technologies

MDPI

Communication

ARSIP: Automated Robotic System for Industrial Painting

Hossam A. Gabbar * and Muhammad Idrees

Faculty of Engineering and Applied Science, Ontario Tech University (UOIT), Oshawa, ON L1G 0C5, Canada
* Correspondence: hossam.gaber@ontariotechu.ca

Abstract: This manuscript addresses the critical need for precise paint application to ensure product durability and aesthetics. While manual work carries risks, robotic systems promise accuracy, yet programming diverse product trajectories remains a challenge. This study aims to develop an autonomous system capable of generating paint trajectories based on object geometries for user-defined spraying processes. By emphasizing energy efficiency, process time, and coating thickness on complex surfaces, a hybrid optimization technique enhances overall efficiency. Extensive hardware and software development results in a robust robotic system leveraging the Robot Operating System (ROS). Integrating a low-cost 3D scanner, calibrator, and trajectory optimizer creates an autonomous painting system. Hardware components, including sensors, motors, and actuators, are seamlessly integrated with a Python and ROS-based software framework, enabling the desired automation. A web-based GUI, powered by JavaScript, allows user control over two robots, facilitating trajectory dispatch, 3D scanning, and optimization. Specific nodes manage calibration, validation, process settings, and real-time video feeds. The use of open-source software and an ROS ecosystem makes it a good choice for industrial-scale implementation. The results indicate that the proposed system can achieve the desired automation, contingent upon surface geometries, spraying processes, and robot dynamics.

Keywords: automated painting; hybrid optimization; genetic algorithm; 3D scanning; ROS; GUI

Citation: Gabbar, H.A.; Idrees, M. ARSIP: Automated Robotic System for Industrial Painting. *Technologies* **2024**, *12*, 27. https://doi.org/10.3390/technologies12020027

Academic Editor: George F. Fragulis

Received: 10 December 2023
Revised: 7 February 2024
Accepted: 10 February 2024
Published: 19 February 2024

1. Introduction

In modern manufacturing, industrial painting has gained significant importance. Applying paint to a product's surface not only enhances its appearance but also extends its durability. Robotics are pivotal in these painting processes, boosting efficiency, productivity, and the quality of the painted surface. Forecasts predict a rise in vehicle production to 111.7 million units by 2023 [1]. Moreover, manual painting processes for industrial parts pose challenges: inconsistent coating quality, prolonged production times, increased environmental impact due to VOC emissions [2], and compromised worker safety [3]. This study aims to explore and implement specific solutions like automation, eco-friendly coatings, and process streamlining to address these issues.

Automating spray painting requires an accurate understanding of object geometry, spraying dynamics, and robot movement. An early 1980s system integrated a paint booth, robot apparatus, and rail mechanism to streamline painting and cut down on wasted paint [4]. Similarly, a software–hardware prototype of an integrated robotic painting system has been developed [5]. The software manages part designs, process planning, robot trajectory generation, and motion control, while the hardware components include a work cell controller, motor drives, a robotic manipulator, a surface scanner, and paint delivery units. Another integrated system utilizes an algorithm to model spray painting and a computer program to simulate a robot for painting curved surfaces [6]. This program optimizes spray painting parameters like gun velocity, distance, and multiple paint paths. Hence, key components in automating painting with robots involve a 3D scanner to create object geometry, a trajectory planner for optimized robot paths, and a robotic system to execute the planned trajectories.

Researchers have explored methods for digitizing geometric models of objects through geometric reconstruction and 3D scanning [7]. Common techniques like CMMs (Coordinate Measuring Machines), laser scanners, and CT (Computed Tomography) scanners serve this purpose. A CMM, while slow due to single-point measurement, is inefficient and uses traditional equipment, so it is unsuitable for swift scanning. Laser scanners, however, offer rapid scanning [8–10]. Kinect Fusion, employing a low-cost depth camera, captures indoor 3D scenes in varying lighting conditions in real time [11]. Metrics like per-vertex Euclidean and angle errors gauge the accuracy of this method for 3D scenes [12]. Sparse reconstruction techniques generate 3D environments from limited depth scans, capitalizing on surface and edge regularity for high reconstruction accuracy [13]. Integrated 3D scanning systems, comprising hardware like line-profile laser scanners, industrial robots, and turntable mechanisms, capture physical object representations, converting them into CAD models [14]. Robotic systems designed for contour tracing employ six-DOF robotic arms, short-range laser scanners, and turntables [15]. Similarly, systems for large-scale object scanning propose laser scanners, turntable mechanisms, and calibrated robots [16].

Gaining a geometric model enables surface trajectory planning, vital for unknown parts' spray trajectories. By employing a direct *PFeatureDetector* approach, this method extracts basic geometries from range sensor data [17]. For spray painting robots, an incremental trajectory generation approach utilizes surface, coating thickness, and spray process models to plan painting paths [18]. Additionally, research explores Bezier curves for planning paint spray paths [19]. The process models paint distribution on a circular area, assuming consistent surface overlap. Using T-Bezier curves in trajectory planning ensures efficient computation, mapping paths along the geometry's U and V principal directions. Recent research indicates using a point cloud slicing technique alongside a coating thickness model to create paint trajectories [20]. These methods rely on object geometry obtained from a laser sensor. Defining key geometric variables on a free-form surface establishes a coating thickness model. Slicing the point cloud yields a specific portion, where a grid projection algorithm extracts points for thickness computation. The optimal slice width and paint gun velocity are determined using the golden section method. This process iterates for all slices, covering the entire surface. Additionally, a new algorithm optimizes transitional segments (straight lines and concave and convex arcs) within intermediate triangular patches of a CAD model, forming a transitional segment trajectory [21].

Furthermore, the integration of components like a 3D sensor, robotic arm, and spray-painting unit is pivotal for autonomy. Specialized robots like KUKA's KR AGILUS KR 10 R1100 [22], FANUC's P-250iB/15 [23], and ABB's IRB 5500 Flex Painter [24] excel in industrial painting. For instance, KUKA's KR AGILUS KR 10 R1100, designed specifically for paint applications, offers a 10 kg wrist payload, a reach of 1100 mm, and six axes. Similarly, FANUC's P-250iB/15, a larger robot, adapts to walls, floors, or narrow spaces. ABB's IRB 5500 Flex Painter is also proficient in industrial painting. Specialized software enables simulation, providing a user-friendly interface for paint trajectory testing and robot program development. This software includes CAD robot models, sample objects for trajectory testing, collision avoidance systems, axis limits, and postprocessors for generating robot programs. Some commercial simulators for paint robots include *RoboDK* [25], *RobCad paint* [26], *Delfoi Paint* [27], *RobotStudio*® Paint *PowerPac* [28], *OLP Automatic* [29], and *RoboGuide PaintPRO* [30].

This paper extensively outlines the development process of ARSIP (automated robotic system for industrial painting), building upon our prior work titled "A hybrid optimization scheme for efficient trajectory planning of a spray-painting robot" [31]. The development process involves integrating a 3D scanner, trajectory planner, and hardware and software components, all discussed herein. Specifically, Section 2 covers system design prototyping involving the 3D scanner, calibrator, trajectory planner, and their integration. Section 3 details the software framework, including the GUI (Graphical User Interface). The optimization results for spray painting processes are outlined in Section 4, while Section 5 provides concluding remarks and future recommendations.

2. System Design and Prototyping

To practically apply a 3D scanning system and trajectory optimizer, creating an integrated system with essential components is crucial for automating the process. This system comprises both hardware and software working in tandem to establish an autonomous robotic painting setup. The hardware elements encompass sensors and actuators, responsible for the system functions' sensing and control. This section will delve into the comprehensive design and development phases of this integrated system, including the 3D scanner and calibrator, the optimal trajectory planner, and a hardware breakdown of the integrated system.

2.1. 3D Scanner and Calibrator

A combination of a rotating turntable mechanism paired with an Intel RealSense D435 sensor [32] captures an object's geometric model, as shown in Figures 1 and 2. The use of an RGBD sensor cuts costs significantly since industrial laser scanners are very expensive. The turntable's servomotor [33] allows precise control with a resolution of 1 degree. Positioning the D435 sensor at 0.47 m from the object accommodates thicker objects, considering the sensor's minimum range of 0.3 m. The object undergoes 30-degree rotations, with RGB and depth images stored for each angle. These images are then transformed into point clouds using the camera projection matrix [34]. Applying a box filter isolates the region of interest, effectively eliminating most of the noisy point cloud data, followed by statistical noise reduction for further refinement [35]. Subsequently, raw alignment is employed to align the 3D scans, succeeded by ICP (Iterative Closest Point) registration [36]. The reference frame of the 3D scanner and calibrator is the camera frame, {C}, which coincides with the origin of the RGBD sensor. A single-point depth sensor with a reference frame {S} is situated directly above the RGBD sensor to locate the real-time position of the axis of rotation. The frame $\{S_1\}$ has the same orientation as the frame {S} and locates the origin of the axis of rotation. To track the rotational angle of the object, another frame, $\{S_{1r}\}$, is introduced, with its origin coinciding with {S} and rotating with the object of interest. With a point cloud, $P^{C(i)}$, in the camera frame, {C}, corresponding to a rotational angle $\theta_{y(i)}$, this cloud can be expressed in the frame $\{S_{1r}\}$ through the following transformations:

$$P^{S_{1r}(i)} = {}^{S_{1r}}_{C}T \; P^{C(i)} \tag{1}$$

${}^{S_{1r}}_{C}T$ can be obtained using the following expression:

$${}^{S_{1r}}_{C}T = {}^{S_{1r}}_{S_1}T \; {}^{S_1}_{S}T \; {}^{S}_{C}T \tag{2}$$

The transformations ${}^{S_1}_{C}T$ and ${}^{S}_{C}T$ are derived through linear offsets along the z- and y-axes within the frames $\{S_1\}$ and {S}, respectively. Meanwhile, ${}^{S_{1r}}_{S_1}T$ is achieved by applying a relative rotation matrix along the *y*-axis, as follows:

$${}^{S_{1r}}_{S_1}T = R_y\left(-\theta_{y(i)}\right) \tag{3}$$

The point cloud in the frame $\{S_{1r}\}$ can be represented by combing Equations (1)–(3).

$$P^{S_{1r}(i)} = R_y\left(-\theta_{y(i)}\right) {}^{S_1}_{S}T \; {}^{S}_{C}T \; P^{C(i)} \tag{4}$$

The aligned point clouds are transformed back into the camera reference frame, {C}, using the transformation matrix ${}^{C}_{S_1}T$:

$$P^{C(i)}_{aligned} = {}^{C}_{S_1}T \; P^{S_{1r}(i)} \tag{5}$$

These individual point clouds are subsequently merged into a single point cloud by employing the optimal rotation matrix, R, and translation vector, t, acquired through the ICP:

$$P_{scan}^{C(i)} = RP_{aligned}^{C(i)} + t \tag{6}$$

The optimal rotation matrix, R, and translation vector, t, are acquired by minimizing the objective functions in the point-to-point or point-to-plane ICP [36], as described below:

$$E(R,t) = \frac{1}{N_{pcd}} \sum_{i=1}^{N_{pcd}} \|q_i - Rp_i - t\|^2 \tag{7}$$

$$E(R,t) = \frac{1}{N_{pcd}} \sum_{i=1}^{N_{pcd}} \left\| (q_i - Rp_i - t).\, n_{q_i} \right\|^2 \tag{8}$$

Algorithm 1 describes the process of ICP fine alignment.

