Journal of Machine Engineering, 2020, Vol. 20, No. 3, 30–43
ISSN 1895-7595 (Print) *ISSN 2391-8071 (Online)*

Received: *20 May 2020* / Accepted: *03 June 2020* / Published online: *25 September 2020*

OpenCasCade, PythonOCC,
off-line robot programming,
powder bed printer, powder suction

Faramarz BAGHERZADEH[1,2*]

AUTONOMOUS OFF-LINE ROBOT PROGRAMMING FOR POWDER SUCTION OPERATION WITH PYTHONOCC

While a powder bed 3D printer device is easy to use, the cleaning task after each print is a tedious job. Consequently, a proper approach is to employ an industrial robot for this task. The robot should be programmed quickly and efficiently with the off-line robot programming (OLP) method. In this paper, an OLP system based on Python and OpenCasCade libraries is introduced to generate robot trajectories for cleaning the printer powder bed immediately and autonomously. The cleaning operation is divided into three sub-operations: top layer raster, raster from the offset, and offset oriented. Several algorithms are employed to satisfy sub-operations autonomously from a CAD model. Raster path, wire, and yaw angle calculators are essential algorithms. Finally, a graphical simulation illustrates the operation efficiency. The proposed system can generate a cleaning path immediately and due to utilizing open resource libraries, there is a wide range of applicable personalization.

1. INTRODUCTION

Since the 3rd industrial revolution in the 20th century, by the presence of memory programmable-controls, automation was made to boost efficiency and productivity. In order to increase the level of automation as much as possible, various technologies have been invented and among them, Industrial robots are considered as one of the most universal equipment. In each step of the production process, various operations are needed. These operations are done by either machines or humans. Industrial robots are widely used for various operations such as pick and place, painting, gluing, welding, and machining. In order to utilize industrial robots for an operation, they should be programmed similarly to traditional CNC machines. This programming is done considering robot native language which is made by a robot manufacturer. In general, there are two main methods of robot programming. Industrial robots usually are programmed either On-Line or Off-Line [1, 2].

On-Line Programming is often known as a robot teaching method. The operator moves manually the robot end-effector to the desired position (point by point) with considering proper orientation to perform each step of the given task [1]. This movement is recorded by

[1] Mechanical Engineering, Gdansk University of Technology, Poland
[2] Mechanics and System Design, Polytechnic University of Tours, France
E-mail: mr.bagherzade@gmail.com
https://doi.org/10.36897/jme/123446

the robot controller and later on translated as executable commands in the native language of the robot (KUKA, Motoman, ABB, FANUC, etc.) [2]. Although this method is time-consuming and tedious, the operator does not have to gain high programming skills. Robot off-Line programming (OLP) refers to programming outside of the robot cell without interrupting production. This strategy is similar to traditional CNC programming (designing in CAD and moving to CAM then generating NC code) with two more steps, kinematic post-processing, and simulation [3]. The production stoppage is the most unpleasant side effect of robot reprogramming. The off-line programming method utilizes a graphical representation and a set of calculations that allows the operator to simulate the robotic scenario and generate demanded trajectories without interrupting the manufacturing process. Only a brief stoppage is necessary for downloading the program into the robot cell controller [4].

2. POWDER BED FUSION

Generally, manufacturing techniques are available in three major branches: subtractive, formative, and additive [5]. The additive method (deposition of material layer by layer) is almost a new technology and maybe a key to the next industrial revolution [6]. Powder bed fusion (PBF) is an additive method that fuses metal powder with the sintering or melting process via specific techniques. Usually, there are five procedures for fusing powder: direct metal laser sintering (DMLS), selective laser melting (SLM), selective laser sintering (SLS), selective heat sintering (SHS), and electron beam melting (EBM). In powder bed fusion, a thin layer of powder is spread overbuild platform with various mechanisms such as roller or blade. Then, a tool of fusion such as laser fuses the new layer to the previous one [7].

When the process is completed, the component is in a bed of powder. Finally, the last step is to clean the powder bed and remove the produced part. Often, this job is done manually. When a human operator cleans the bed, there are several drawbacks such as operator health risk, labour cost, purchasing consumable safety equipment, and time losses [8, 9]. The particular aim of this research is to develop an open-access autonomous OLP package for industrial robots to remove the excess powder around the produced component via a suction tube.

3. BACKGROUND ON OLP

Generally, OLP software divides into three groups with considering the type of software producer. OLP offered by robot manufacturers concerning specific tasks (Example: Fanuc's "Coordinated motion package", "KUKA.ConveyorTech", ABB's "Motion Coordination") [10], offered by generic robotic software producer (Example: DELMIA, Robotmaster, RoboDK) [11] or research institution which developed their packages [2]. Mostly researchers develop OLP software in two methods. It might be developed around existing CAD software like Inventor, Matlab, SolidWorks, and

Rhinoceros in the form of scripting and parametric programming or from scratch with OpenGL, C++, Python, and OpenCasCade.

Recently, [11] utilized a commercially available package, so-called DELMIA Automation, to generate a path for a complex welding robotic cell in a vehicle hull manufacturing company in Australia. The aim is to analyse the feasibility of the mentioned software for the proposed job. After importing a CAD model, the user should select an edge to introduce a seam, then tag the start and endpoint of the seam. DELMIA generates the path with proper end-effector orientation concerning user-defined properties and cell dimensions. Also, reachability, clash checking, and simulation offered by this package. Although commercial packages are available, they do not support all manufacturing fields.

As mentioned in [3] one of the most typical problems in the field of architecture design is manufacturability. Therefore, they are proposing a special type of OLP which helps designers in a bottom-up process. The designer can check the robot's end-effector position, orientation, and maximum tool length instantly while designing the model and in case of any conflict, the designed model can be modified at the moment. This method is implemented via parametric programming with the Grasshopper module in Rhinoceros CAD software. Finally, the production process can be simulated and a post-processor translates trajectories to KUKA Robot Language (KRL).

A common method to establish an OLP software is to employ an application programming interface (API). Recently, [12] developed an OLP around Autodesk Inventor with utilizing the API environment. This program is offering modules for pick and place operations to defined obstacles in a 3D environment. It also introduces other possibilities for developing similar OLP platforms for a given robotic cell. A set of simulation algorithms and collision detection codes form API modules that help to generate and post-process a suitable trajectory.

