Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 118 (2023) 507–512

www.elsevier.com/locate/procedia

16th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME '22, Italy

Automated Tool Trajectory Generation for Robotized Deburring of Cast Parts Based on 3D Scans

Ingrid Fjordheim Onstein[a,]*, Magnus Bjerkeng[b], Kristian Martinsen[b]

[a]*Department of Manufacturing and Civil Engineering, Norwegian University of Science and Technology, Gjøvik, Norway*
[b]*Department of Mathematics and Cybernetics, SINTEF Digital, Oslo, Norway*
[c]*SINTEF Manufacturing AS, PO Box 163, 2831 Raufoss, Norway*

* Corresponding author. Tel.: +47 48200366; *E-mail address:* ingrid.f.onstein@ntnu.no

Abstract

Manual removal of burrs on castings introduces health, safety, and environmental concerns. Automated removal of highly variable casting burrs could improve safety, but requires a solution based on robots, smart sensors, and advanced algorithms to tackle the problem in a flexible and cost-effective way. This paper presents a system for automatic tool trajectory generation for robotic deburring of cast parts with a specific focus on the robotic tool trajectory generation algorithm. The system generates a robotic trajectory adapted to the specific workpiece based on CAD model and 3D scans of the workpiece. The registered 3D scans and CAD model is used to generate a 3D model of each individual workpiece. This is fed into the tool trajectory generation algorithm. The algorithm uses the generated 3D model as well as a priori knowledge of the casting process to generate the tool trajectory. The tool trajectory planning algorithm has been tested on a set of various 3D models of cast parts, and the generated robotic tool path has been both simulated and tested on a KUKA KR 60.

Keywords: Deburring; Robotics; Manufacturing; 3D Scan

1. Introduction

Fettling is a process to remove sprues, runners, risers, and flashing from sand casted parts. Flashing, also called burrs, is the excess material left on the cast part in the separation plane between the two molds, see Fig. 1. Fettling is important to ensure that the part meets its design requirements, and to ensure safe handling of the part for the operators. Sprues, risers, and runners can be removed in a cutting process that leaves a burr to be removed. These burrs, in combination with the flashing, is removed in a deburring process. Deburring is mostly done manually by operators today, where they are exposed to high noise and vibration levels. Therefore, it is desirable to automate the deburring process. In addition, For High-Mix Low-Volume manufacturing, setup time for a new part is a critical component of the total cost. A low setup time requires an automated method for generating a tool trajectory for the specific workpiece.

A deburring tool trajectory is traditionally planned using computer-aided manufacturing (CAM) software and a reference CAD model, or by demonstrating the tool path directly on a reference workpiece with the robot manipulator [1]. A significant source of errors with these methods is that they are based on a known reference model. With cast parts, the geometry of the workpiece varies due to uneven shrinkage and deformation that can occur during solidifying in the casting process. The uneven shrinkage and deformation can be controlled by optimizing the casting process, but it cannot be completely avoided [2]. The shape and size of the burrs also vary, and they are on average larger than in other deburring processes. These variations means that the deburring system cannot be based on reference models alone but must be adapted to each individual workpiece.

10.1016/j.procir.2023.06.087

1.1. Related research

A deburring tool path is traditionally generated off-line based on the CAD model using a CAM software. This reference path can then be registered onto the workpiece by finding the corresponding relationship between the CAD model and the physical workpiece. The most widely used method for registration of 3D shapes is the iterative closest point (ICP) algorithm [3]. The ICP algorithm minimizes the difference between two points clouds – one point cloud is a 3D measurement of an actual workpiece – and another point cloud is a sampled CAD model.

Kosler et al. [4] present a method where a tool trajectory is generated using off-line teaching. This trajectory is adapted to the workpiece by finding the transformation through ICP registering of a 3D scan of the workpiece with the CAD model. A similar approach is presented in Song and Song [5] where the tool trajectory is generated using CAM software and corrected using ICP. Both methods assume that the deformations caused by solidification can be ignored since they are not significant for the overall geometry.