Algorithm 1: ICP fine alignment

Input: $P_{aligned}^{C(i)}$ **(Raw aligned point clouds for each index** i**).** $N = \sum i$ **(# of point clouds)**

Output: P_{scan}^{C} **(Merged point cloud in the frame {C})**

1. Assign merged point cloud to the point cloud at index 0

$$P_{scan}^{C} = P_{aligned}^{C(i)}$$

For i in range $(N-1)$: **(Perform step 2 to step 5)**

2. Apply the point-to-point ICP and obtain optimal *R* and *t*

$$(R,t)_{p2p} = p2pICP\left(P_{scan}^{C}, P_{aligned}^{C(i+1)}\right)$$

3. Apply the point-to-plane ICP and obtain optimal *R* and *t*

$$(R,t)_{p2plane} = p2planeICP\left(P_{scan}^{C}, P_{aligned}^{C(i+1)}\right)$$

4. Compare fitness scores

$$\begin{gathered} if\ \left(fitness_{p2p} \geq fitness_{p2plane}\right): \\ (R,t) = (R,t)_{p2p} \\ else\ \text{if}\ \left(fitness_{p2p} < fitness_{p2plane}\right): \\ (R,t) = (R,t)_{p2plane} \end{gathered}$$

5. Transform the source point cloud to the target and merge

$$P_{scan}^{C} = merge\left(P_{scan}^{C},\ T_{(R,t)}P_{aligned}^{C(i+1)}\right)$$

If the ICP-registered point cloud (P_{scan}^{C}) has noise and missing details, the CAD model can be used for trajectory planning instead. Given P_{scan}^{C} and P_{CAD}, representing the scan point cloud in the frame {C} and the CAD point cloud in some arbitrary frame, their transformations into an eigen coordinate system yield the following:

$$P_{scan}^{eig} = u_{scan}\left(P_{scan}^{C} - c_{scan}\right) \tag{9}$$

$$P_{CAD}^{eig} = u_{CAD}(P_{CAD} - c_{CAD}) \tag{10}$$

u_{scan} and c_{scan} represent the eigenvectors and principal center of the scan, while u_{CAD} and c_{CAD} represent the eigenvectors and principal center of the CAD. Using the ICP, P_{CAD}^{eig}

is aligned in the reference frame of P_{scan}^{eig} via the optimal rotation matrix, R, and translation vector, t, such that:

$$P_{CAD}^{aligned} = R\, P_{CAD}^{eig} + t \tag{11}$$

The aligned CAD model can then be represented in the frame {C} by applying the following transformations:

$$P_{CAD}^{C} = u_{scan}^{-1}\, P_{CAD}^{aligned} + c_{scan} \tag{12}$$

The fully calibrated CAD point cloud (P_{CAD}^{C}) forms the basis for our trajectory planning and optimization processes.

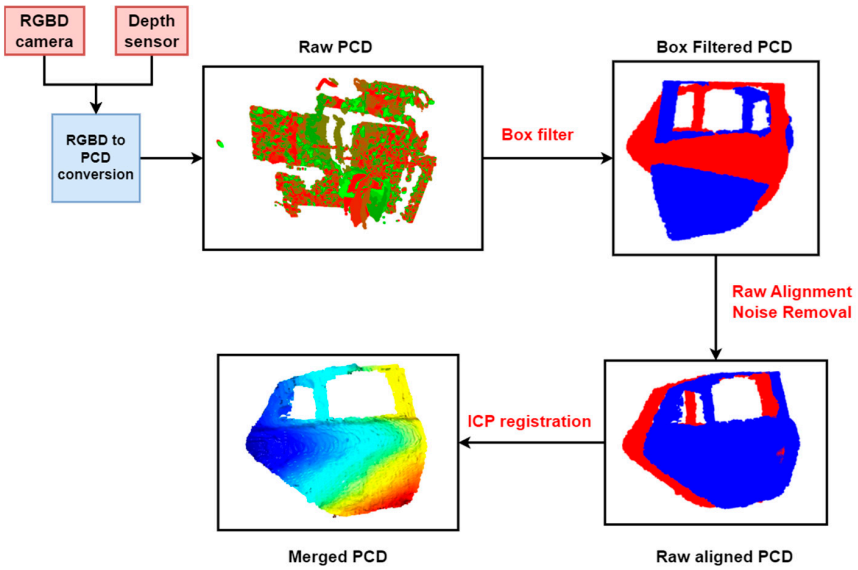


Figure 1. 3D scanning methodology.

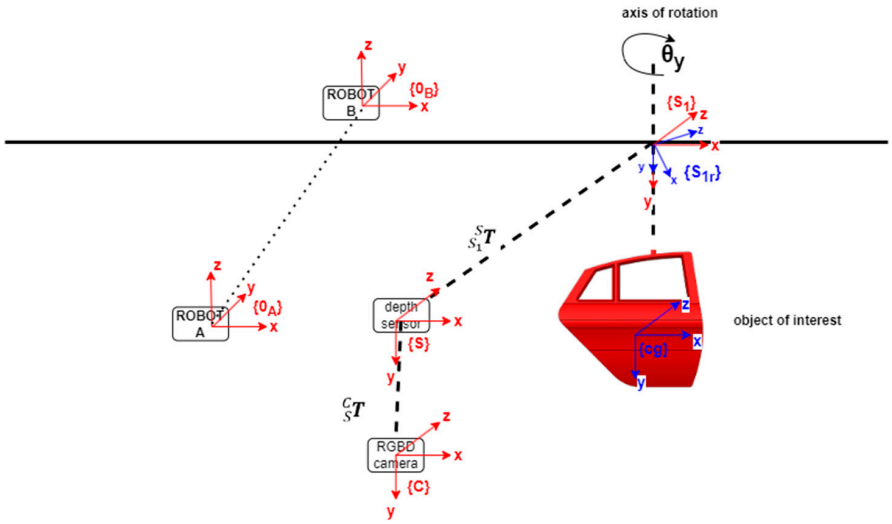


Figure 2. Schematic of rotating servomechanism for 3D scanning.

2.2. Optimal Trajectory Planner

Optimal trajectory planning for spray painting involves modeling coating deposition, robot dynamics, the spraying process, and an optimization scheme to obtain optimal paint parameters. A trajectory planner that uses a hybrid objective function to obtain optimal paint trajectories is considered [31]. This is our prior work, and the optimization variables include the slice width (δ), slice speeds ($v_1,\ v_2$), slicing direction (θ), and inverse kinematic

configuration (ik_{cf}) of the manipulator. The coating deposition employs a double-beta distribution model since it has higher practicality for HVLP (high-volume low-pressure) spray painting systems. It can be extended to other painting methods by modifying the coating distribution function. The optimizer starts by slicing the surface of the calibrated CAD (P^{C}_{CAD}) at an arbitrary slicing angle using randomly assigned optimization variables. Using the coating deposition and robot dynamic models, the mean robot energy, the mean trajectory time, the coating deviation error, and the mean squared coating error for a slice are computed. A genetic algorithm then changes the optimization variables through crossover and mutation until convergence is achieved. This process is repeated for all slices until the entire surface is covered. In this way, optimal trajectories are obtained for the surface. A summary of the hybrid optimization scheme is presented in Figure 3.

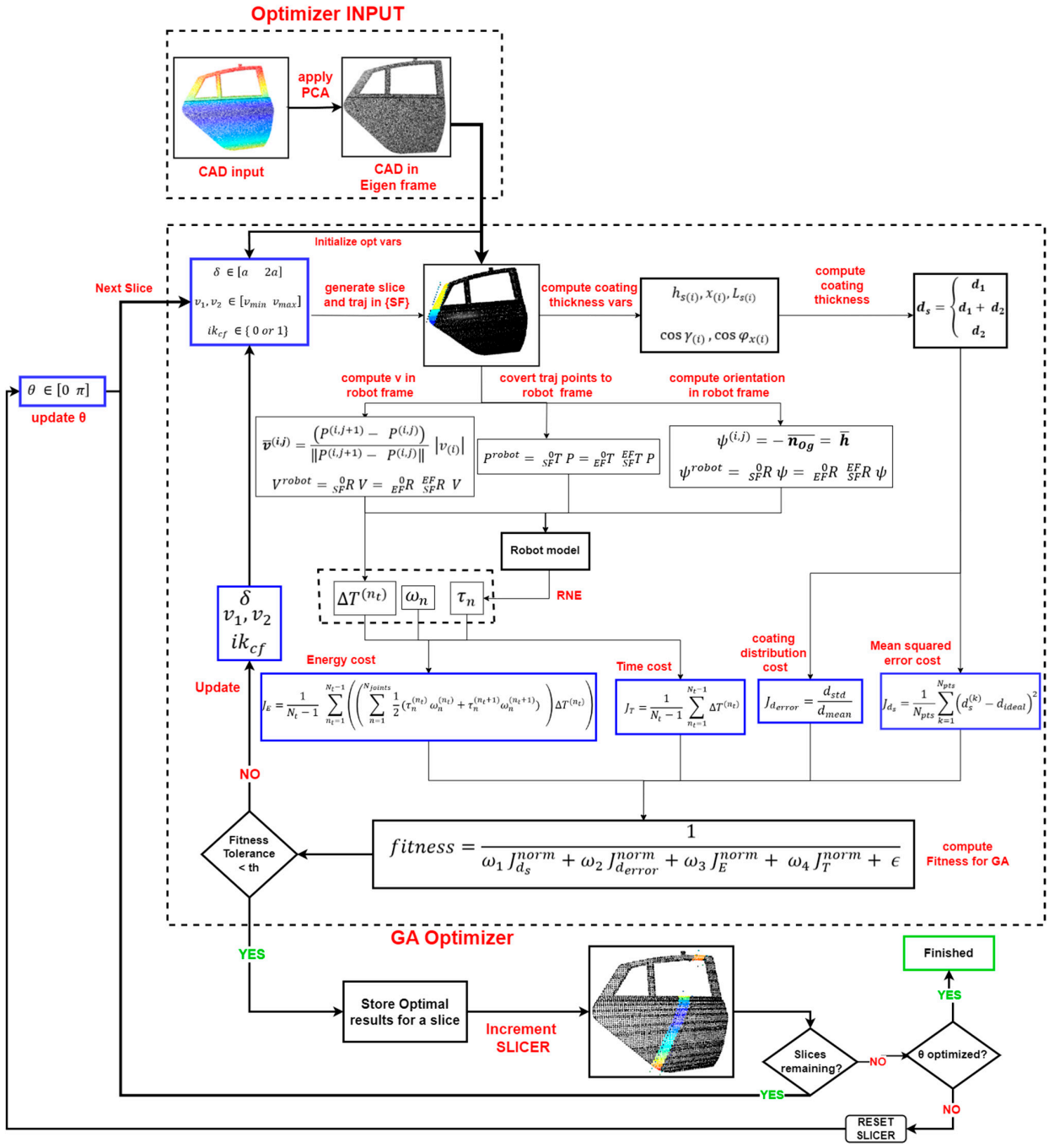
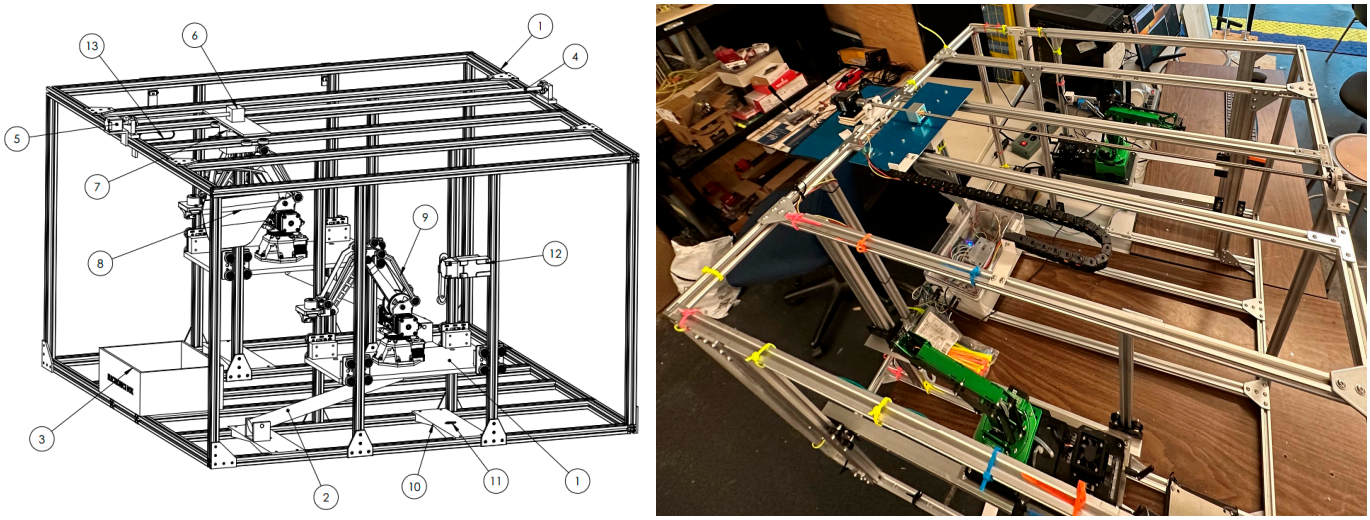