Similarly, in [12] an off-line programming system is developed based on SolidWorks via API functions to weld pipe intersecting joints with ABB IRB1410 arc welding robot. In a similar approach, [13] UG/C++ OLP system is developed around UG software (a CAD package) via programming in C++ language and modules from UG/OPEN. It takes advantage of the genetic algorithm for generating an effective trajectory for 6 DOF industrial robots to use in welding car doors. After defining robot links and robotic cells, the path planning modules use Descartes space path planning. It generates a proper trajectory concerning time function which calculates the acceleration and speed of end effector.

Commercially available CAD packages mostly provide tools for users to interact freely via scripting. The research in [14] introduces an OLP system for a hyper-flexible welding robotic cell. It indicates a personalized workbench built in FreeCAD via Python scripting. This program reads the CAD model and generates the trajectory and orientations concerning welding parameters such as torch type and welding distance.

Recently, [15] and [16] established an OLP package with Python programming language and OpenCasCade (OCC) libraries. This library also is known as OpenCasCade Technology (OCCT). After importing a CAD model and selecting an edge or wire by the user, the program will make a curve. Then, OCC functions generate a map of points on the curve. Next, the system calculates the roll, pitch, and yaw angles by an orthogonal matrix. A loop algorithm checks accessibility to avoid gimbal lock. When all information regarding end effector movement is available, then a reverse kinematic module carries out transformation data for robot joints. This data represents in a 3D simulation environment and system checks for any collision. The authors ran two experiments, one for gluing in a circular path and another for moving a manipulator on a curve on a ball.

Similarly, [17] utilized PythonOCC to develop an auxiliary software robot milling OLP system. This program generates a path for machining free form surfaces via industrial robots. It offers a novel algorithm for increasing machining accuracy with the new method in calculating the path with the Cutter Location point path (CL) cross-section method. Finally, the generated path is simulated in Vericut software to compare the surface quality with traditional methods.

4. OVERVIEW OF PROPOSED OLP SYSTEM

The proposed OLP package is based on Python 3.7 with taking advantage of OCC libraries. This program is designed to generate trajectories autonomously and immediately for cleaning the printer bed. The system includes input, core codes, and output (Fig. 1). At first, the user data is collected by Graphical User Interface (GUI), then core codes process the input data via utilizing OCC functions and finally, outputs are end-effector path, orientation, and operation simulation.

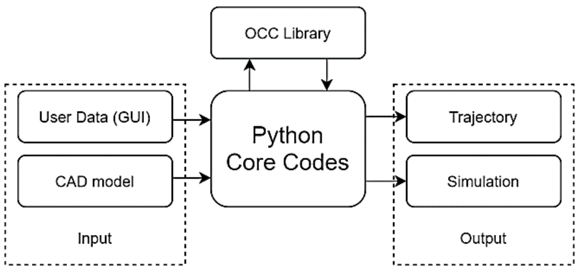


Fig. 1. General Structure of the system

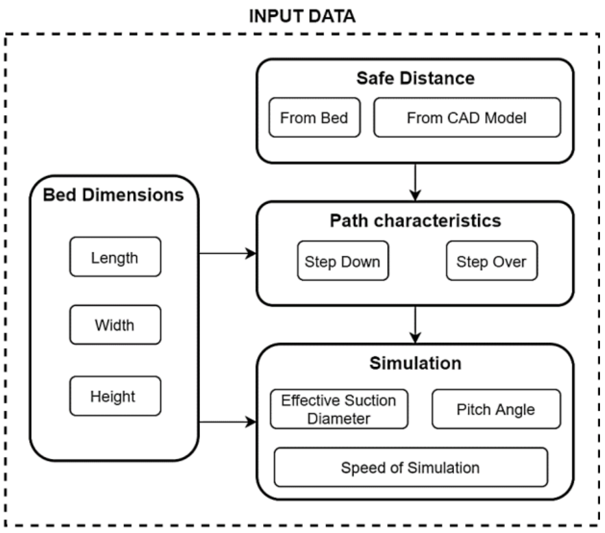


Fig. 2. Input data structure (inheritance mechanism)

Input data is collected in advance under 4 main packages (Fig. 2). Each package contains classes for generating a specific kind of data. In order to avoid rereading input data, higher-level input packages are sharing information from lower-level classes (inheritance mechanism). For example, the powder bed dimensions are collected via GUI then a class creates the bed material box, then this box is shared with the simulation input package to perform the simulation.

There are four items in the menu bar for the user to communicate with the application. Each item takes some parts of the input data and returns specific results.

- Display Model: it displays the STL model and powder bed through sub-functions (Fig. 3).
- Erase All: it deletes everything from the display.
- Generate Path: this option creates the trajectory autonomously by utilizing pre-defined functions and sub-functions (Fig. 4).
- Simulate Path: it receives the trajectory from the previous step and simulates the operation with moving the end-effector on the trajectory, simultaneously, it is subtracting powder from powder bed concerning the effective diameter of the suction tube.

Cleaning the powder bed via the suction tube is the main task of the proposed OLP system. We divide this task into 3 sub-operations: top layer raster, combined raster, and offset oriented. Each sub-operation utilizes several functions. These functions build the structure of the application. Often, a function is used by more than one operation.

During the suction operation, the suction tube should be close to the powder as much as possible for two main reasons. First, the operation effectiveness decreases dramatically with distancing. Second, the distance between the powder and suction tube makes a dusty environment. For these reasons, often it is desired to have tangential or interference position of the suction tube.

