Contour-Following Adhesive Cobot.

Mr.HARI THOTA ME Electrical
Scientist D
Smart Manufacturing, IIOT and AI group
Center for Smart Manufacturing Precession Machine Tools & aggregates(SMPM),
Central Manufacturing Technology Institute, Bengaluru
e-mail:- harithota@cmti.res.in

UDAY HIREMATH B.E Mechatronics
Project Associate-I
Smart Manufacturing Demo and Development Cell.
Central Manufacturig Technology Institute, Bengaluru.
e-mail:- udayhiremath02@gmail.com

***Abstract*—This paper explains the methodology of contour-based gluing. The robot moves the tool along the desired path on the job with the assistance of an attached depth-based vision system. This depth-based vision camera is capable of obtaining the coordinates of the contour points. The algorithm can process the data and adjust it based on the requirements, thereby modify the robot's position for the task. This enables the Cobot to follow the defined path for the gluing task. The paper includes experimental results of the robot's contour-following and gluing over the contour of the job.**

Keywords—Depth-based vision system, Cobot, Contour-following)

I. INTRODUCTION

Industry 4.0 aims to reduce human intervention in routine, time-consuming tasks by using advanced technology. Initially, robots began this effort, and now Cobots have taken over with advantages. With routine tasks automated, the focus shifts to non-routine, analytical tasks using vision, AI, and ML. Vision-based technology is proven in the industry, with programs enabling automated and analytical jobs also further reduces human intervention. The Cobot with vision system can achieve the automated wielding, Gluing, painting, Screwing and some other tasks, where human presence earlier was necessary. Present these works under R&D, few were installed in industry.

Contour path following can be achieved by many methods, such as feature extraction, contour tracking algorithms, stereo vision, and deep learning methods. Each method has its pros and cons. After analysing these methods, ‘contour-based point-cloud’ methods stand as an improved version at present.

Here, the vision-based system works on point-based contours. The Intel RealSense D435 is capable of capturing the point cloud of the 3D model, and by applying some filters, it leads to the exact real-world coordinates of the contour path. These points are fed to the UR robot through RoboDK, allowing the robot to follow the contour. This method is significantly better and more accurate for these kinds of tasks. The algorithms filter out unwanted points and depth data. The accuracy of the model depends entirely on the positioning of the camera.

The technology can be employed in the small, medium and large-scale industry, where non-routine works are more. Here In Section-II it lays light on the Calibration of the Cobot, camera, lighting and Active area in section-III concentrates on the algorithms, graphs, tests and its outputs, Section-IV integration of algorithm, Cobot and software.

II. CALLIBRATION OF THE COBOT, CAMERA AND ACTIVE AREA.

A. Callibration of the Cobot

Callibration of the Cobot is intial and prominent step. Calibrating a Universal Robots (UR) robot is essential for ensuring precise operation and alignment of the robot's movements with its environment. Calibration typically involves both hardware and software adjustments to fine-tune the robot’s joint configurations and coordinate systems. The robot intial orientation and height of the tool calculated and placed.The base of the Cobot will be above with respect to the position of the 3D model.This gives the complete fexibility for the Cobot movement.

a. Cobot with tool holder and glue injecter

B. Callibration of the camera.

```
config = rs.config()
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
profile = pipeline.start(config)
depth_intrinsics = profile.get_stream(rs.stream.depth).as_video_stream_profile().get_intrinsics()
```

b. Camera configuration.

As per the camera specifications, the camera needs to be placed at a minimum distance of 21 cm from the model. The camera will be connected at the front end of the cobot and positioned in such a way that the distance between the camera and the 3D model is above 21 cm. The camera specifications are calibrated with respect to the defined resolution. The points collected will depend on the resolution, position, and lighting. Furthermore, the tool holder, which holds the glue injector, will be connected to the tip of the cobot. The TCP will be set with respect to the tip of the nozzle in RoboDK. The camera should not be disturbed by any of these setup.