Villagrossi et al. [6] present a method that generate a tool trajectory through teaching on a reference workpiece and adapt this to the actual workpiece in two steps: rough registration using ICP and compensate for deformations through a set of control points. The control points are chosen in a region where burrs are highly unlikely. This method assumes that the deformations are similar for the whole part. In Kuss et al. [7], a method is proposed where a series of CAD models are generated based on the dimensional tolerances of the workpiece. A 3D scan of the workpiece is then registered towards this pool of CAD models to find the best fit. The CAD model of the best fit is used to generate a tool trajectory using CAM software. This accuracy of this method relies of the pool of augmented CAD models. For more complex geometries, this pool must be very large to capture all possible geometric deformations.

Béarée et al. [8] and Huang et al. [2] both propose methods that segment the part into smaller sections and do registration of each sub-segment. The generated tool trajectory is then adapted to each segment. These methods assume that the deformations are constant within one segment.

An alternative method to generating a tool trajectory and adopting this to the workpiece is to generate trajectory directly on the workpiece using machine vision. In Lai et al. [9] and Princely et al. [10], a 2D vision system is used to recognize the outline of the workpiece and use this as the tool path. Both methods are based on straight line recognition and are verified on simple geometries with straight edges.

1.2. Contribution

The existing solutions handle the geometric variations of the workpiece in various ways that work for parts with simple geometry and small geometric variations. For cast parts with more complex geometry and large geometric deformations, a new method is necessary. This paper presents a pipeline that automatically generates a robot trajectory for deburring specifically generated for each individual workpiece. The method is only using the available CAD model of the part and 3D scans captured of the workpiece. The pipeline has been verified through simulation and experiment using physical robot manipulator. The paper is structured as follows: Section 2 introduce the tool trajectory pipeline and implementation, Section 3 present the verification and test of the pipeline, Section 4 discuss the presented solution and future work.

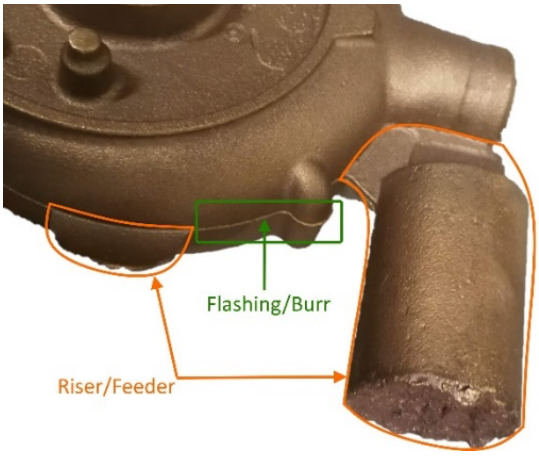

Fig. 1. Cast part with riser, feeder, and flashing/burr.

2. Tool trajectory pipeline

There are several steps to go from workpiece and CAD model to a tool trajectory. The pipeline of the process is shown in Fig. 2. Input to the system is the CAD model of the workpiece as well as the workpiece with burrs as indicated with the green boxes. A geometric representation of the workpiece to be deburred is necessary to generate a tool trajectory for the specific workpiece. This representation is found by 3D scanning the workpiece from different angles and aligning these point clouds. Once the point clouds are registered, the combined point cloud can be converted into a triangular mesh. This mesh is then used to generate a tool path. A tool trajectory is then calculated for the specified robot manipulator based on the tool path. The following sections will describe the various steps in more detail with a particular focus on the tool path and trajectory generation.

2.1. 3D scanning, registration, and burr region

The workpiece is 3D scanned from different angles to capture the whole piece with a Zivid, an industrial grade HD color 3D camera. The output from the Zivid is a point cloud containing the (x, y, z) coordinates and RGB colors. Scans captured from different viewpoints must be registered to get them aligned into the same coordinate system. There are many registration methods, one of the most common methods is the iterative closest point (ICP) method. Because of the geometric variations of the workpiece, the large burrs, and because the scans are partial, traditional registrations methods fail to give a robust and accurate solution. A new and more robust method using sparse convolutional neural networks was therefore developed in collaboration with other researchers. This new

method allows differences between the reference model and the 3D scans and allow partial scans. The method is hence robust to unmodeled geometry (burrs) and geometric deformations. This method is presented in Mohammed et al. [11]. The work on 3D scanning and registration is presented in the paper and will not be presented in more detail here.

