

3.2. Calculation of two end-points

```
size_of_list = int(np.size(mask_polygon)/2)
print(size_of_list)

# find the farthest two points
Max_L = 0
point = []
index = 0

for i in range(size_of_list):
    for j in range(size_of_list):
      distance = np. linalg. norm(mask_polygon[i]- mask_polygon[j])
      if distance > Max_L:
        point = [mask_polygon[i],mask_polygon[j]]
        index = i
        index_2 = j
      else:
        pass
      Max_L = max(Max_L, distance)

# map the endpoints with the contour
print(np.linalg.norm(point[0]-point[1]))

plt.plot(point[0][0], point[0][1], "ro")
plt.plot(point[1][0], point[1][1],"ro")

x_major_locator = MultipleLocator(200)
y_major_locator = MultipleLocator(200)

ax.xaxis.set_major_locator(x_major_locator)
ax.yaxis.set_major_locator(y_major_locator)
plt.xlim(0,1024)
plt.ylim(0,1024)

p1 = [point[0][0], point[0][1]]
p2 = [point[1][0], point[1][1]]

plt.plot(x,y,color='orange')
plt.show
```

3.3. Path calculation

```
# create a new list to store path coordinates
print(mask_polygon[index])
print(i)
new_size_of_list = size_of_list-1
print(new_size_of_list )
path = np.empty(shape=[0, 2])
k = index+1
l = index-1

# calculate the center points from the endpoint p1
while k!=index or l!=index:
  if k == index_2 or l == index_2:
    break
  elif k <= new_size_of_list and l>=0:
    new_point =
((mask_polygon[k][0]+mask_polygon[l][0])/2,(mask_polygon[k][1]+mask_polygo
n[l][1])/2)
    path = np.append(path, [new_point], axis = 0)
    k = k + 1
    l = l- 1
  elif k > new_size_of_list and l>=0:
    k = 0
    new_point =
((mask_polygon[k][0]+mask_polygon[l][0])/2,(mask_polygon[k][1]+mask_polygo
n[l][1])/2)
    path = np.append(path, [new_point], axis = 0)
    k = k + 1
    l = l- 1
  elif k > new_size_of_list and l<0:
    k = 0
    l = new_size_of_list
    new_point =
((mask_polygon[k][0]+mask_polygon[l][0])/2,(mask_polygon[k][1]+mask_polygo
n[l][1])/2)
    path = np.append(path, [new_point], axis = 0)
    k = k + 1
    l = l- 1
  elif k<=new_size_of_list and l<0:
    l = new_size_of_list
    new_point =
((mask_polygon[k][0]+mask_polygon[l][0])/2,(mask_polygon[k][1]+mask_polygo
n[l][1])/2)
    path = np.append(path, [new_point], axis = 0)
    k = k + 1
    l = l - 1


r_1 = int(np.size(path)/2)

# transform the format of the list
co_x = []
co_y = []
for h in range(r_1):
  co_x.append(path[h][0])
  co_y.append(path[h][1])

# map the path with the detection image
x1 = np.array(co_x)
y1 = np.array(co_y)

ax = get_ax(1)
r = results[0]
visualize.display_instances(image, r['rois'], r['masks'], r['class_ids'],
                            dataset.class_names, r['scores'], ax=ax,
                            title="Predictions")
```

```
log("gt_class_id", gt_class_id)
log("gt_bbox", gt_bbox)
log("gt_mask", gt_mask)
ax.xaxis.set_major_locator(x_major_locator)
ax.yaxis.set_major_locator(y_major_locator)
plt.plot(x1, y1)
plt.plot(x1,y1,color='white', linewidth= 5)
plt.show
```