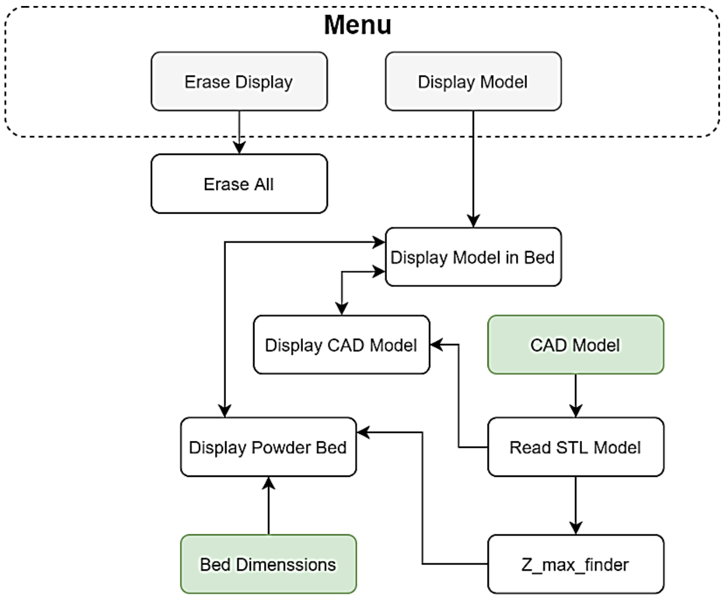


Fig. 3. Menu structure (Display)

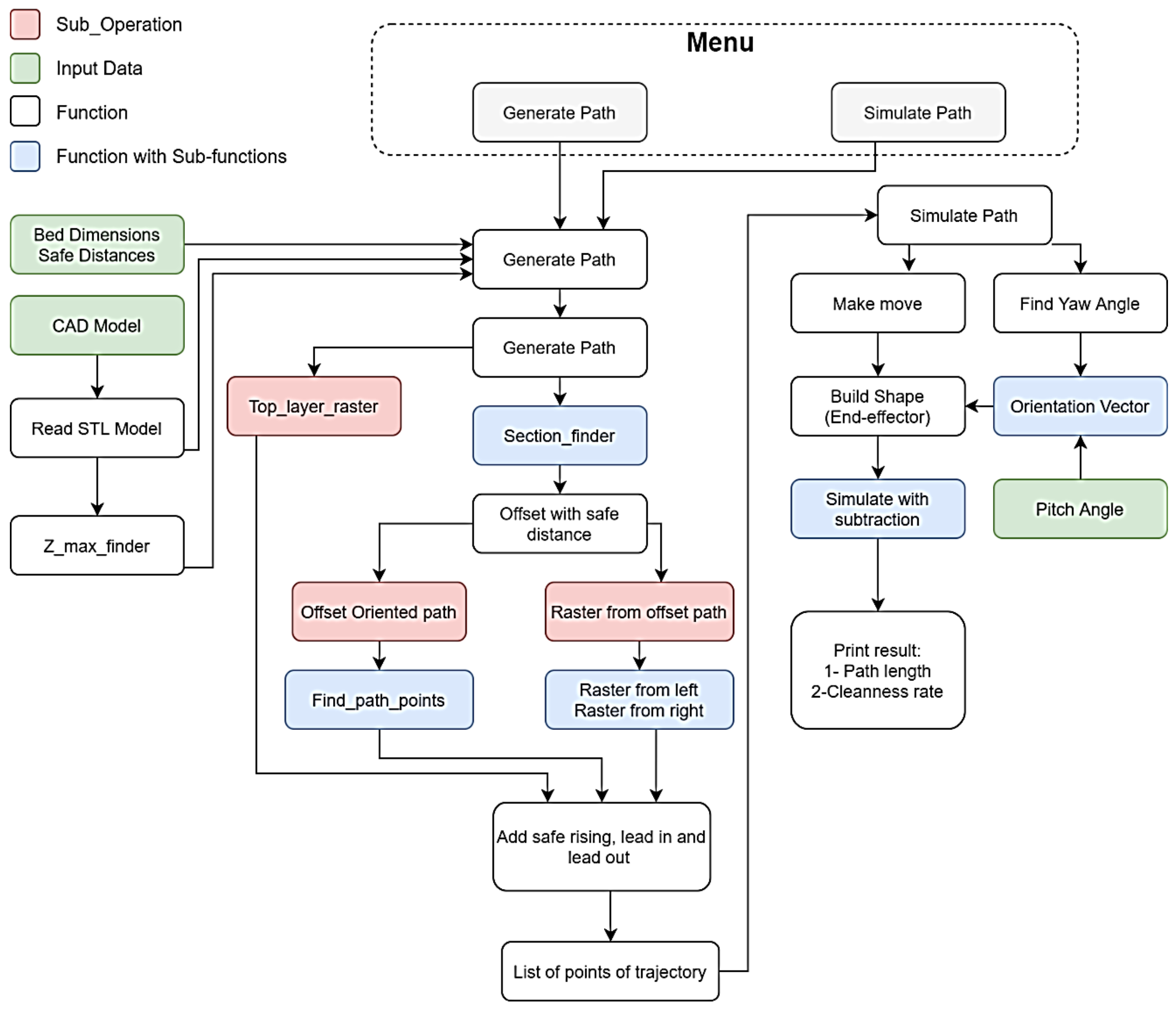

Sub_Operation
Input Data
Function
Function with Sub-functions
Menu
Generate Path
Simulate Path
Bed Dimensions Safe Distances
CAD Model
Read STL Model
Z_max_finder
Generate Path
Generate Path
Top_layer_raster
Section_finder
Offset with safe distance
Offset Oriented path
Raster from offset path
Find_path_points
Raster from left Raster from right
Add safe rising, lead in and lead out
List of points of trajectory
Simulate Path
Make move
Find Yaw Angle
Build Shape (End-effector)
Orientation Vector
Simulate with subtraction
Pitch Angle
Print result: 1- Path length 2-Cleanness rate

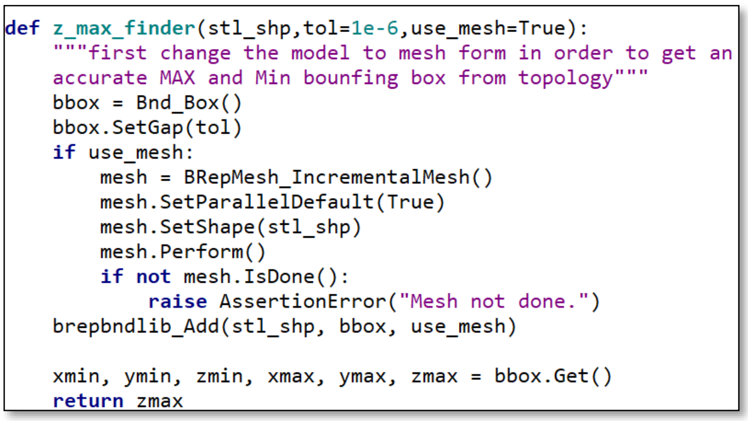
Fig. 4. Menu structure (Generate Path and Simulate)

