

Multiple Object Detection and Segmentation for Automated Removal in Additive Manufacturing with Service Robots

Pascal Becker, Anastasiia Maklashevskikh,
Arne Rönnau, and Rüdiger Dillmann

FZI Research Institute for Information Technology
Haid-und-Neu-Strasse 10-14, Karlsruhe, Germany
{pbecker, maklashevskikh, roennau, dillmann}@fzi.de

Abstract. 3D printing is nowadays getting more important in industrial production plants, especially in low quantity productions. Currently, almost no printer model for fused filament fabrication (FFF) has the capability to start a new print automatically after the present one is finished. While the printed object is still on the build plate, the printer cannot continue and this is noneconomical. Manual work is required to be able to start a consecutive job. To get one step closer to full automation of the 3D printing process, the removal process should be automated process, for example with robots.

This approach presents a method to determine the number, positions and sizes of all printed objects by analyzing the G-code file of the current print job. It is determined whether the objects can be removed by a robotic arm and in which order. Furthermore, a depth camera is used to verify the hypothesis right after the print process is done. The additional verification is necessary to detect possible changes in the printed structures due to errors during the printing process. In the last step the objects are automatically removed by a robot from the build plate.

Keywords: Automated Removal, Computer Vision, Additive Manufacturing, Industrial Robotics, Robots for Industry 4.0, Intelligent Perception, Applied Robots

1 Introduction

With more need for individualisation and low quantity production 3D printing is gaining more importance. Long periods of printer inactivity, like waiting for the removal of the printed objects, are a profit loss for the company. Currently, almost no printer model uses its sensors, like cameras, to monitor the print process automatically. Therefore, manual monitoring is necessary to abort the printing process in time in case of an error or, after the print is completed, to remove the object from the build plate. To get one step closer to full automation of the 3D printing process, one should automate the removal process [1]. In almost all existing solutions for fused filament fabrication (FFF) printing, the

printed objects have to be manually removed from the printing plate or printing foil. In addition, there is no way to directly check the finished printed objects for printing errors with these processes.

An application for automated removal of multiple objects needs to be highly versatile and easy to adopt to different platforms.

The software can for example rely on available G-code data to perform the calculation about the currently printed objects and their position and sizes.

However, it is desirable to add intelligence to the system, which will also enable the extension of functionality, such as in-process monitoring for premature print error detection. In our approach, a depth camera is mounted at the tool center point (tcp) of a robot and used to verify information and hypotheses obtained from CAD data. The robot is able to move the camera to different positions for getting the best perspective according to the current print job as well as to use its gripper to remove the objects from the build plate.

By analysing the 3D point cloud the hypothesis is confirmed and the object removal is carried out, otherwise the grasp position is adjusted and in case a human has to intervene, the corresponding person is notified. The additional check with an external sensor is needed to verify the hypothesis and the finally printed objects as the structure might be different from the originally planned on. This is due to errors like layershift, spaghetti or a clogged nozzle.

Visual data is actively used to solve the problem of grasping both known and unknown objects. Most model-based methods for grasping known objects use depth data to segment the objects from the scene and then search the model database for grasp poses for these objects. If a known grasp is found it is applied to handle the objects [2].

If there is no model database, the grips have to be computed from the segmented 3D point clouds. Some approaches have been presented, based on the consideration that similar objects can be grasped similarly. In Aleotti et al. image segmentation is used to detect objects, build Reeb graphs and classify objects based on them [3]. In Herzog et al. 3D image segmentation is applied to build height maps of the objects to be grasped, which are matched with the stored templates [4]. In Rao et al. 3D image segmentation is used to identify graspable segments in the scene, determine their 3D shapes, and subsequently find grasp points [5]. In Moreno et al. 3D point clouds are used to compute local descriptors that are used to teach a robot to grasp by exploring the graspabil-