The output from the registration step is a point cloud with all the registered 3D scans as well as a labeled burr region. This burr region is a point cloud of the points that have been labeled as being part of a burr by the neural network. This labeling is initially used to improve the registration of the scans in the network and will also be further used in the tool path generation step.

2.2. Point cloud to mesh conversion

Output from the registration is a combined point cloud of the entire workpiece. It can be computationally heavy to work with point clouds, and most geometric algorithms in computer vision and graphics operate on representations of 3D data based on surfaces [12]. It is therefore desirable to convert the point cloud into a surface representation like polygonal mesh, or more specifically triangular mesh to speed up computation.

A point cloud can be converted into mesh using a surface reconstruction algorithm. Poisson surface reconstruction is a well-known technique for creating surfaces from oriented point samples acquired from 3D scanners that is robust to noisy data [13]. Since the point cloud from Zivid only contain the (x, y, z) coordinates, it is necessary to first compute the point normals to get oriented point samples. Output from running Poisson Surface Reconstruction is a triangular mesh.

2.3. Tool path generation

Burrs are the excess material left on the cast part in the separation plane between the two molds. The tool path curve is found by taking the intersection between this separation plane and the reconstructed mesh from the previous step.

2.3.1. Implementation

The algorithms for calculating the tool path curve are implemented using Grasshopper [14] which is a visual programming language that runs within the Rhinoceros 3D CAD application. The software enables a programming approach where it is possible to visualize all geometries in an interactive and intuitive environment.

2.3.2. Intersection curve

The intersection curve is the curve along the workpiece where the two molds meet during casting, and it is along this curve the burrs are located. The intersection curve is found by intersecting the separation plane and the mesh. The algorithm for calculating the intersection is presented in Algorithm 1. The mesh (M) was found in the previous step, and the burr region (B) is the point cloud of the labeled burr region calculated by the neural network for registration. The algorithm finds the z-coordinate of the plane from the burr region, and the x and y dimension of the plane from the bounding box of the mesh. Output of the algorithm is the intersection curve (c_I) given as a polyline curve. A polyline curve is a series of connected line segments, or more formally a curve specified by a sequence of points called its vertices.

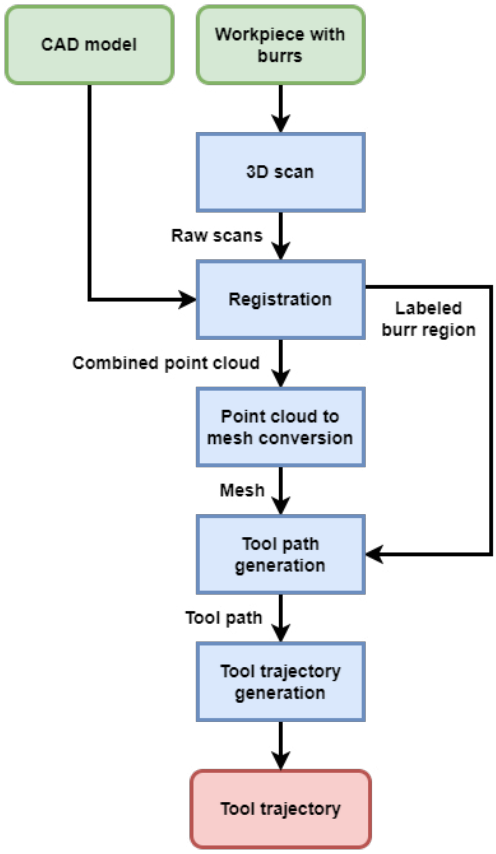

Fig. 2. Tool trajectory generation pipeline.