Important functions are listed below with a brief description. They may not appear in the main structure (Fig. 4) as they are used in the form of sub-functions:

- Section_finder: creates an intersecting plane with the CAD model and returns the intersection topological data.
- Make_edge_list: receives a set of topological data (from Section_finder) then it explores the data and returns a list of available edges in the topological object.
- Find_outer_boundary: receives a list of edges (from Make_edge_list) and returns a close wire.
- Find_direction: it gets two points (from trajectory) and constant pitch angle from input data, then it returns the vector of end-effector orientation.
- Build_shape: it generates a shape (robot end-effector) that can be animated in a 3D environment concerning the end-effector vector (from Find_firection).
- Z_max_finder: receives the CAD model and returns the maximum height of the CAD model.

For example, *Z*_max_finder (Fig. 5) is used in both path generation and simulation. At first, this function creates a bounding box (*Bnd_Box*()) and a mesh object from the CAD model (*BrepMesh_IncrementalMesh*()), then it fits the box on the mesh object (*brepbndlib_Add*(*stl_model*,*Bbox*,*use_mesh*)), finally it returns the maximum height of the bounding box (*return zmax*).

It is not possible to mention all functions and sub-functions in this paper, but the code is available for the public at the GitHub repository of the author.



```
def z_max_finder(stl_shp,tol=1e-6,use_mesh=True):
    """first change the model to mesh form in order to get an
    accurate MAX and Min bounfing box from topology"""
    bbox = Bnd_Box()
    bbox.SetGap(tol)
    if use_mesh:
        mesh = BRepMesh_IncrementalMesh()
        mesh.SetParallelDefault(True)
        mesh.SetShape(stl_shp)
        mesh.Perform()
        if not mesh.IsDone():
            raise AssertionError("Mesh not done.")
    brepbndlib_Add(stl_shp, bbox, use_mesh)

    xmin, ymin, zmin, xmax, ymax, zmax = bbox.Get()
    return zmax
```

Fig. 5. Function code (Z_max_finder)

The top layer raster is a trajectory with the classical raster movement on the top of the bed for cleaning the first layer. Considering there is no obstacle in this layer, it generates the path without pitch angle to remove powder from the bed with a small safe distance (Fig. 6). The input data is the bed dimensions, step over and CAD model height. The CAD model height is obtained via *Z*_max_finder. Finally, a raster algorithm (Fig. 7) generates the path in the form of the classical raster on the top of the bed.

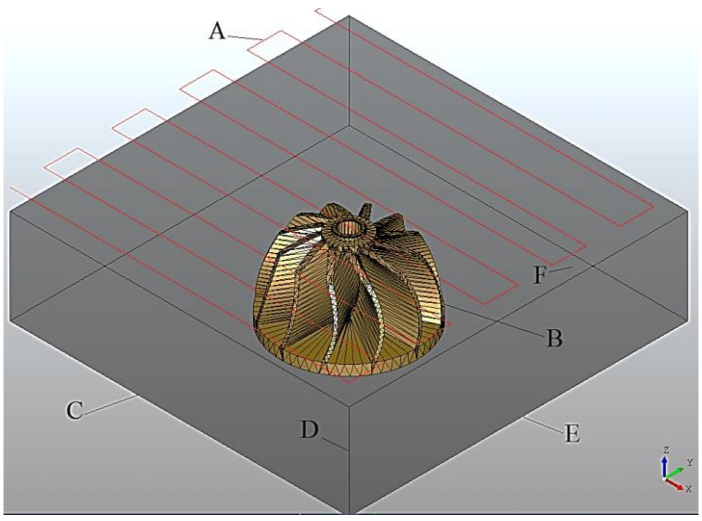


Fig. 6. Top layer raster (A: Trajectory, B: CAD model, C: Bed length, D: Bed height, E: Bed width, F: Safe distance)