C. Active area Setup.

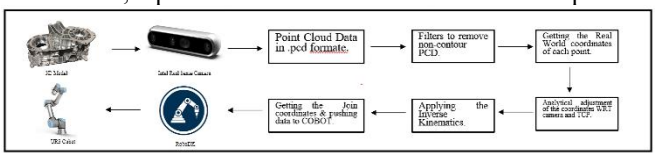
The active area is nothing but the overall area within which the camera can capture the point cloud. This region is affected by the presence of the light and shadow of

the Cobot, so while in consideration of the active area, the calibration and position of Cobot need to be done. The active regions make camera to focus on particular region and gets the point cloud, this will be defined in the camera algorithm as ROI (Region of interest).

c. Active area Setup

III. Algorithms, Graphs, tests and Outputs.

For this task the python finds to be more approachable language which have all flexibility that meets the project requirement in all the aspects. Mainly for vision-based work, OpenCV and VTK libraries are more helpful.



d. Workflow of the algorithm.

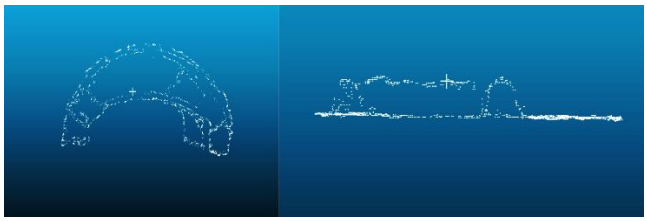
OpenCV is employed to perform the required task. The camera collects the point cloud data of the 3D model placed within the active area. The points collected are pushed to the filters, where the non-contour points are removed. The real-world coordinates are calculated based on the depth of the contour path coordinates.

The obtained coordinates need to be analytically adjusted with respect to the TCP, allowing the cobot to move its joints relative to the tool tip. After obtaining the coordinates, the corresponding joint positions are obtained by applying inverse kinematics, and those positions are fed to the cobot via RoboDK

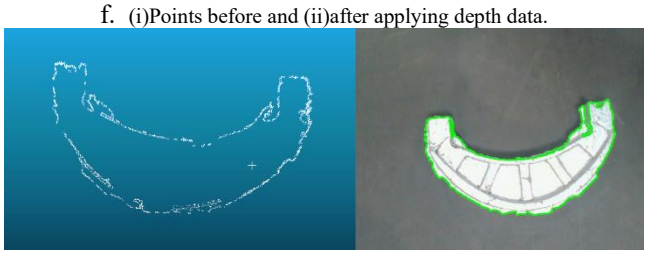
.

e. i. Natural light ii. Artificial Light iii. Combination

Initial tests were conducted on the collection of the point cloud. The above images show the possible outcomes for the model we tested under different conditions, such as the presence of artificial light, natural light, and both. The natural light can provide the broad and even lighting but in changes of weather and time of the day affects the points and area of the points covered. Under artificial light, consistent and controlled illumination can be achieved but the shadow and hotspot affect the accuracy and acquisition. So, test conducted using the both artificial and natural light, the results were promising, the number of the points acquired are almost close to mathematical calculation.



f. (i)Points before and (ii)after applying depth data.



(iii) After applying filters to coordinates.

The data initially will be in raw format, the coordinates for each point need to be calculated based on the depth data, (i)image shows data before applying the depth information (ii) after including depth information and converting into real world coordinate (iii)After applying the filters to coordinates.

Conversion of the raw data into real world coordinates is transforming the 2D data to 3D coordinates. Each pixel in an image is identified by its position (u,v) in the image coordinate system. This system starts at the top-left corner of the image. These parameters relate to the camera's internal characteristics, such as the focal length f and the principal point (c_x,c_y). They are usually represented in a camera matrix K:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Here, fx and fy are the focal lengths in pixels, and (c_x,c_y) is the principal point. Extrinsic parameters define the camera's position and orientation in the world. They are represented by a rotation matrix R and a translation vector t:

$$[R|t]$$

The relationship between 3D world coordinates (X, Y, Z) and 2D image coordinates (u,v) is given by the projection equation:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \, [R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

To find the 3D coordinates from 2D image points, additional information is needed, such as depth data from stereo vision or a depth sensor. The reprojection involves solving for (X, Y, Z) using known depth Z:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = Z \cdot K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

In a stereo vision setup, two cameras provide two different views of the same scene. By finding corresponding

points in both images and using triangulation, you can calculate the depth Z and then determine (X, Y, Z).