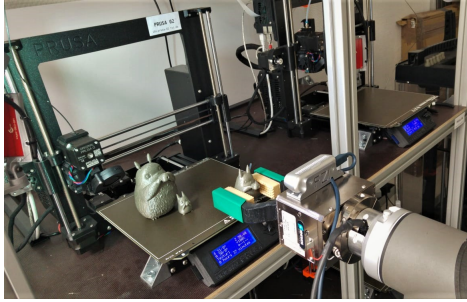


Fig. 1: Using a service robot equipped with a stereo camera and an industrial gripper to verify the hypothesis concerning the printed objects and to unload the objects from the printer automatically

ity of the object’s environment [6]. In Mousavian et al. Deep Neural Networks (DNNs) are used to generate and evaluate multiple 6-Degrees-of-Freedom (DoF) grasps directly from the 3D point clouds without additional use of descriptors or other representations of the objects [7]. The centers of mass of the objects are taken into account to increase the stability of the grasps. The problem of grasping objects positioned arbitrarily in space, as well as objects whose surfaces are composed of freeforms, is complicated and continues to be actively explored.

A number of companies have already demonstrated that the removal of 3D-printed objects from the printer is generally possible. But all of those systems are specialized to a given setup and not transferable to any other product. In 2017, Voodoo Manufacturing presented the Skywalker project [8] where a robotic arm operates several FDM 3D printers. Once the object is printed, the robot removes the complete build plate, places it in the rack, takes a new empty build plate from the supply rack and places it in the printer. In the same year, Formlabs announced its FormCell [9]. In a self-contained system, a robot mounted on linear axes, operates several stereolithography (SLA) 3D printers that use different photopolymers depending on the application. When a print needs to be started, the robot takes a build plate from the rack and inserts it into the printer. After the print is done, the robot removes the plate, places it in a rinsing device with the printed objects for post-processing, and finally places the plate in the rack. Solutions without additional robots exist also partially. In 2017, Stratasys introduced the Continuous Build 3D Demonstrator [10]. Each FFF printer in the demonstrator is equipped with a container and the objects are not printed directly on the build plate, but on a special foil. After the part is finished, the foil is rolled further and cut off. Still sticking on the foil, the object is falling into the container for manual post-processing.

BlackBelt products have also been on the market since 2017. The build plate from a BlackBelt printer is a conveyor belt, which in principle allows infinite FFF prints as well as prints of more complicated shapes without additional supports [11]. The removal of the object is also solved by the fact that the object is pushed further on the conveyor belt and at the end has no contact points with the printing plate anymore.

In 2018, start-up Triditive presented its solution for FFF printing [12]. At the center of the complete Amcell block is the conveyor belt. The FFF printers, which can print both plastics and metals, are placed around it. After the printing is finished, the complete build plate is pushed onto the conveyor belt line and transported out. But removing the build plate leads to calibration issues when a new plate is mounted.

In the same year, Becker et al. proposed a solution in which a lightweight robot removes objects from printers without having to remove the entire plate and insert a new one [13]. For this purpose, they also developed a universal gripper that can grasp objects of different shapes.

In 2019 the NextGenAM project was completed, which was launched in 2017 by Daimler AG in cooperation with Premium AEROTECH and EOS [14]. The core of the pilot production chain for additive series manufacturing is a four-

laser system for metal-based industrial 3D printing. Since metal powder is used in a sealed cell, it is difficult to use a robotic arm to remove the objects. For this purpose, the complete printing cell is moved away by an automated guided vehicle (AGV) and handed over to further automated components for removing the remaining metal powder, post-processing and quality control. In 2019, the IDAM (Industrialization and Digitization of Additive Manufacturing (AM) for Automotive Series Processes) research project started, also aiming to integrate 3D printing into industrial processes [15]. Even there are some approaches they either need specially build printing hardware or are only able to handle on object per print job. In this paper we are going to demonstrate that the segmentation, detection and removal of multiple objects per print is possible.

2 Approach

Using the whole build plate of a printer by printing multiple objects at once is effective and can easily take the printer several hours until the job is done. But currently there exists no way to automatically remove these objects after a successful print without using any special printer hardware and to start the next job directly afterwards. This is done manually until now.