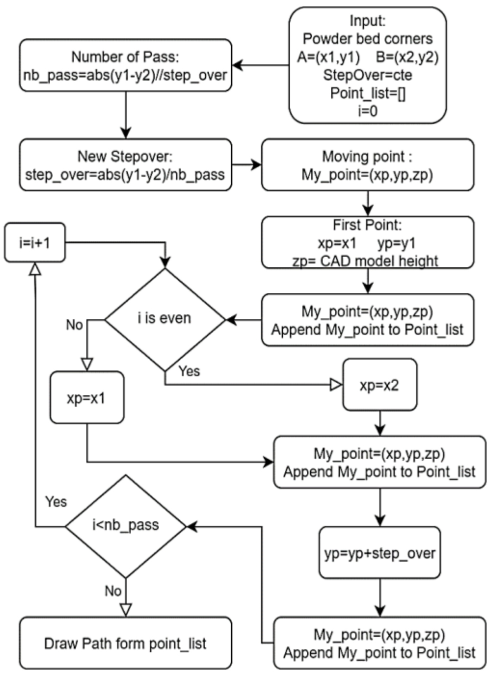


Fig. 7. Algorithm for generating raster path

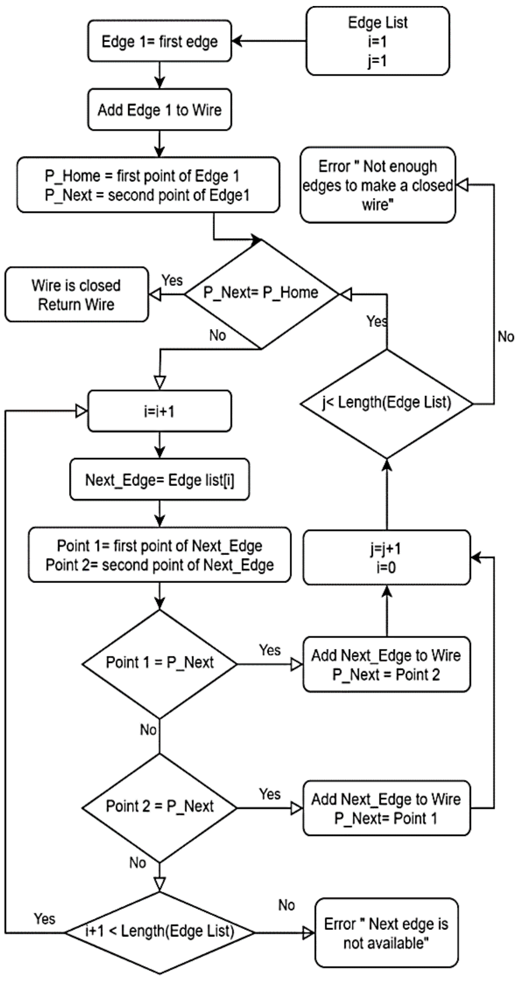


Fig. 8. Algorithm for autonomous wire generating

A combined raster trajectory is a 2D path concerning a safe distance from the CAD model cross-section and the powder bed walls. This sub-operation aims to clean the bed as much as possible, thus the pitch angle is zero. For generating this trajectory at first, a loop slices the CAD model according to user input slicing depth. The depth of each layer (step down) is constant in order to avoid incoherence of material (powder) removing [18]. In each slicing level, the system defines a plane (*gp_Pln*()-OCC Function for drawing a plan) to find the cross-section of the model (*BRepAlgoAPI_Section*()-OCC Function to obtain intersection between objects). Then the algorithm (Fig. 8) obtains a wire from cross-section topological data autonomously. Next, the program generates an offset wire with safe distance (*BRepOffsetAPI_MakeOffset*()-OCC Function for generating an offset from a wire) and divides the raster path into two regions to have access all around the model. Next, the raster is modified considering intersection points with offset wire. Finally, the raster path and offset wire are combined with no user intervention (Fig. 9).

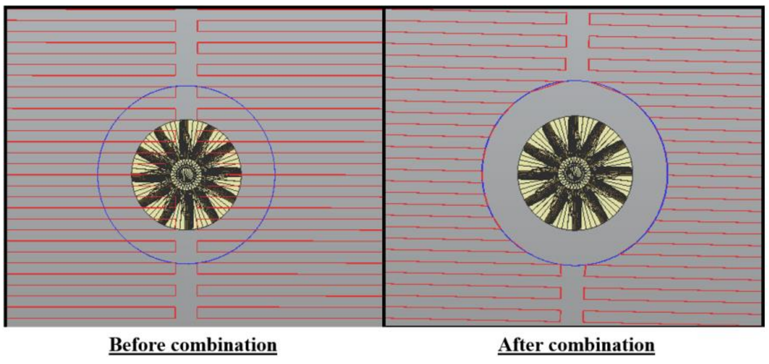


Fig. 9. Raster and offset combination

Offset oriented trajectory is defined to clean the printed component as much as possible, therefore, the suction tube orientation is always towards the component with a constant user-defined pitch angle (Fig. 10). This sub-operation uses an offset wire with less distance than one in the combined raster to get closer to the printed part. During this task, the end-effector yaw angle should always be perpendicular to the path. The algorithm (Fig. 11) shows the calculation process of the yaw angle.

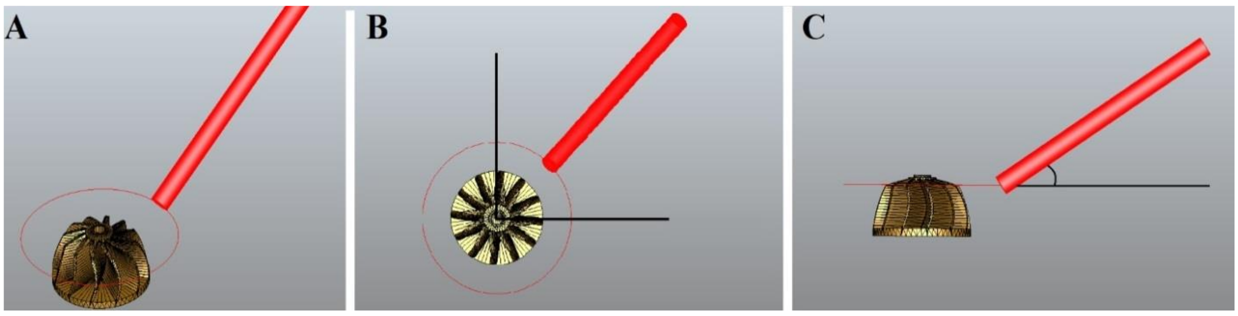


Fig. 10. Offset oriented path with suction tube – A: Perspective view, B: Top view (Yaw angle), C: Side view (pitch angle)