Among the obtained point cloud of the whole model, the contour point cloud filters out, those points will be more and need to be reduced.

After getting real world coordinates smoothing filters needed to reduce the number of points which reduce the shivering effect. 'Moving Averaging filter' smooths the data by averaging each point with its neighbors. This filter reduces the points to half of the original points.

$$\tilde{p}_i = \frac{1}{2k+1} \sum_{j=-k}^{k} p_{i+j}$$

The reduced points in (X, Y, Z) format, that need to be update into 6 joint coordinates, that involves analytics. The process employed is inverse kinematics, to understand the mathematical process of inverse kinematics for a Universal Robots (UR) robotic arm when given specific 3D coordinates, we need to delve into the transformation and kinematic equations that the robot uses internally. This involves calculating the joint angles that achieve a specified end-effector position and orientation in space. The inverse kinematics (IK) process typically involves two main steps:

- Position Kinematics: Determining the joint angles required to position the end effector at a specific point in space.
- Orientation Kinematics: Aligning the end effector in the desired orientation.

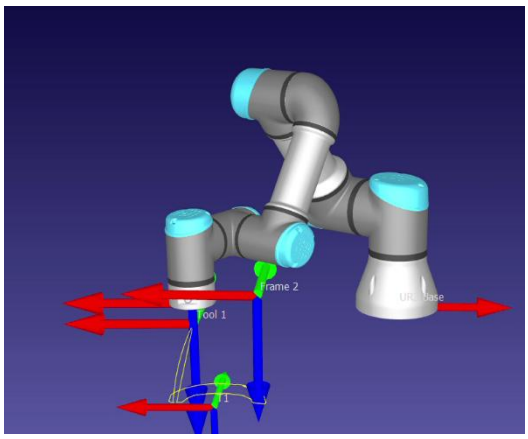
For some robots, especially those with simpler kinematic configurations, analytical solutions can be derived. For a 6-DOF arm, this can be complex and often involves several trigonometric equations:

- Position Equations: Determine the wrist center position by considering the position vector minus the orientation effects. Solve for the base and shoulder joint angles (θ_1, θ_2, θ_3)
- Orientation Equations: Solve for the wrist joint angles (θ_4, θ_5, θ_6) using the orientation part of T_{target}.

To find the set of joint configurations Q = {qi} where qi = (θ^i_1, . . ., θ^i_6) ∈ [0, 2π) 6 that satisfies the equation.

$$ {}_0B_6(\theta_1^i, \theta_2^i, \theta_3^i, \theta_4^i, \theta_5^i, \theta_6^i) = ({}_0B_6^d) = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The orientation for each coordinate will be fed to the Cobot by RoboDK. Here RoboDK capable to handle errors related to connectivity and motion, It provides the seamless action between the Cobot , Camera and Glue injection.

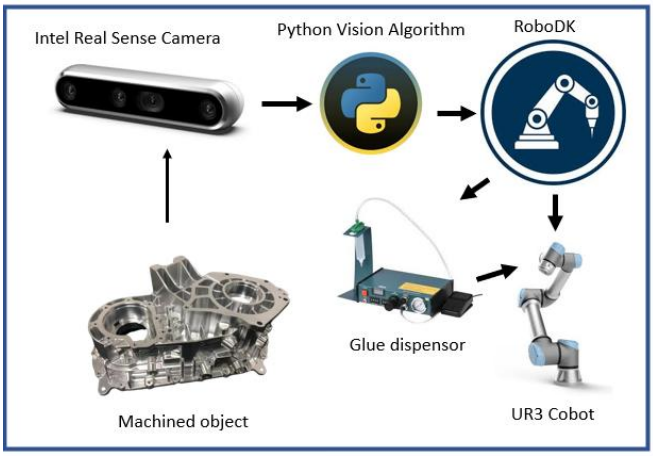


g. Cobot initiation, positioning and Coordinates.