The proposed approach is able to calculate the position of each object from the G-code and to move a robot in a way that all objects are removed automatically one after each other. It consists of two aspects: Building the hypothesis of the printed objects based on the G-code and verifying the hypothesis based on a measured point cloud. Afterwards, the service robot is able to remove consecutively all objects depending on the hypothesis for each object.

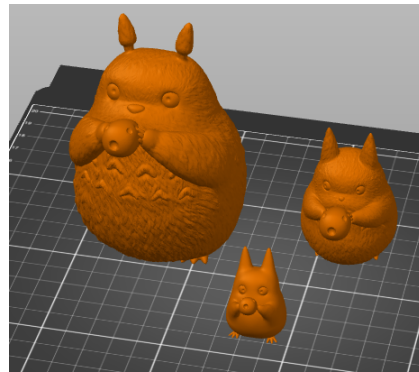


Fig. 2: Example stl files placed on the build plate and the basis for the generated G-code

2.1 Determination of the Object Hypothesis based on G-code

Determination of the Amount of Objects The structure of a G-code file allows very simple extraction of the (X, Y, Z) coordinates belonging to the objects by using the regular expressions. To find the position of all objects all (X, Y, Z) coordinates are projected onto the (X, Y) plane. In doing so, the data set remains meaningful but can be easily clustered by standard algorithms.

Cluster analyses are procedures for discovering similarities in usually very large data sets. In the context of parsing a G-code file, similarity is defined as the membership of (X, Y) points to the same object. It is unknown to a robot in advance how many objects are printed on a build plate at the same time.

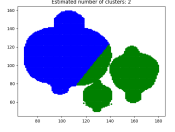
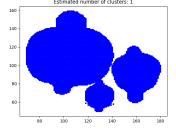
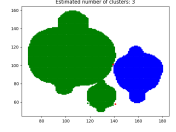
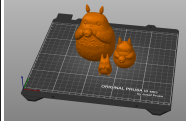
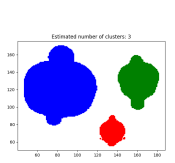
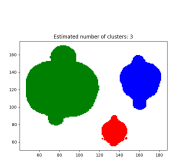
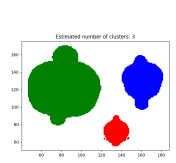

	MeanShift	DBScan	OPTICS	STL-file
Amount of clusters	2	1	2 (+ outliers)	
Graph				
Amount of clusters	3	3	3	
Graph				

Table 1: Different clustering algorithms *without* (top) and *with* (bottom) the minimum distance between objects in the G-code file

Therefore, only the algorithms that automatically detect the number of clusters without passing them as a function parameter are suitable for this recognition task. Parameters for Birch would have to be entered manually for each print job. Only MeanShift [16], DBSCAN [17] and OPTICS [18] were found to be suitable. By iteration, the minimum distance between printed objects should be at least 20 mm on the (X, Y) plane. Table 1 shows the results of clustering with different algorithms and distance width. In the first row, no minimum distance was set. In the second case, the minimum distance was 20 mm. During further testing, 20 mm minimum spacing was found to be optimal. With 10 or 15 mm spacing, clustering still gives wrong results.

Determination of the Camera Orientation To get an optimal camera perspective onto the object later on, a suitable camera orientation is calculated in advance. By means of linear regression a straight line is found which best approximates the cluster points on the (X, Y) -plane. This can be done by approximating either all (X, Y) coordinates or, for simplicity, only coordinates of the cluster centers. Gradient k of the calculated straight lines is approximately the same in both cases as shown in Fig. 3. The parameter k is adopted for the positioning of the tcp: the X axis of the camera’s coordinate system should be as parallel as possible to the calculated line.