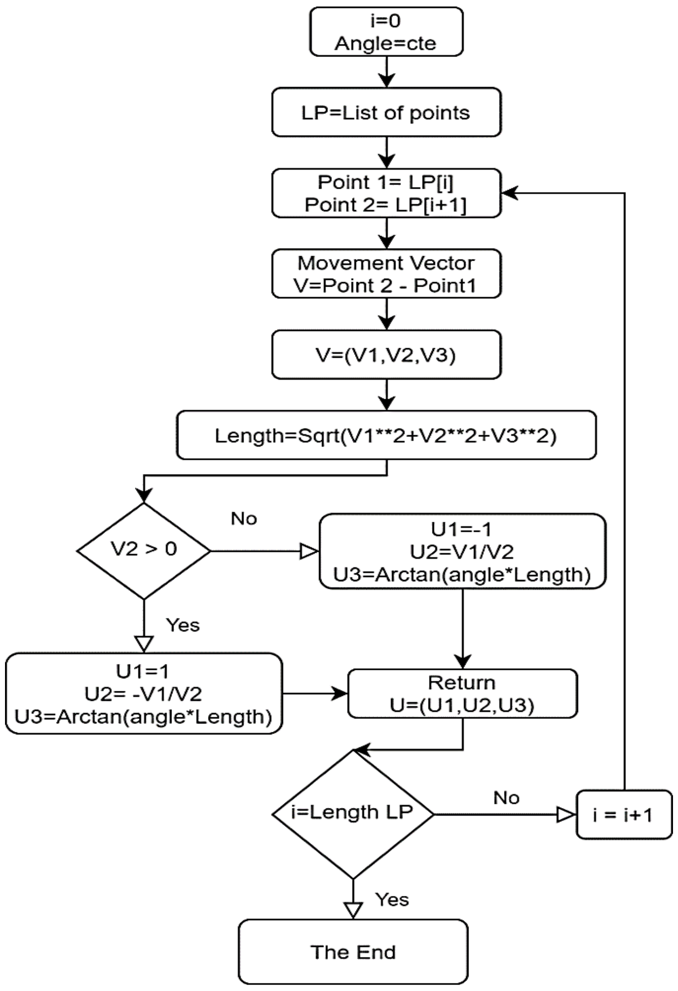


Fig. 11. Algorithm of yaw angle

Simulation is a necessary step to make sure the generated trajectory is effective and user-defined data such as step over and pitch angle are properly selected (Fig. 12). Simulation function receives the trajectory in the form of a list of points from other functions (Fig. 4). It generates a set of movement orders to *Make_move* considering simulation speed (input data). Simultaneously, it calculates the yaw angle of the end-effector and the orientation vector. Before initiating the movement, it creates the end-effector cylinder (a moveable object) through the functions *Build_shape*. Next, the simulation movement (moving end-effector) and simulation subtraction (subtraction from powder) begin simultaneously.

During the simulation with subtraction, a Boolean operation occurs between the powder bed and a cylinder on the tip of the robot end-effector. This cylinder is known as the effective suction region. The diameter of the cylinder is defined by the user concerning the suction power and diameter of the suction tube. The effective suction diameter is obtained by experimental data on each type of powder, thus this data is specifically defined for each machine by the user of application or machine operator. In the end, demanded data regarding trajectory and simulation is printed out, for example, cleaning ratio and path length as described in Section 5.

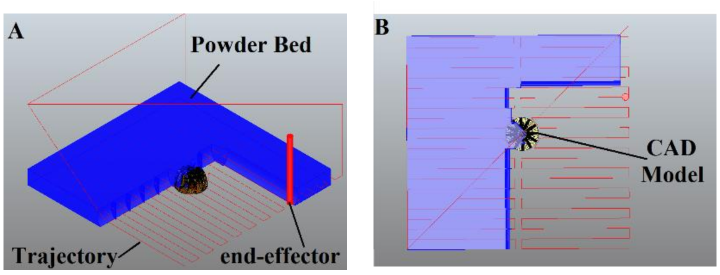


Fig. 12. Simulation: A – Perspective view, B – Top view

5. THEORETICAL EXPERIMENT

A theoretical experiment is done on a sample part (Fig. 13) in a powder bed (60 cm × 60 cm × 22 cm) to check the program process and demonstrate the practical data generated by the program. In this experiment, a component (Number 90) form [19] is chosen as the CAD model. The simulation and results are calculated concerning various suction tube diameters and step over ratios (equation 5.1). The following equations are considered during the experiment.

$$\text{Step over ratio } = \frac{\text{Step over}}{\text{Tube diameter}} \times 100 \tag{5.1}$$

$$\text{Step down } = \frac{\text{Tube diameter}}{2} \tag{5.2}$$

$$\text{Cleaning ratio } = \left(1 - \frac{\text{Remained powder}}{\text{Bed volume} - \text{CAD model volume}}\right) \times 100 \tag{5.3}$$

Fig. 13. CAD model from [19]

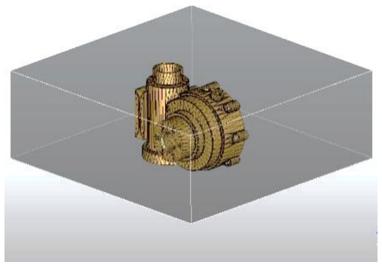


Fig. 14. Display CAD model

6. RESULTS

At first, the model is displayed in a powder bed (Fig. 14), then the path is generated around the CAD model (Fig. 16). After that, the program prints out the length of the generated path. Next, the simulation process begins. During the simulation process, the user can observe the progress in suction operation (Fig. 17). The effective suction diameter is assumed to equal to the diameter of the suction tube with a semi-circle cross-section. With moving a semi-circle profile on the path curve, the sweep operation makes a half-cylinder object. The half-cylinder removes the material. This part of the program can be adjusted to any profile in later improvements. At the end of the simulation, the leftover powder is displayed (Fig. 15).