The above images show the final output and the improvement of several factors such as lighting, object positioning, background color, object color, cobot position, and glue dispenser position and its operations, which affected the glue dispensing. All these factors were analyzed one after another with all possible methods, and the synchronized solution was obtained and implementedIntegration of Cobot, Glue Dispenser, Algorithm and RoboDK.

IV. Integration of Cobot, Glue Dispenser, Algorithm and RoboDK.

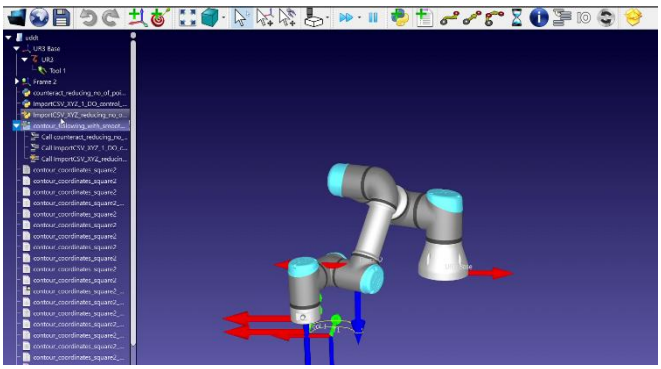
The coordination of the cobot, glue dispenser, algorithm, and RoboDK is a vital part of this project. All these components have internal feedback based on the output. Initially, RoboDK checks the position of the cobot. If the position of the cobot is correct, it initializes the camera. The camera first focuses on the area and sets its active region. Once the active region and lighting are verified, it moves to collect data. This data is also verified and validated by filters. Once validation is approved, it goes for inverse kinematics. After inverse kinematics is applied, the algorithm checks the availability of the glue dispenser by toggling the device.



Once availability is confirmed, RoboDK connects to the cobot. Here, RoboDK also validates all these steps once and instructs the cobot with the position data.

h. The workflow of Glue Application Cobot.

verifications, validation and approves internally and faster in coordination, if any one is failed the error pops up and stops the working.



i. The User Interface of RoboDK

V. CONCLUSION

In this paper we presented the “Point cloud-enabled contour detection for adhesive application.”, Here the methodology used in this application is more approachable in the industry, this involved very basic instruments with cooperative algorithms. All the above test and graphs shows the improvisation and approach towards the task. This methodology can also be employed for the screwing, painting, welding and some other tasks where the contour following is necessary. The changes in tool, TCP position and program can easily enable the above opportunity.

The integration of Cobots equipped with advanced vision systems marks a significant advancement in Industry 4.0, reducing human intervention in routine tasks and shifting focus to more complex, analytical activities. By employing point-cloud-based contour detection, particularly with the Intel RealSense D435, we achieve higher accuracy and efficiency in automated processes such as gluing, welding, painting, and screwing. This method leverages the real-world coordinates of contour paths, enabling precise control of Cobots via RoboDK.

The calibration of the Cobot, camera, and active area is crucial for the system's performance, ensuring that the vision-based algorithms can accurately interpret and follow contour paths. The use of OpenCV and VTK libraries in Python provides the necessary flexibility and robustness for processing point cloud data and controlling the Cobot's movements.

This technology can be effectively applied across small, medium, and large-scale industries, enhancing productivity and consistency in non-routine tasks. The successful integration of the Cobot, glue dispenser, vision algorithms, and RoboDK demonstrates a seamless workflow, validated through rigorous testing and real-time feedback mechanisms.

In summary, the point-cloud-based contour detection method represents a significant improvement in automated industrial applications, providing a reliable, accurate, and efficient solution for complex tasks traditionally requiring human intervention. Future developments in this field hold the promise of even greater advancements in automation and efficiency.

Vi. REFERENCE.

[1]. Marcin Grzegorzek, Christian Theobalt and Reinhard Koch, “Time-of-Flight and Depth Imaging. Sensors, Algorithms and Applications”.

[2]. Kelsey P. Hawkins, “Analytic Inverse Kinematics for the Universal Robots UR3”

[3]. Adrian Kaehler, Gary Bradski “ Learning OpenCV”

[4]. RoboDK Documentation.
‘https://robodk.com/doc/en/Basic-Guide.html’

[5].UR Cobot Documentation.