Algorithm 1: Intersection curve

Input:	3D mesh (M), Burr region (B)
Output:	Intersection curve (c_I)
Step 1:	Rotate M and B to fit burr region to the $x-y$ plane
Step 2:	$(X_B, Y_B, Z_B) = Decompose(B)$
Step 3:	// Calculate z-coordinate of plane $z_{average} = Average(Z_B)$
Step 4:	$BB_M = BoudingBox(M)$
Step 5:	$(x_{BB}, y_{BB}, z_{BB}) = Decompose(BB_M)$
Step 6:	// Location and dimension of plane $p_O = (x_{BB}, y_{BB}, z_{average})$
Step 7:	// Construct plane with direction $plane_B = ConstructPlane(p_O, \{1,0,0\}, \{0,1,0\})$
Step 8:	// Calculate intersection between mesh and plane $c_I = IntersectionMeshPlane(M, plane_B)$

2.3.3. Section curve

The intersection curve consists of all intersections between the mesh and separation plane with a random start and stop point of the curve. It is desirable to be able to determine both start and stop point. They do not have to be in the same point in case it is desirable to not deburr some region of the part.

The algorithm for setting start and stop point and sectioning the curve is given in Algorithm 2. The start and stop points are

set as points in Rhinoceros somewhere close to where mesh and intersection curve. They do not have to be set precisely on the intersection as this is calculated in the algorithm. The output of the algorithm is the sectioned polyline curve with modified start and stop point. This curve is the tool path for deburring.

Algorithm 2: Segment curve

Input:	3D mesh (M), Intersection curve (c_I), Start and Stop point (p_{start}, p_{stop})
Output:	Tool path (c_{tool})
Step 1:	// Finding closest point on curve to start point $(p_{c(start)}, t_{start}, D_{start}) = CurveClosestPoint(p_{start}, c_I)$
Step 2:	$(P_c, W_c, K_c) = ControlPoints(c_I)$
Step 3:	// Finding closest point in point collection $(p_{collection}, i, d) = ClosestPoint(p_{c(start)}, P_c)$
Step 4:	// Shift point collection to start at starting point $P_{shift} = Shift(P_c, i)$
Step 5:	// Reconstruct polyline curve $c_{shifted} = Polyline(P_{shift})$
Step 6:	// Finding closest point on curve to stop point $(p_{c(stop)}, t_{stop}, D_{stop}) = CurveClosestPoint(p_{stop}, c_{shifted})$
Step 8:	// Segment curve to get tool path $(c_{tool}, c_{other}) = Shatter(c_{shifted}, t_{stop})$

2.4. Tool trajectory generation

The tool path contains a description of how to get from the starting point to the end point given in cartesian coordinates. For the robot to follow the path, the inverse kinematics for the tool path must be solved.

The robot programming is implemented in the same solution as the tool path generation using Rhino and Grasshopper. In addition, HAL Robotics is used for the robot programming specific tasks. HAL Robotics is an extensible and modular software which adaptive programming of robot tasks, and motion planning for one or many robots working together. It is a robot vendor independent framework that support various robot vendors like KUKA, ABB, Universal Robots, and more.

2.4.1. Robot cell

A KUKA KR 60 is used in these experiments because it is the one available in our lab facilities, see picture of the simulated robot in Fig. 5. An HSD ES929 Spindle was chosen because it was in the default library of HAL robotics, and this is a machining application. No tool is attached to the physical robot.

2.4.2. Trajectory generation

The generated tool path is given as a polyline curve. This curve can be fed directly into a solver, together with the robot cell configuration, to solve the inverse kinematics. The movement space is set to Cartesian since the polyline curve is a set of cartesian coordinates. The generated solution can be evaluated directly in the software for feasibility and collision as well as be simulated to evaluate if the trajectory looks satisfactory.

Once evaluated, the generated solution can be exported to robot vendor specific code ready to be uploaded on the controller of the specific robot.