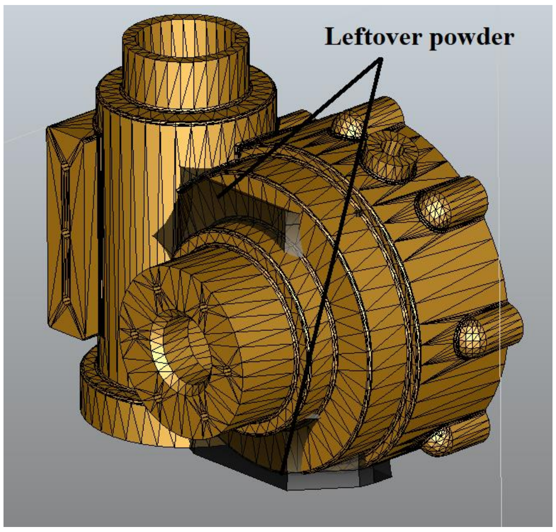


Fig. 15. Leftover powder

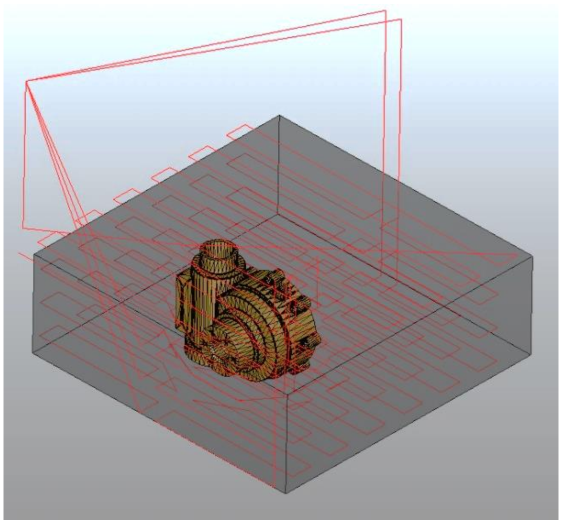


Fig. 16. Generate path

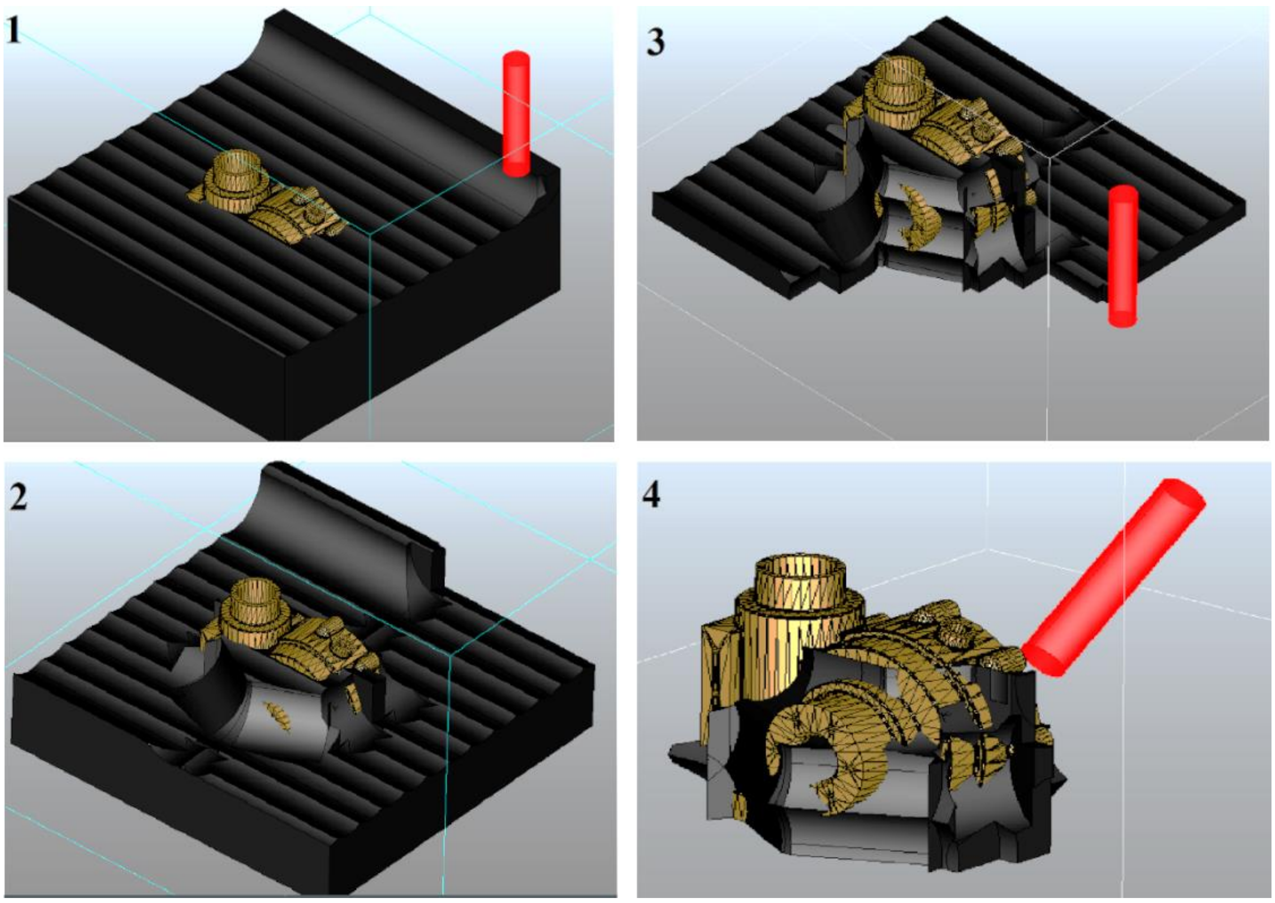


Fig. 17. Simulation process (1 – Top layer raster, 2 and 3 – Raster from offset, 4 – Offset oriented)

Finally, the trajectory length (Fig. 18) and the cleaning ratio (Fig. 19) (according to equation 5.3) is printed out. The graph illustrates the results of the experiment for the four-step over ratios (40%, 60%, 80%, and 100%), and helps the user to choose a proper step over and suction tube.

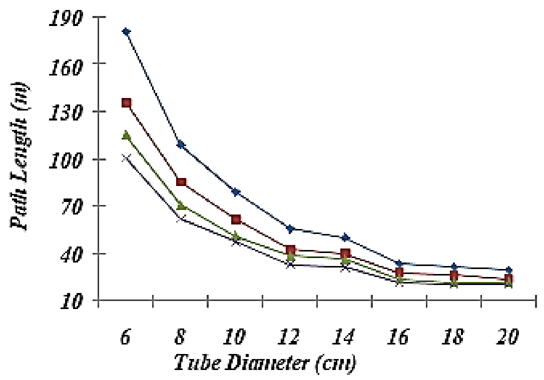


Fig. 18. Path length

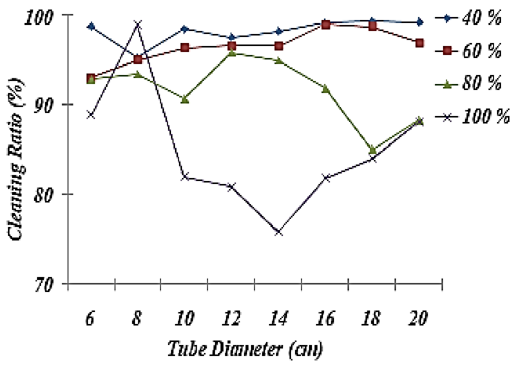


Fig. 19. Cleaning ratio

7. CONCLUSION

The proposed method generates a coherent trajectory for the mentioned task in three sub-operations: top layer raster, raster from the offset and offset oriented. The introduced algorithms extract essential data from the CAD model (STL) and calculate the suction tube path and orientation for each sub-operation. The simulation result indicates the proposed method is fast, effective, and with enough accuracy. It removes a considerable amount of powder from the bed and the printed object. This stand-alone program can generate trajectories immediately and autonomously with several mouse clicks. Due to utilizing free open-access modules, there are many more possibilities for further investigation of personalized features and improvements. Although CAD models that are produced via 3D printer machines are very complicated, for the suction operation they can be introduced by a simple CAD equivalent geometry to avoid errors and miss calculations. Finally, to translate the generated trajectory to executable robot commands, a post-processor should be introduced separately concerning the robot language and dimensions.