Figure 3. Summary of optimal trajectory planner for spray painting [31].

2.3. Integrated System

The integration of the designed 3D scanner and trajectory planner into a robotic system is intended to accomplish seamless automation. The enclosure, measuring 1000 by 1000 by 600 mm, primarily employs aluminum railings, as depicted in Figure 4. A horizontal slider, propelled by a stepper motor, facilitates the movement of the object of interest across the 3D scanner and the two installed robots. To enable 3D scanning (outlined in Section 2.1), a servomotor is affixed to the horizontal slider, allowing for the rotation of the object. Each of the two robots, mounted on vertical slides, possesses independent movement capabilities to cover all facets of the object. For safety measures, the vertical slides are regulated by linear actuators integrated with limit switches. TOF (Time of Flight) sensors, positioned on both the horizontal and vertical slides, serve to acquire accurate position feedback. Control over these components is managed through a Raspberry Pi controller equipped with appropriate motor drivers housed in the electronics box. Furthermore, an Arduino, connected to the Raspberry Pi, monitors the real-time power consumption of the robots using current sensors. For trajectory execution, two Jetmax three-DOF manipulators are employed due to their affordability. They are also compatible with the ROS (Robot Operating System) ecosystem, making their integration into the system straightforward. The trajectory planning scheme is not confined to this robot, as the planning is conducted in task space and not joint space. A comprehensive breakdown of each component is presented in Table 1. The dimensions of complete assembly are illustrated in Appendix A.



(**a**) (**b**)

Figure 4. (**a**) CAD model of ARSIP with component IDs; (**b**) laboratory setup of ARSIP.

Table 1. Breakdown of hardware components with component IDs.

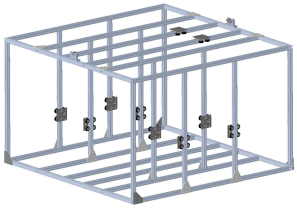
ID	Component Description	CAD Model
1	Aluminum framing used for building the structure of the entire system. It also contains corner brackets, T-brackets, and gantries for achieving smooth linear motion.	

Table 1. *Cont.*

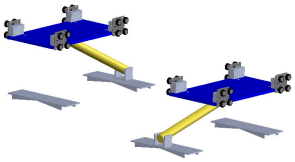
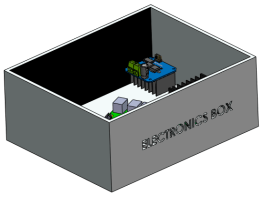
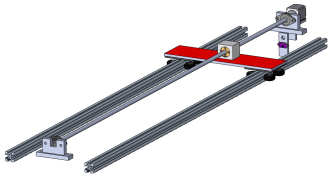
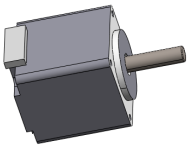
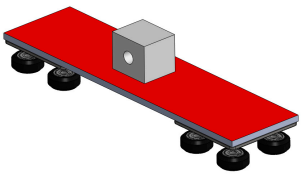
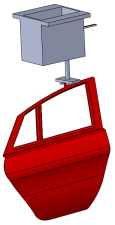
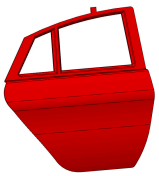
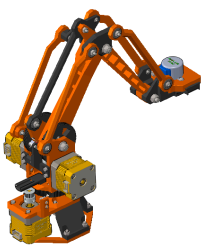
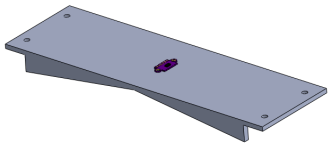
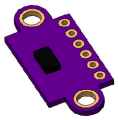
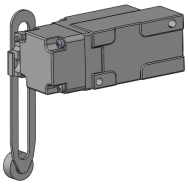
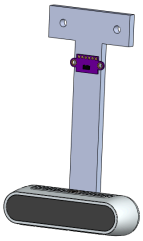
ID	Component Description	CAD Model
2	Vertical sliding mechanism. It contains two linear actuators, a support base for lifting the robots, and a base mount plate for securing the linear actuators. For position feedback, VL53L0X sensors are installed. These actuators are used for extended reach in case of large objects.	
3	Electronics box for keeping the electrical components. It contains a Raspberry Pi controller, an Arduino, two motor controllers for the linear actuators, current sensors, and a stepper driver. The choice of Raspberry Pi controller is due to its low cost and easy integration with the ROS ecosystem, while Arduino is used for its easy to access a DAC (Digital-to-Analog Convertor).	
4	Horizontal sliding mechanism. It contains a threaded rod, two guide rails with linear gantries, bearings and bearing supports for the threaded rod, a stepper motor for driving the mechanism, and a distance feedback sensor.	
5	Stepper motor [37] for moving the horizontal slider. This motor is a suitable choice for the system due to its easy availability, high torque for a low price, and convenient integration with the Raspberry Pi.	
6	Horizontal slider jockey. It contains a lead screw head connected to the base plate, which is then connected to the linear gantries for moving along the guide rails.	
7	Rotating servomechanism for 3D scanner. The servomechanism [33] is connected to the object via a gripper that can be tightened and loosened. The object is a car door, as illustrated in the CAD.	
8	A downscaled CAD model of a car door for 3D scanning and trajectory planning.	
9	Two 3-DOF Jetmax robots for controlling the x, y, and z locations of the end effector [38]. The robot combined with the vertical sliding mechanism gives a total of 4 DOF for executing the trajectory over the surface of a complex free-form surface (e.g., car door). The choice of a Jetmax robot is due to its low price and convenient integration with the ROS ecosystem.	

Table 1. *Cont.*

ID	Component Description	CAD Model
10	Position feedback platform for the linear actuators with distance feedback sensor.	
11	VL53L0X sensor [39]. It is a time-of-flight (TOF) sensor for measuring distance. It has a measurement range of 3 cm to 2 m and an accuracy of ±1 mm. These sensors are more accurate and precise than SONAR (Sound Navigation and Ranging) sensors, which makes them a good choice for the integrated system.	
12	A limit switch used for disconnecting the power from the linear actuators [40].	
13	3D scanning hardware, including an Intel Real Sense D435 sensor [32] and a VL53L0X sensor for axis calibration. These sensors are low-cost compared to laser scanners, which makes them a suitable choice for the integrated system.	

3. Software Development

Once the CAD modeling and fabrication of the system are complete, it becomes crucial to design a software mechanism that effectively fulfills the autonomous system's intended functions. This software needs to establish seamless communication with the hardware components and adeptly optimize trajectories across intricate, free-form surfaces. Additionally, the incorporation of a user-friendly GUI becomes pivotal, enabling users to interact with the system, adjust settings, and import CAD and trajectory files. Section 3 elaborates on the software development process involved in achieving these objectives.

3.1. Software Framework

Figure 5 illustrates the software breakdown and the GUI. The system's core components interconnect through the ROS ecosystem [41], facilitating smooth communication among software scripts/nodes via topics and services. Within this ecosystem, the ROS MASTER serves as the primary server, hosting essential nodes responsible for trajectory optimization and 3D scanning. Meanwhile, another instance of ROS operates on the Raspberry Pi controller, directly linked to the hardware. This hardware encompasses horizontal and vertical sliding mechanisms, servomechanism, VLX sensors for distance monitoring, and the D435 sensor for depth scanning. The Raspberry Pi is programmed to respond to specific topics through ROS subscribers and publishers, allowing the MASTER node to access any sensor data via subscription and effect real-time changes in actuator states by publishing to the Raspberry Pi node.