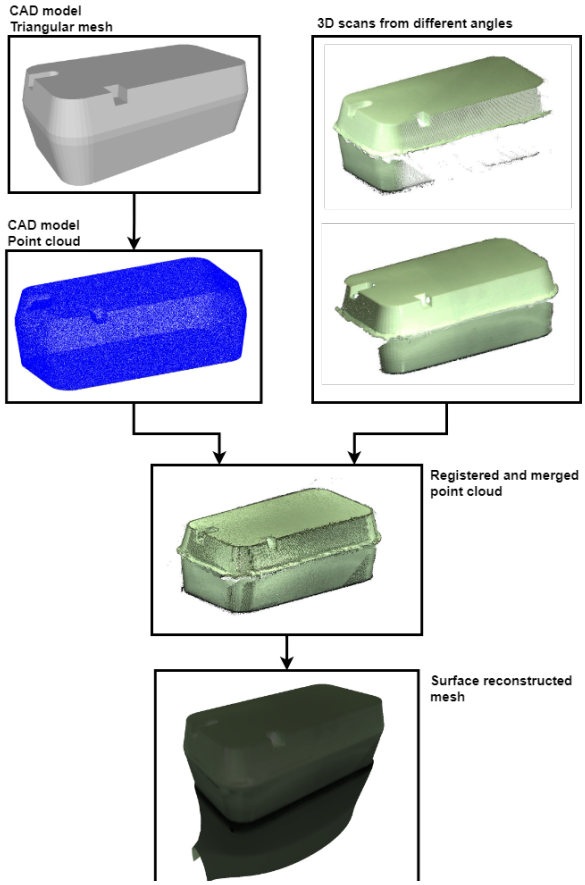

Fig. 3. The flow of geometric entities.

2.4.3. Approach and exit

A critical part om a deburring process is the beginning where the tool is approaching the workpiece. To ensure a safe approach, an approach point has been set normal to the start point 10 cm away. Another point has been set 10 cm above this approach point in the z direction. The first movement to get from an arbitrary robot starting position to the first approach point is a joint move, to remove the risk failure due to moving close to a singular position. The same movement is repeated in opposite order for the exit point. Both movements are added to

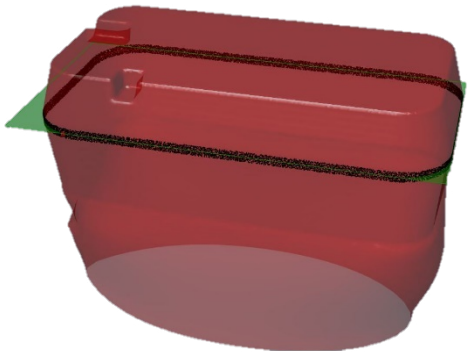

Fig. 4. Reconstructed mesh, burr region, and calculated burr plane.

the generated robot trajectory, one at the start and the other at the end.

3. Tool trajectory generation, simulation, and experiment

An experiment was performed to verify the presented pipeline. Each step of the process was tested separately with the output from each step fed into the successive step.

3.1. Test part

The cast part in figure 1 is manufactured in bronze. This part is shiny, has few distinct geometric features and is hard compared to for example polymers. For experimental purposes it was therefore designed a test part with distinct geometric features to improve registration. The part is 3D printed in PETG to make is easier to 3D scan, cheaper and quicker to produce, and less required force in machining. The test part can be seen in Fig. 3. Glue was added to the sides after 3D printing to resemble burrs.

3.2. 3D scanning and registration

The test part was scanned from different angles on a rotary board using a Zivid. A manual registration process was used since the deep learning network is still under development.

The manual processing and registration of the 3D scans was performed in CloudCompare. First, the background of the scans was removed. Secondly, the scans were registered roughly using a point picking registration algorithm. Lastly, the scans underwent fine registration using ICP towards the point cloud of the CAD model. Using ICP directly without point picking first or without the CAD model resulted in a poor solution, proving the need for an improved registration system.

The labeled burr region was labeled manually by cutting a slice from the registered point cloud. The result of both the partial scans and the registered point cloud can be seen in Figure 3.