REFERENCES

[1] KOHR C.T., STAMP R., PIPE A.G., KIELY J., SCHIEDERMEIER G., 2013, *An Online Robot Trajectory Planning and Programming Support System for Industrial Use,* Robot. Comput. Integr. Manuf., 29/1, 71–79, Feb., DOI: 10.1016/j.rcim.2012.07.010.

[2] GAN Y., DAI X., LI D., 2013, *Off-Line Programming Techniques for Multirobot Cooperation System,* Int. J. Adv. Robot. Syst., 10, DOI: 10.5772/56506.

[3] BRAUMANN J., BRELL-COKCAN S., 2011, *Parametric Robot Control: Integrated CAD/CAM for Architectural Design*, Integr. Through Comput. – Proc. 31st Annu. Conf. Assoc. Comput. Aided Des. Archit. ACADIA, 242–251.

[4] MITSI S., BOUZAKIS K.D., MANSOUR G., SAGRIS D., MALIARIS G., 2005, *Off-Line Programming of an Industrial Robot for Manufacturing*, Int. J. Adv. Manuf. Technol., 26/3, 262–267.

[5] KHAIRALLAH S.A., ANDERSON A.T., RUBENCHIK A., KING W.E., 2016, *Laser Powder-Bed Fusion Additive Manufacturing*: *Physics of Complex Melt Flow and Formation Mechanisms of Pores, Spatter, and Denudation Zones*, Acta Mater., 108, 36–45, DOI: 10.1016/j.actamat.2016.02.014.

[6] BOURELL D.L., ROSEN D.W., LEU M.C., 2014, *The Roadmap for Additive Manufacturing and Its Impac*t, 3D Print. Addit. Manuf., 1/1, 6–9, DOI: 10.1089/3dp.2013.0002.

[7] ASHRAF M., GIBSON I., RASHED M.G., RASHED M.G., 2018, *Challanges and Prospects of 3D Printing in Structural Engineering*, 13th Int. Conf. Steel, Sp. Compos. Struct., 13, 1–9, (Online), Available: https://www.researchgate.net/publication/320943125.

[8] MOYLAN S., SLOTWINSKI J., COOKE A., JURRENS K., DONMEZ M.A., 2013, *Lessons Learned in Establishing the NIST Metal Additive Manufacturing Laboratory*, NIST Rep., DOI: 10.6028/NIST.TN.1801.

[9] *5 most common myths about SLS powder handling*, Sinterit, Manufacturer of high quality desktop SLS 3D printers, https://www.sinterit.com/5-most-common-myths-about-sls-powder-handling/ (accessed May 11, 2020).

[10] MIKUSZ M., CSISZAR A., 2015, *CPS Platform Approach to Industrial Robots: State of the Practice, Potentials, Future Research Directions*, (Online), Available: http://aisel.aisnet.org/pacis2015http://aisel.aisnet.org /pacis 2015/176.

[11] POLDEN J., PAN Z., LARKIN N., VAN DUIN S., NORRISH J., 2011, *Offline Programming for a Complex Welding System Using DELMIA Automation*, Lect. Notes Electr. Eng., 88/341–349, DOI: 10.1007/978-3-642-199 59-2_42.

[12] NETO P., MENDES N., 2013, *Direct off-Line Robot Programming Via a Common CAD Package*, Rob. Auton. Syst., 61/8, 896–910, DOI: 10.1016/j.robot.2013.02.005.

[13] SHEN H., 2017, *Research on the Off-Line Programming System of Six Degree of Freedom Robot in Vehicle Door Welding Based on UG*, M2VIP 2016 – Proc. 23rd Int. Conf. Mechatronics Mach. Vis. Pract., 1–5, DOI: 10.1109 /M2VIP.2016.7827309.

[14] FERREIRA L.A., FIGUEIRA Y.L., IGLESIAS I.F., SOUTO M.Á., 2017, *Offline CAD-based Robot Programming and Welding Parametrization of a Flexible and Adaptive Robotic Cell Using Enriched CAD/CAM System for Shipbuilding*, Procedia Manuf., 11, 215–223, DOI: 10.1016/j.promfg.2017.07.228.

[15] BEDAKA A.K., LIN C.Y., 2017, *Autonomous Path Generation Platform for Robot Simulation*, Int. Conf. Adv. Robot. Intell. Syst. ARIS, 63–68, DOI: 10.1109/ARIS.2017.8297186.

[16] BEDAKA A.K., LIN C.Y., 2018, *CAD-Based Robot Path Planning and Simulation Using OPEN, CASCADE,* Procedia Comput. Sci., 133, 779–785, DOI: 10.1016/j.procs.2018.07.119.

[17] ZHANG B., HUANG B.B., SONG Y.Q., TANG C., 2015, *A Novel Tool Path Generation Method for Robot Milling Process*, Mechanics and Mechatrinics, 936–945, DOI: 10.1142/9789814699143_0114.

[18] JAROSZ K., LÖSCHNER P., NIESLONY P., KROLCZYK G., 2017, *Optimization of CNC Face Milling Process of Al-6061-T6 Aluminum Alloy*, J. Mach. Eng., 17/1, 69–77.

[19] CONDEI D., 2016, *ALBUM cu 100 piese mecanice*, Bucuresti, ISBN: 978-606-8707-23-5.