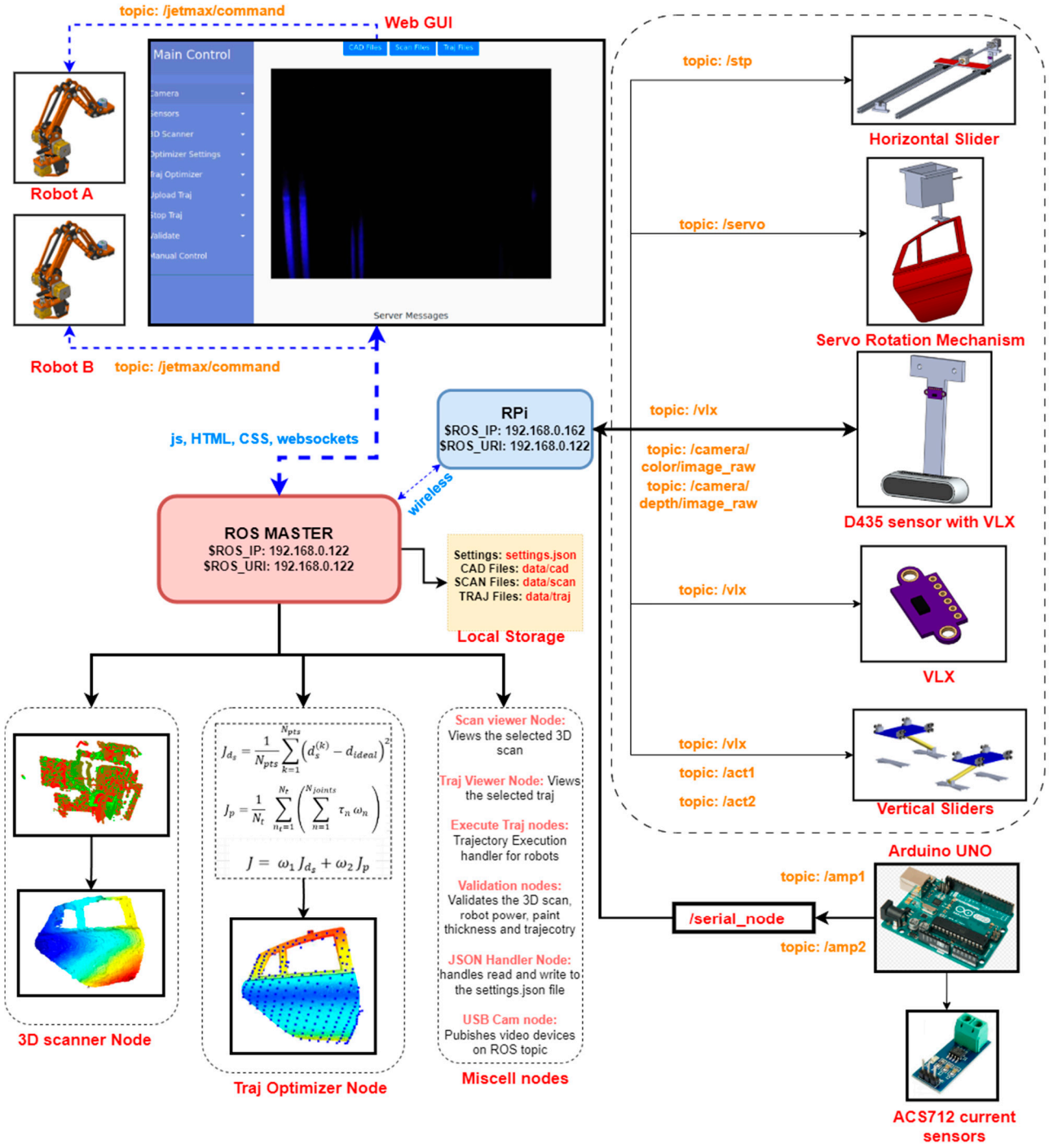


Figure 5. Schematic of the software framework for ARSIP.

The web-based GUI connects to the ROS MASTER and the two robots via web sockets programmed in JS (JavaScript) [42]. HTML scripts form the web page's foundation, while CSS and JS elements from Bootstrap and jQuery [43] dictate its style. The integration of the ROS ecosystem with JS is enabled through the ROSLIBJS script developed by [44]. Trajectory commands, presented as x, y, and z coordinates and a time-variable defining speed, are dispatched to the two robots. Upon user request through the web GUI, the 3D scanner node initiates a 3D scanning process, saving the scan file locally in the scan directory upon completion. Similarly, the trajectory optimizer node, triggered from the GUI, performs trajectory optimization and stores results locally as a NumPy array in the *traj* folder.

Likewise, various nodes fulfill specific functions within the system: CAD calibration, the validation of 3D scans, viewing trajectories, and managing process settings. A JSON handler node is dedicated to storing and updating these process settings in a JSON file. This node actively listens to a string topic and promptly updates the fields upon user requests from the web page. Additionally, a USBCAM node is responsible for capturing video feeds from a USB device and publishing them on a ROS topic, enabling real-time monitoring via the webpage. In a separate setup, the Raspberry Pi is programmed to receive messages from the ROS MASTER, controlling motors and actuators and publishing readings from the VLX sensors and image streams captured by the RealSense D435 sensor. The Arduino facilitates serial communication, transmitting current readings to the ROS ecosystem efficiently.

3.2. Graphical User Interface

The web-based GUI serves as an interface, facilitating communication between users and the system's software components. Its primary objective is to offer a user-friendly platform for executing diverse system functions. This GUI incorporates a sidebar navigation menu housing interactive buttons, such as Camera, Sensors, 3D Scanner, Optimizer Settings, *Traj* Optimizer, Upload *Traj*, Stop *Traj*, Validate, and Manual Control, enabling seamless interaction with the system. Additionally, the top navigation bar contains four buttons—CAD Files, Scan Files, CAD-CAL files, and *Traj* Files—managed by the JavaScript node. Clicking these buttons triggers the execution of the respective functionalities. Figure 6 depicts the main page of the graphical user interface.

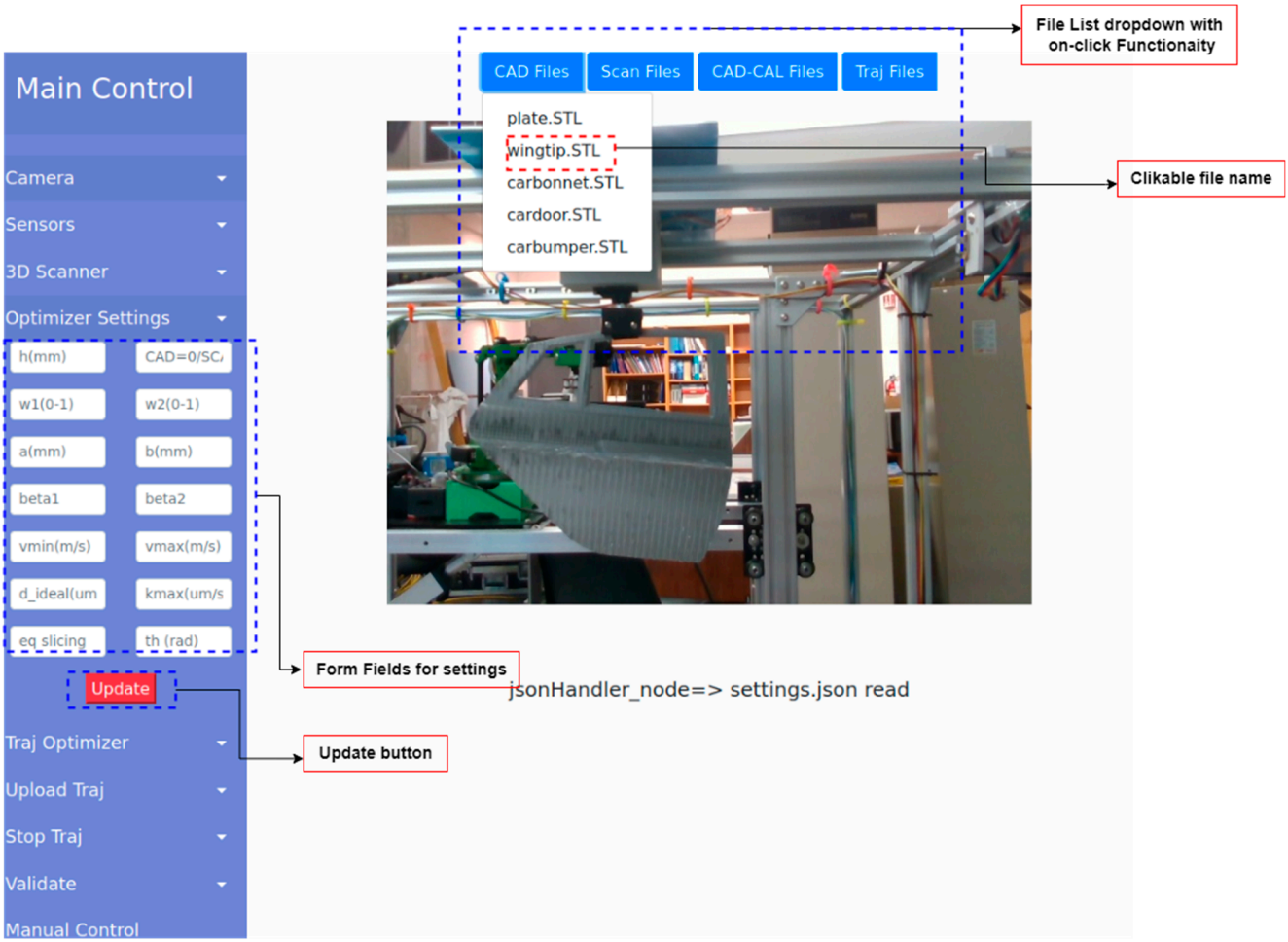


Figure 6. Graphical user interface for ARSIP.

The additional functionalities embedded within the GUI are illustrated in Figure 7. The *Traj* Optimizer button is subdivided into two distinct buttons, one designated for optimizing the trajectory and the other for visualizing the chosen trajectory. To initiate trajectory optimization, the optimizer settings require the CAD/SCAN field to be set to 0 or 1, representing CAD or SCAN, respectively. The backend utilizes the selected CAD or SCAN file to plan the trajectory accordingly. Similarly, the Upload and Stop *Traj* buttons activate the trajectory handler node, enabling the commencement or cancellation of trajectory uploads to the robots at any given point. The Validate button encompasses three subfields for assessing the 3D scan accuracy concerning the chosen CAD and SCAN files, energy evaluation, and paint quality pertaining to the selected trajectory file. Additionally, Figure 8 showcases the Camera field dropdown, facilitating the selection of a specific camera feed, the Sensors dropdown for monitoring sensor readings, and the 3D scanner button to initiate and observe the 3D scanning process.