3.3. Surface reconstruction

The manually registered point cloud from the previous step was used in the mesh generation. Both the point normals and surface reconstruction is calculated using MeshLab. The point normals are calculated using the built-in function for calculating normal for point sets. Poisson surface reconstruction is calculated using the implementation of the algorithm presented in Kazhdan and Hoppe [15]. The reconstructed mesh is shown in the last picture in Figure 3. The generated mesh matches the geometry of the test part for most of the part, while the lower section is not well matched. This is because the part is not scanned underneath meaning that the reconstruction algorithm cannot know what the part is supposed to look like underneath. Since it is the burr region that is of interest, it is not important what the part looks like underneath.

3.4. Tool path and trajectory generation

The generated mesh from the previous step and the labeled burr region was used to verify the tool path and trajectory generation. A figure showing the calculated burr plane can be seen in figure 4 where the mesh is red, burr region is black and burr plane is green. The corresponding tool path, given start and stop point around front left corner, is indicated in green. The generated tool path was fed into the trajectory generator. The corresponding robot trajectory for a KUKA KR 60 can be seen in blue in figure 5.

3.5. Experiment with physical robot

The KUKA KR 60 specific robot code exported from the generated tool trajectory was tested on a physical robot in the lab facilities. A picture of the robot can be seen in Fig. 4. During testing, it was found that it was necessary with minor changes to the exported robot code for it to run on the robot controller.

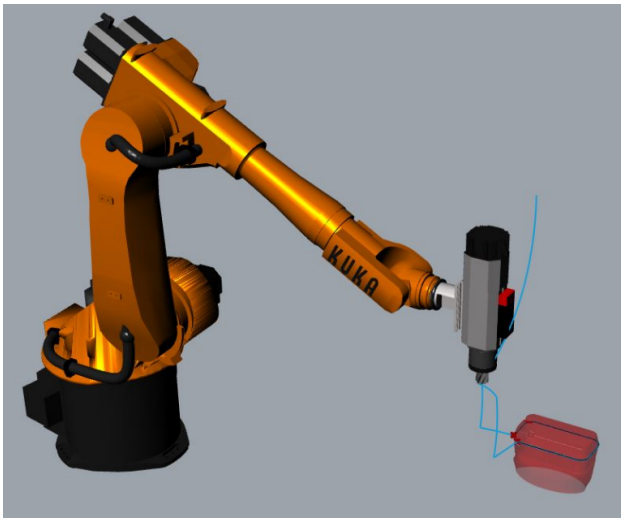

Fig. 5. Simulated tool trajectory for KUKA KR 60.

The robot followed the trajectory well. The difference between the target coordinates and the measured position (measured using KUKA RSI) was negligible in the straight sections of the curve, and ± 0.1 mm in the curved section.

4. Evaluation and concluding remarks

4.1. Calibration

Calibration have not been mentioned in the presented solution. This is because every step of the process has been tested separately and calibration has not been necessary. When every step is connected it will be necessary with camera and robot calibration, and a stiff mounting of the workpiece.

4.2. Automatic solution

The goal and value of the presented method is to be fully automatic. This means feeding in the CAD model of the part

and 3D scans and out comes a finished tool trajectory. Today, the method includes several manual steps, like registration and surface reconstruction. Future work consists of connecting every step in the process to get a fully automated pipeline.

4.3. Cascading (in)accuracy

Converting the CAD model from solid to mesh and to point cloud, 3D scanning the part, registering the scans, surface reconstruction and the robot's accuracy all affect the overall accuracy of the system. This can be referred to as cascading (in)accuracy [16]. It is important to be aware of the accuracy of each step, and how it affects the accuracy of the overall system. This has not been considered in this work and will be evaluated in future work.

Acknowledgements

The work reported in this paper was based on activities within center for research-based innovation, SFI Manufacturing, in Norway, and is partially funded by the Research Council of Norway under contract number 237900.