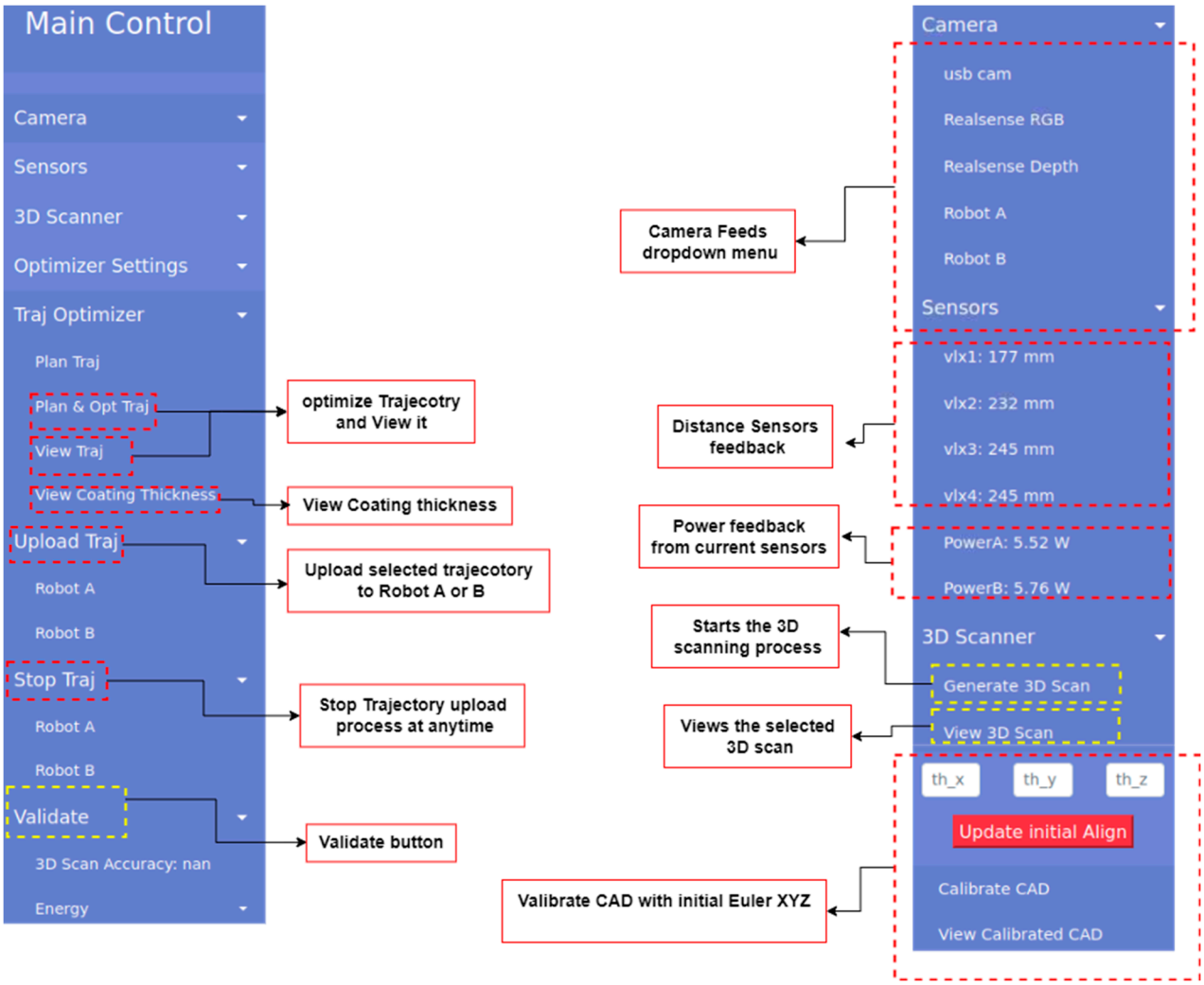


Figure 7. Breakdown of submenus of graphical user interface.

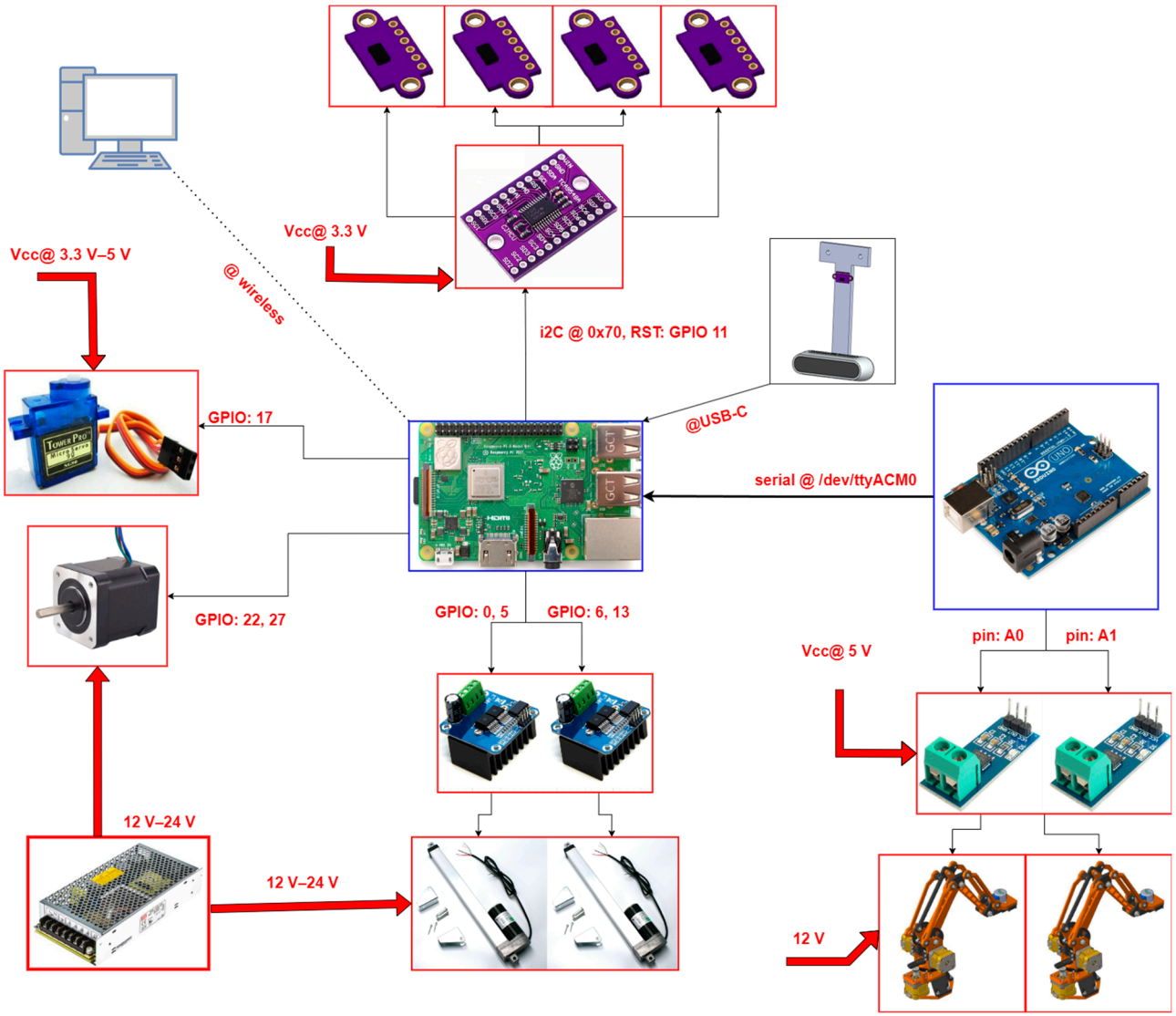


Figure 8. Circuit connection diagram of ARSIP.

3.3. Circuit Connection Diagram

The integration of hardware and software is pivotal in realizing the intended objective of the ARSIP. This integration involves connecting sensors, motors, and actuators to the Raspberry Pi controller via GPIO pins, while the ACS712 current sensors interface with Arduino's analog pins. The motor drivers facilitate motor operation as the microcontroller alone lacks sufficient power output. After establishing a wireless connection, the Raspberry Pi links to the LINUX server through the ROS ecosystem. Furthermore, the VLX sensors undergo multiplexing via a TCA9548 and are then connected to the Raspberry Pi's i2C port. A detailed depiction of the circuit connections, inclusive of pin numbering, is presented in Figure 8.

4. Results and Discussions

The software framework is coded in Python [45], coupled with the ROS ecosystem. Achieving the best trajectory planning involves several steps: conducting a 3D scan of the object, calibrating the CAD within the camera's reference frame, and optimizing the objective functions detailed in Section 2. This section will cover and analyze the processes of 3D scanning and object calibration, as well as delving into the resulting optimal trajectories within the integrated system. Three objects are considered for the analysis, including a car

door, a hood, and a bumper. For reference, the fixed parameters utilized in the analysis can be found in Table 2.

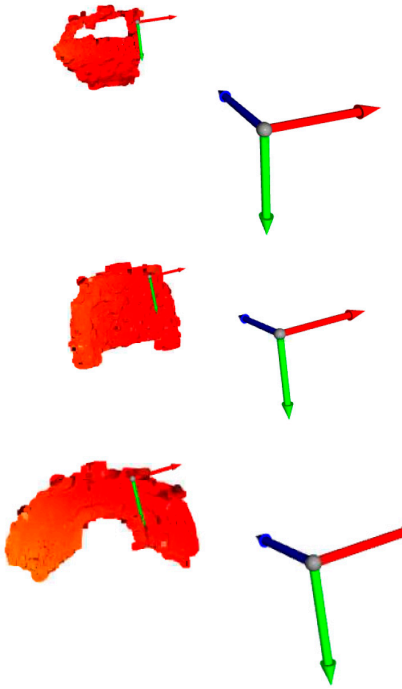
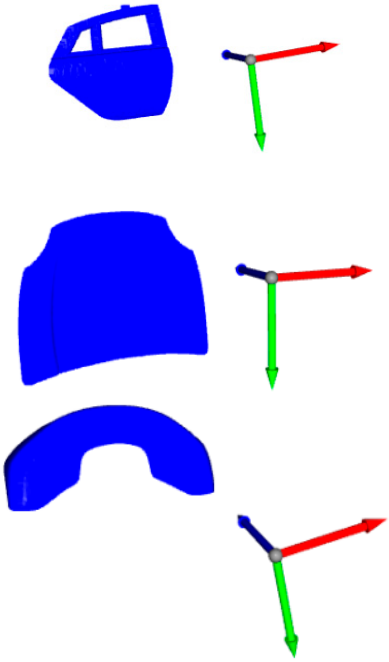
Table 2. Constant parameters used in the results analysis.

Parameter	Description	Value
	Spraying process parameters	
a	Ellipse longer side for the coating model	15 mm
b	Ellipse shorter side for the coating model	5.6 mm
β_x	Coating distribution beta along the X direction of ellipse	2.3
β_y	Coating distribution beta along the Y direction of ellipse	4.5
k_{max}	Coating deposition rate	50.0 µm/s
d_{ideal}	Desired coating thickness	20 µm
v_{min}	Minimum speed of the spray gun	3 mm/s
v_{max}	Maximum speed of the spray gun	15 mm/s
h	Spray gun height from the surface	10 mm
	Robot model parameters	
d_{stroke}	Link 0 stroke length	254 mm
L_1	Manipulator Link 1 length	92.54 mm
L_2	Manipulator Link 2 length	128.4 mm
L_3	Manipulator Link 3 length	144.8 mm
M	Manipulator Link 0 mass	2.5 kg
m_1	Manipulator Link 1 mass	0.5 kg
m_2	Manipulator Link 2 mass	0.5 kg
m_3	Manipulator Link 3 mass	0.5 kg
	Optimizer Parameters	
ω_1	Scaling factor for mean-squared error	0.40
ω_2	Scaling factor for coating deviation error	0.20
ω_3	Scaling factor for mean energy consumption	0.20
ω_4	Scaling factor for mean trajectory time	0.20
ϵ	Hyper-parameter in the fitness function	1.0
r_m	Mutation rate in GA	0.1
c_{type}	Crossover type in GA	Two points
m_{type}	Mutation type in GA	Random
$N_{parents}$	Number of mating parents in GA	2
N_{gen}	Number of generations in GA	25
N_{sol}	Number of solutions per population in GA	2

4.1. 3D Scanning and CAD Calibration

The 3D scanning system captures surface point clouds of the investigated object. These point clouds, acquired at each rotational index (30°), undergo filtering and initial alignment using previously defined transformations from Section 2.1. Further alignment is achieved using the ICP to merge the point clouds into a unified reference frame. Additionally, statistical noise removal is applied if any residual noise persists. The calibration involves scanning these objects and aligning their CAD models with the camera's reference frame, {C}, for trajectory planning. Scanning accuracy is assessed using the D_1 and D_2 metrics [46]. Table 3 provides an overview of the scanned models and their calibrated CAD representations within the camera frame, {C}. When compared with their CAD counterparts, the scanned models unveil 95% accuracy for the car door, 93% for the car hood, and 92% for the car bumper, as tabulated in Table 4. In Figure 9, the process of selecting and viewing a scan file through the GUI is demonstrated, while Figure 10 depicts the calibration procedure within the system's GUI interface.

Table 3. 3D scanned geometries and their corresponding calibrated CAD models.

Scanned Geometry in Frame {C}	Calibrated CAD in Frame {C}
	

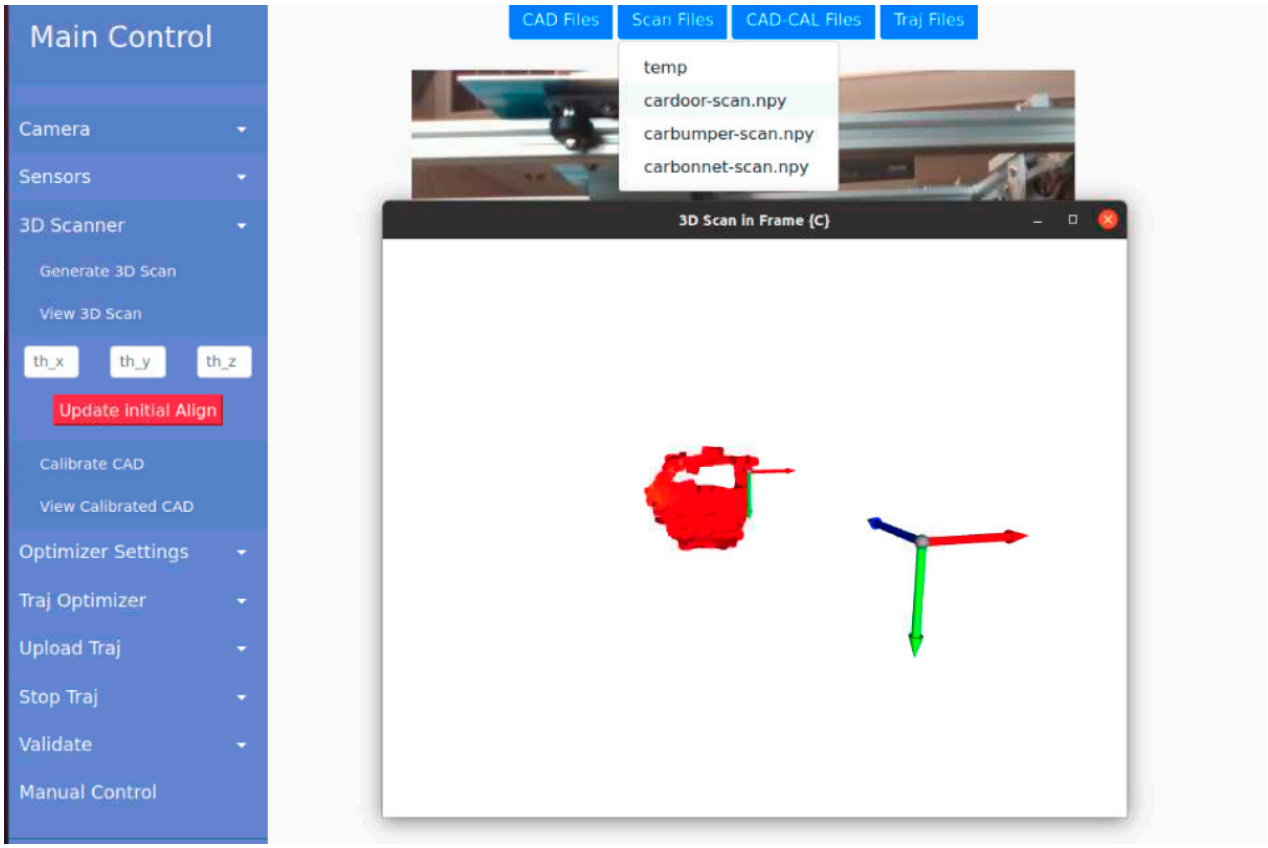


Figure 9. Viewing a 3D scan in camera frame via the GUI.

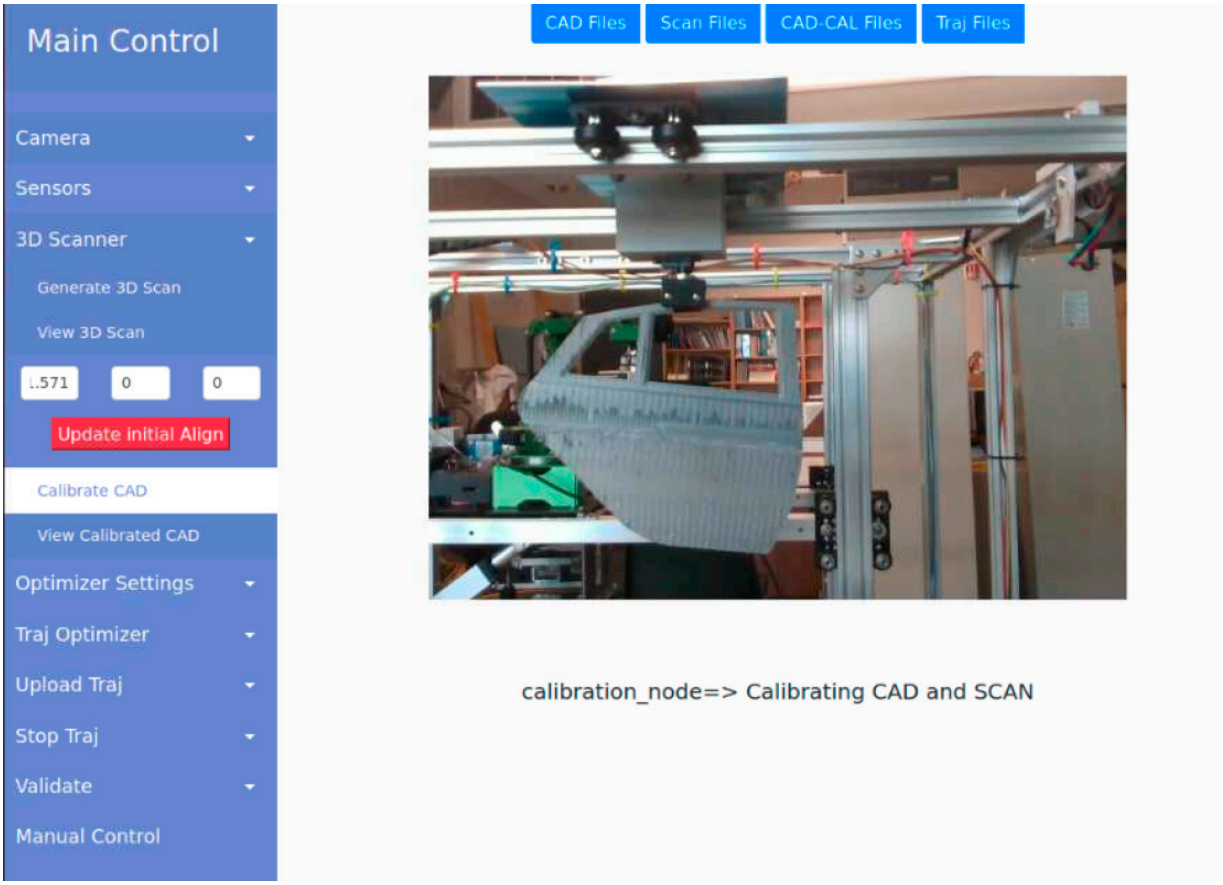


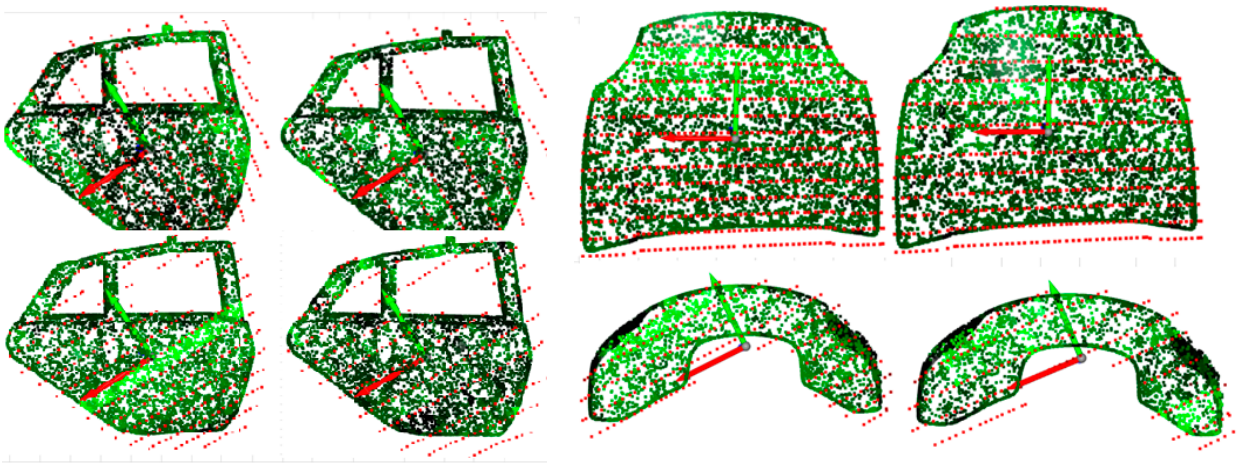
Figure 10. CAD calibration using the GUI.

Table 4. Similarity scores of the scanned and CAD models.

	D_1 Score	D_2 Score	Avg Score
Car door	0.9639	0.9434	0.9536
Car hood	0.9524	0.9228	0.9376
Car bumper	0.9488	0.9082	0.9285

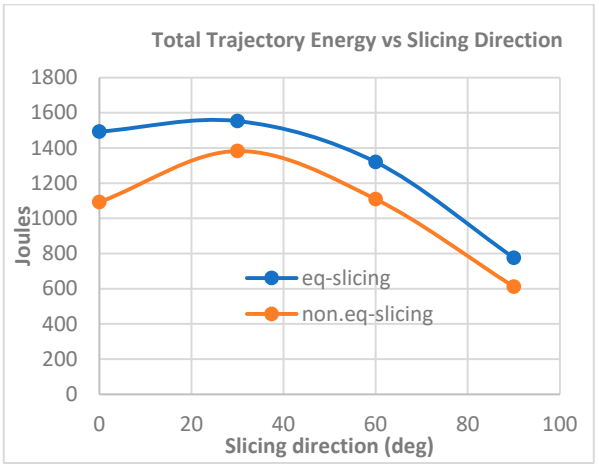
4.2. Optimal Trajectory Planning

After the CAD model is aligned with the scanned one to ensure accurate positioning in both the camera and robot frames, the optimization algorithm is applied to generate efficient trajectories for the painting process. The results reveal that achieving the desired coating thickness depends on the spray parameters and robot model. The coating thickness varies due to an object's geometry, the paint gun speed, the slice width, and the slicing direction. Equidistant slicing offers more uniformity, but non-equidistant slicing is more energy- and time-efficient, covering surfaces with fewer slices. For a car door, the energy efficiency peaks at 90° with non-equidistant slicing, 60% less than the least efficient. The optimal time occurs at 30° with non-equidistant slicing, 33% faster than the slowest. Equidistant slicing at 30° achieves a close thickness (19.21 µm), while non-equidistant slicing at the same angle minimizes the coating error (14%) but yields a slightly thinner result (18.38 µm). Figure 11 shows a few of the planned trajectories for three objects, while Figure 12 displays the energy consumption, time, coating deviation, and relative coating error for each slicing direction for a car door. The validity of the trajectory of energy consumption is confirmed by measuring the actual energy used by the robots in real time. The total energy values across all trajectory points are aggregated and then compared against the experimental observations, detailed in Table 5, for comparison.