References

[1] I. F. Onstein, O. Semeniuta, and M. Bjerkeng, "Deburring Using Robot Manipulators: A Review," in *2020 3rd International Symposium on Small-Scale Intelligent Manufacturing Systems, SIMS 2020*, 2020, doi: 10.1109/SIMS49386.2020.9121490.

[2] W. Huang, X. Mei, G. Jiang, D. Hou, Y. Bi, and Y. Wang, "An on-machine tool path generation method based on hybrid and local point cloud registration for laser deburring of ceramic cores," *J. Intell. Manuf.*, 2021, doi: 10.1007/s10845-021-01779-y.

[3] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," *Sens. Fusion IV Control Paradig. Data Struct.*, vol. 1611, no. April 1992, pp. 586–606, 1992, doi: 10.1117/12.57955.

[4] H. Kosler, U. Pavlovčič, M. Jezeršek, and J. Možina, "Adaptive Robotic Deburring of Die-Cast Parts with Position and Orientation Measurements Using a 3D Laser-Triangulation Sensor," *Stroj. Vestnik/Journal Mech. Eng.*, vol. 62, no. 4, pp. 207–212, 2016, doi: 10.5545/sv-jme.2015.3227.

[5] H.-C. Song and J.-B. Song, "Precision Robotic Deburring Based on Force Control for Arbitrarily Shaped Workpiece Using CAD Model Matching," *Int. J. Precis. Eng. Manuf.*, vol. 14, no. 1, p. 85, 2013, doi: 10.1007/s12541-013-0013-2.

[6] E. Villagrossi, C. Cenati, N. Pedrocchi, M. Beschi, and L. Molinari Tosatti, "Flexible robot-based cast iron deburring cell for small batch production using single-point laser sensor," *Int. J. Adv. Manuf. Technol.*, vol. 92, no. 1–4, pp. 1425–1438, Sep. 2017, doi: 10.1007/s00170-017-0232-2.

[7] A. Kuss, M. Drust, and A. Verl, "Detection of Workpiece Shape Deviations for Tool Path Adaptation in Robotic Deburring Systems," in *Procedia CIRP*, 2016, vol. 57, pp. 545–550, doi: 10.1016/j.procir.2016.11.094.

[8] R. Béarée, J. Y. Dieulot, and P. Rabaté, "An innovative subdivision-ICP registration method for tool-path correction applied to deformed aircraft parts machining," *Int. J. Adv. Manuf. Technol.*, vol. 53, no. 5–8, pp. 463–471, Mar. 2011, doi: 10.1007/s00170-010-2875-0.

[9] Z. Lai, R. Xiong, H. Wu, and Y. Guan, "Integration of Visual Information and Robot Offline Programming System for Improving Automatic Deburring Process," in *2018 IEEE International Conference on Robotics and Biomimetics, ROBIO 2018*, 2018, pp. 1132–1137, doi: 10.1109/ROBIO.2018.8665148.

[10] F. Leo Princely and T. Selvaraj, "Vision assisted robotic deburring of edge burrs in cast parts," in *Procedia Engineering*, 2014, vol. 97, pp. 1906–1914, doi: 10.1016/j.proeng.2014.12.344.

[11] A. Mohammed, J. Kvam, I. Onstein, M. Bakken, and H. Schulerud, "Automated 3D burr detection in cast manufacturing using sparse convolutional neural networks," no. 3, pp. 1–12.

[12] W. A. P. Smith, "3D Data Representation, Storage and Processing," in *3D Imaging, Analysis and Applications*, Springer International Publishing, 2020, pp. 265–316.

[13] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, 2006, doi: 10.1145/2487228.2487237.

[14] S. Davidson, "Grasshopper," 2022. [Online]. Available: https://www.grasshopper3d.com/. [Accessed: 03-May-2022].

[15] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, pp. 1–13, 2013, doi: 10.1145/2487228.2487237.

[16] I. F. Onstein, C. Haskins, and O. Semeniuta, "Cascading trade-off studies for robotic deburring systems," *Syst. Eng.*, pp. 1–12, 2022.