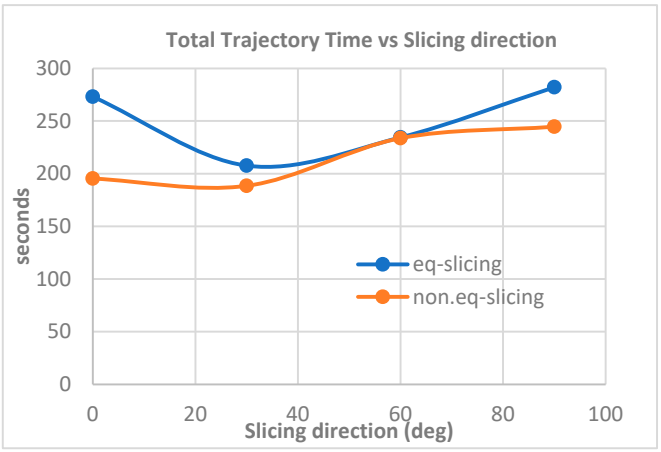


(**a**) (**b**)

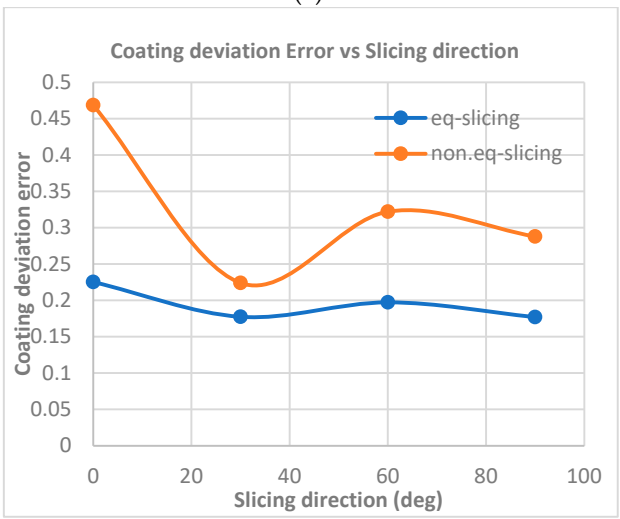
Figure 11. (**a**) Planned trajectories for a car door at 0° and 90° slicing directions; (**b**) planned trajectories for a car hood and bumper at a 90° slicing direction. Bright green color shows high coating thickness values, and vice versa.



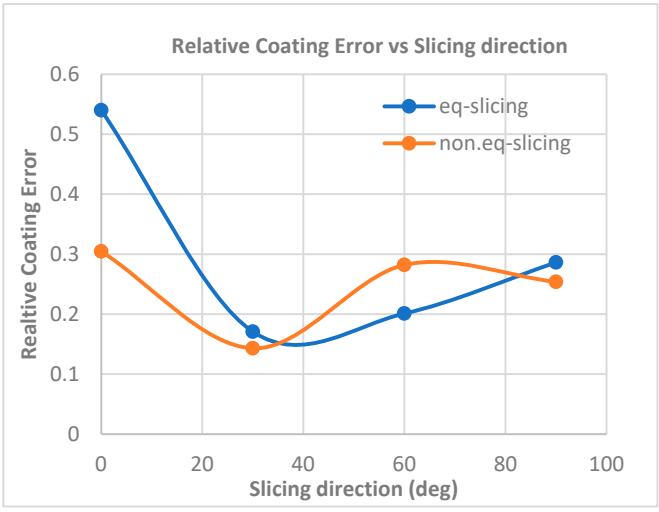
(**a**)



(**b**)



(**c**)



(**d**)

Figure 12. (**a**) Total energy; (**b**) trajectory time; (**c**) coating deviation; (**d**) relative coating error vs. slicing direction for a car door.

Table 5. Experimental validation of energy consumption for selected trajectories.

	Slicing Scheme	θ	Experimental E_{sum}	Theoretical E_{sav}	Experimental E_{sav}
Car door	Non-equidistant	90°	2085 J	60%	44%
	Equidistant (ref.)	30°	3003 J	0%	0%
Car hood	Non-equidistant	90°	1569 J	73%	51%
	Equidistant (ref.)	0°	3212 J	0%	0%
Car bumper	Non-equidistant	90°	1275 J	64%	33%
	Equidistant (ref.)	0°	1894 J	0%	0%

4.3. Comparison with State of the-Art

The coating uniformity, trajectory time, and energy consumption in paint applications depend on the object's geometry, the dynamics of the spraying process, and the robot executing the trajectory. These factors will vary based on the scenario presented in the optimization process. The analysis of the results leads to the conclusion that the proposed integrated system can generate efficient trajectories for a painting process. It not only achieves coating uniformity, as indicated by low coating deviation and relative coating errors, but also optimizes the energy consumption and trajectory time of the manipulator, thereby enhancing the overall efficiency of the painting process. To the best of our knowledge, the proposed system is the first of its kind, as the previous literature shows no optimization with respect to robot energy and trajectory time. A summary of the results in comparison with existing solutions is tabulated in Table 6.

Table 6. Results comparison summary with state of the art.

Article	U-Direction [19]	V-Direction [19]	Equidistant Slicing [20]	Non. Eq Slicing [20]	Transitional-Seg Opt [21]	Proposed Scheme	Proposed Scheme	Proposed Scheme
Object of interest	Oval Bucket	Oval Bucket	Motorcycle spoiler	Motorcycle spoiler	Aircraft wing	Car door	Car hood	Car bumper
Desired coating thickness	50 µm	50 µm	23 µm	23 µm	70 µm	20 µm	20 µm	20 µm
Mean coating thickness	51.1 µm	52.2 µm	25.95 µm	22.27 µm	68.7 µm	19.21 µm	21.02 µm	21.02 µm
Standard deviation	2.775 µm	3.8 µm	6.52 µm	5.71 µm	3.2 µm	3.41 µm	6.51 µm	5.35 µm
Mean coating deviation error	5.43%	7.60%	25.12%	25.64%	4.60%	17.75%	29.4%	25.4%
Mean relative coating error	2.2%	4.4%	12.83%	3.17%	1.86%	3.95%	5.1%	5.1%
Max time savings	17%	0%	N/A	N/A	N/A	33%	14.18%	27.13%
Max energy savings	N/A	N/A	N/A	N/A	N/A	60%	73%	64%
Coating mean-squared error cost	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Coating deviation cost	No	No	No	No	No	Yes	Yes	Yes
Energy cost	No	No	No	No	No	Yes	Yes	Yes
Process time cost	No	No	No	No	No	Yes	Yes	Yes

5. Conclusions and Future Work

This manuscript outlines the design and development of an automated robotic system for industrial painting. A 3D scanner, employing a turntable mechanism and a low-cost RGBD sensor, was designed. Additionally, a calibration mechanism was devised to represent the CAD models in the camera frame, addressing incomplete point clouds. The integration of an optimal trajectory planner into the system enhanced the efficiency of the painting trajectories. Moreover, this study detailed the meticulous hardware–software design of the ARSIP using open-source software: ROS and Python. A web-based graphical interface was also developed to enhance user interaction with the system. The findings

ascertain that the integrated hardware–software system proposed can effectively automate trajectory planning and execution, leading to energy-, time-, and coating-efficient spray processes. The following are the key characteristics of the ARSIP:

- A low-cost 3D scanner and calibrator that capture complex free-form surfaces with accuracies of up to 95%.
- An efficient trajectory planning scheme with energy savings of up to 73% and time savings of up to 33%.
- A trajectory planner capable of achieving optimal coating quality with relative coating errors as low as 5% and deviation errors as low as 17%.
- An easily scalable autonomous hardware–software framework using open-source software such as ROS and Python.
- An interactive web-based graphical user interface providing user control over the system and real-time monitoring of camera feeds, power consumption, and sensor states.

For future work, simulating the paint system using tools like Gazebo or MATLAB is suggested to analyze optimal paint paths. Gazebo, requiring URDF for precise robot modeling within an ROS, facilitates system evaluation on physical setups. Utilizing six-axis robots with HVLP spray guns is advised for accurate paint application, compensating for the orientation constraints of three-DOF robots. Moreover, the optimization scheme can be extended to other types of spray-painting methods by modifying the coating deposition function. For large-scale industrial use, a hybrid optimization approach in Python, parallelization via hyperthreading, GPU processing for batch tasks, and Docker containers for seamless software distribution are recommended.

6. Patents

The ARSIP, in contribution with Cherkam Industrial Ltd., is undergoing the process of patent filing.

Author Contributions: H.A.G. is the principal investigator. M.I. conducted the research, design, and development process of ARSIP. M.I. wrote the main draft of the paper, and H.A.G. reviewed and accepted the text. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Mitacs: 215488.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All relevant data are within the manuscript.

Acknowledgments: The proposed integrated system is a part of my thesis study titled “Development of an Automated Industrial Painting System with Optimized Quality and Energy Consumption”. Special thanks to the program supervisor, Hossam Gaber, for supporting the thesis and project. Gratitude to Mitacs Accelerate and Cherkam Industrial Ltd. (an industrial partner) for sponsoring and funding the project.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

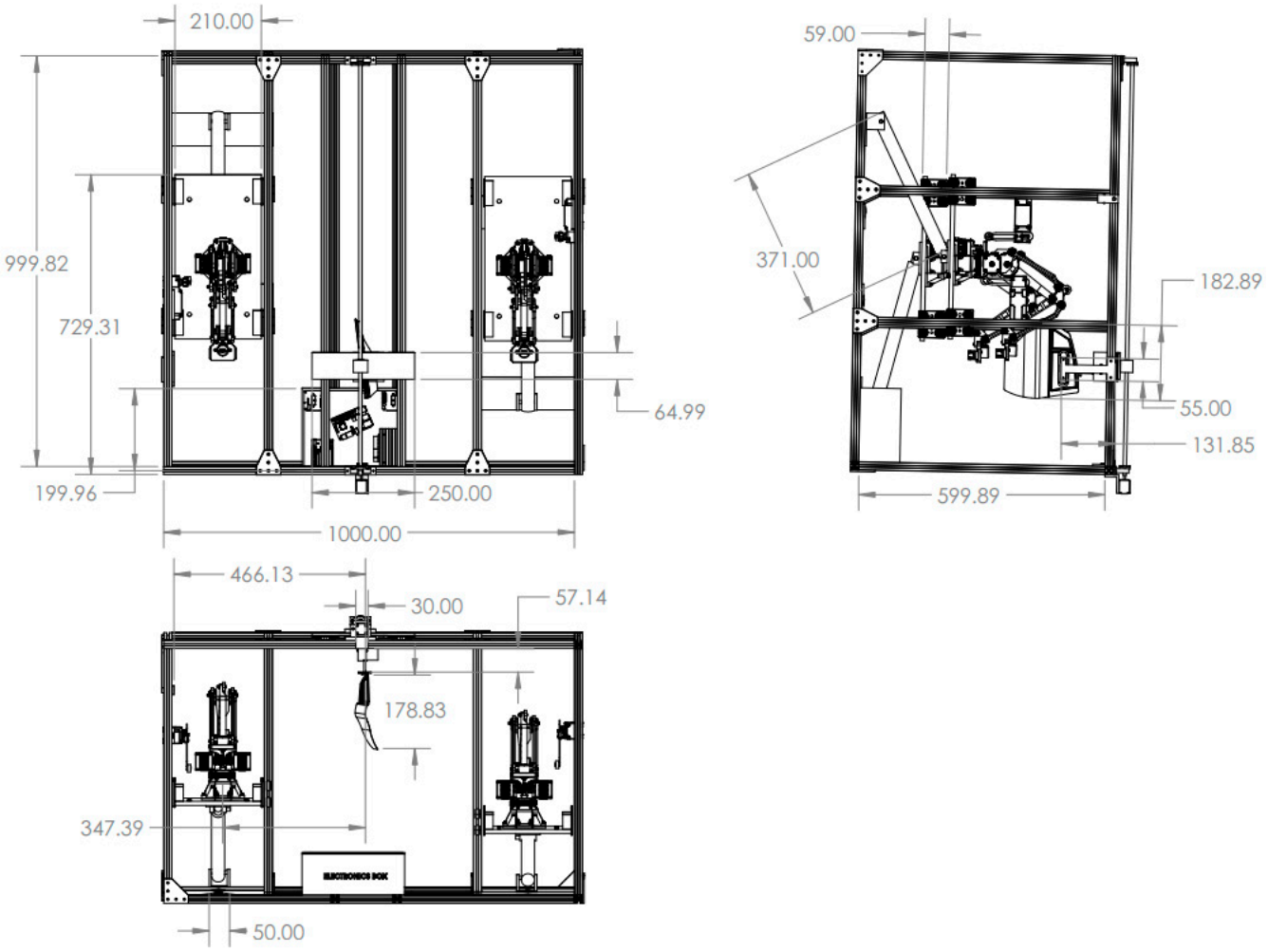


Figure A1. CAD drawings of ARSIP.

References

1. KPMG. *Global Automotive Executive Survey*; Technical Report; KPMG: Amstelveen, The Netherlands, 2017.
2. United States Environmental Protection Agency. Volatile Organic Compounds' Impact on Indoor Air Quality. Available online: https://www.epa.gov/indoor-air-quality-iaq/volatile-organic-compounds-impact-indoor-air-quality#:~:text=Health%20effects%20may%20include:,kidney%20and%20central%20nervous%20system (accessed on 1 November 2023).
3. Qlayers. Gone to Waste: Exploring the Environmental Consequences of Industrial Paint Pollution. Available online: https://www.qlayers.com/blog/gone-to-waste-exploring-the-environmental-consequences-of-industrial-paint-pollution (accessed on 1 November 2023).
4. Bambousek, G.J.; Bartlett, D.S.; Schmidt, T.D. Spray Paint System including Paint Booth, Paint Robot Apparatus Movable Theren and Rail Mechanism for Supporting the Apparatus Thereout. U.S. Patent 4,630,567, 23 December 1986.
5. Suh, S.-S.; Lee, J.-J.; Choi, Y.-J.; Lee, S.-K. A Prototype Integrated Robotic Painting System: Software and Hardware Development. In Proceedings of the 1993 IEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan, 26–30 July 1993.
6. Arıkan, M.A.S.; Balkan, T. Process Modeling, Simulation, and Paint Thickness Measurement for Robotic Spray Painting. *J. Field Robot.* **2000**, *17*, 479–494.
7. Javaid, M.; Haleem, A.; Pratap, S.; Suman, R. Industrial perspectives of 3D scanning: Features, roles and it's analytical applications. *Sens. Int.* **2021**, *2*, 100114. [CrossRef]
8. Du, L.; Lai, Y.; Luo, C.; Zhang, Y.; Zheng, J.; Ge, X.; Liu, Y. E-quality Control in Dental Metal Additive Manufacturing Inspection Using 3D Scanning and 3D Measurement. *Front. Bioeng. Biotechnol.* **2020**, *8*, 1038. [CrossRef]
9. Dombroski, C.E.; Balsdon, M.E.R.; Froats, A. The use of a low cost 3D scanning and printing tool in the manufacture of custom-made foot orthoses: A preliminary study. *BMC Res. Notes* **2014**, *7*, 443. [CrossRef]
10. Yao, A.W.L. Applications of 3D scanning and reverse engineering techniques for quality control of quick response products. *Int. J. Adv. Manuf. Technol.* **2005**, *26*, 1284–1288. [CrossRef]

11. Newcombe, R.A.; Izadi, S.; Hilliges, O.; Molyneaux, D. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality, Basel, Switzerland, 26–29 October 2011.
12. Meister, S.; Izadi, S.; Kohli, P.; Hammerle, M.; Rother, C.; Kondermann, D. When Can We Use KinectFusion for Ground Truth Acquisition? In Proceedings of the Workshop on Color-Depth Camera Fusion in Robotics, Daejeon, Repubic of Korea, 5–9 November 2012.
13. Ma, F.; Carlone, L.; Ayaz, U.; Karaman, S. Sparse depth sensing for resource-constrained robots. *Int. J. Robot. Res.* **2019**, *38*, 935–980. [CrossRef]
14. Larsson, S.; Kjellander, J.A. Motion control and data capturing for laser scanning with an industrial robot. *Robot. Auton. Syst.* **2006**, *54*, 453–460. [CrossRef]
15. Borangiu, T.; Dumitrache, A. Robot Arms with 3D Vision Capabilities. In *Advances in Robot Manipulators*; IntechOpen: Rijeka, Croatia, 2010.
16. Li, J.; Chen, M.; Jin, X.; Chen, Y.; Dai, Z.; Ou, Z.; Tang, Q. Calibration of a multiple axes 3-D laser scanning system consisting of robot, portable laser scanner and turntable. *Optik* **2011**, *122*, 324–329. [CrossRef]
17. Pichler, A.; Viiicze, H.; Andersen, H.; Hladseii, O. A Method for Automatic Spray Painting of Unknown Parts. In Proceedings of the International Conference on Robotics & Automation, Washington, DC, USA, 11–15 May 2002.
18. Andulkar, M.V.; Chiddarwar, S.S. Incremental approach for trajectory generation of spray painting robot. *Ind. Robot. Int. J.* **2015**, *42*, 228–241. [CrossRef]
19. Chen, W.; Tang, Y.; Zhao, Q. A novel trajectory planning scheme for spray painting robot with Bézier curves. In Proceedings of the Chinese Control and Decision Conference (CCDC), Yinchuan, China, 28–30 May 2016; pp. 6746–6750.
20. Yu, X.; Cheng, Z.; Zhang, Y.; Ou, L. Point cloud modeling and slicing algorithm for trajectory planning of spray painting robot. *Robotica* **2021**, *39*, 2246–2267. [CrossRef]
21. Guan, L.; Chen, L. Trajectory planning method based on transitional segment optimization of spray transitional segment optimization of spray. *Ind. Robot. Int. J. Robot. Res. Appl.* **2019**, *46*, 31–43. [CrossRef]
22. KUKA. KUKA Ready2_Spray. Available online: https://pdf.directindustry.com/pdf/kuka-ag/kuka-ready2-spray/17587-748199.html (accessed on 5 November 2023).
23. FANUC. FANUC P-250iB Paint Robot. Available online: https://www.fanucamerica.com/products/robots/series/paint/p-250ib-paint-robot (accessed on 5 November 2023).
24. ABB. IRB 5500-22/23. Available online: https://new.abb.com/products/robotics/industrial-robots/irb-5500-22 (accessed on 5 November 2023).
25. RoboDK. Simulate Robot Applications. Available online: https://robodk.com/examples%7B#%7Dexamples-painting (accessed on 5 November 2023).
26. Robotic and Automated Workcell Simulation, Validation and Offline Programming. Available online: www.geoplm.com/knowledge-base-resources/GEOPLM-Siemens-PLM-Tecnomatix-Robcad.pdf (accessed on 11 November 2023).
27. Delfoi. Delfoi PAINT. Available online: https://www.delfoi.com/delfoi-robotics/delfoi-paint/ (accessed on 11 November 2023).
28. ABB. RobotStudio® Painting PowerPac. Available online: https://new.abb.com/products/robotics/application-software/painting-software/robotstudio-painting-powerpac (accessed on 12 November 2023).
29. Inropa™ OLP Automatic. Available online: https://www.inropa.com/fileadmin/Arkiv/Dokumenter/Produktblade/OLP_automatic.pdf (accessed on 14 November 2023).
30. FANUC. FANUC ROBOGUIDE PaintPro. Available online: https://www.fanucamerica.com/support/training/robot/elearn/fanuc-roboguide-paintpro (accessed on 14 November 2023).
31. Idrees, M.; Gabbar, H.A. A hybrid optimization scheme for efficient trajectory planning of a spray-painting robot. In Proceedings of the 3rd International Conference on Robotics, Automation, and Artificial Intelligence (RAAI), Singapore, 14–16 December 2023.
32. IntelRealSense. Depth Camera D435. Available online: https://www.intelrealsense.com/depth-camera-d435/ (accessed on 15 November 2023).
33. TowerPro. SG90 Digital. Available online: https://www.towerpro.com.tw/product/sg90-7/ (accessed on 15 November 2023).
34. Davies, E.R. *Machine Vision Theory, Algorithms, Practicalities*; Elsevier: San Francisco, CA, USA, 2005.
35. Zhou, Q.-Y.; Park, J.; Koltun, V. Open3D: A Modern Library for 3D Data Processing. In Proceedings of the Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018.
36. Besl, P.; McKay, N.D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 239–256. [CrossRef]
37. Amazon. Nema 23 Stepper Motor Bipolar 1.8 Degree 2.8A. Available online: https://www.amazon.com/JoyNano-Nema-23-Stepper-Motor/dp/B07H866S2F?th=1 (accessed on 18 November 2023).
38. Hiwonder. Jetmax Jetson Nano. Available online: https://www.hiwonder.com/products/jetmax?variant=39645677125719 (accessed on 18 November 2023).
39. ESPHome. VL53L0X Time of Flight Distance Sensor. Available online: https://esphome.io/components/sensor/vl53l0x.html (accessed on 18 November 2023).
40. Amazon. Electrical Buddy Adjustable Rod Lever Arm Momentary Limit Switch. Available online: https://www.amazon.ca/Electrical-Buddy-Adjustable-Momentary-Me-8107/dp/B07Y7C9188 (accessed on 19 November 2023).
41. Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; Ng, A. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009.

42. JavaScript. Pluralsight. Available online: https://www.javascript.com/ (accessed on 20 November 2023).
43. Bootstrap. Available online: https://getbootstrap.com/docs/4.2/getting-started/introduction/ (accessed on 21 November 2023).
44. The Standard ROS JavaScript Library. ROS.org. Available online: https://wiki.ros.org/roslibjs (accessed on 21 November 2023).
45. van Rossum, G.V. Python. Available online: https://www.python.org/ (accessed on 24 November 2023).
46. Osada, R.; Funkhouser, T.; Chazelle, B.; Dobkin, D. Matching 3D models with shape distributions. In Proceedings of the International Conference on Shape Modeling and Applications, Genova, Italy, 7–11 May 2001; pp. 154–166.