Determination of Object Center or Center of Mass In the MeanShift method cluster centers are calculated automatically, so no additional determination is needed. Thus, the (X, Y) coordinates of the respective object center are already available. When using DBSCAN or OPTICS cluster centers have to be calculated explicitly. The corresponding Z -coordinate is determined by re-parsing the G-code file. For this, the search is restricted to the coordinates of

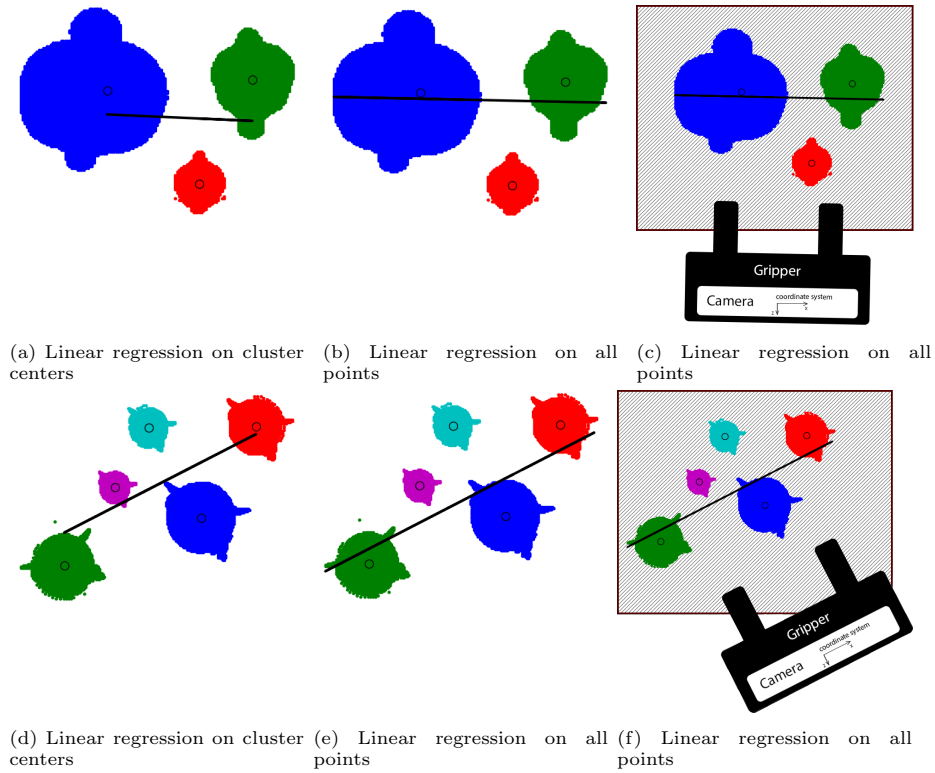


Fig. 3: Difference in gradient for the camera perspective depending on the amount of objects and their placement

the members of the corresponding cluster, which is possible by determining the constraints such as x_{max} , y_{max} , x_{min} , y_{min} .

After the parsing is finished, (X, Y, Z) coordinates belonging to the object are available. Extracting the Z -coordinates and adding them to a set leads to an ascending sorted list due to the structure of the G-code file, in which each Z -coordinate occurs once. The mean value of this list is the needed Z -coordinate Z_{center} . Combining this with the (X, Y) coordinates of the cluster center provides the point that the robot's tool center point is expected to approach for grasping.

However, the object center is not necessarily the center of mass. To increase the stability of the grip, it is useful to find the object's center of mass assuming that the object is not hollow. For this purpose, a coordinate $Z_{mass} \in Z_{obj}$ is found such that the difference between the sum S_{under} of the cross-sectional areas for $Z_i < Z_{mass}$ and S_{over} of the cross-sectional areas for $Z_i > Z_{mass}$ is the smallest.

Since arbitrary complicated shapes are used, it is not possible to calculate the exact cross-sectional areas. Therefore, the areas are reduced to the circumscribed quadrilaterals or circles. For each Z coordinate of the object x_{max} , y_{max} , x_{min} , y_{min} are determined.

If the area is to be reduced to a quadrilateral, the area is calculated as.

$$A = (x_{max} - x_{min}) \cdot (y_{max} - y_{min}).$$

If the area is to be reduced to a circle, the radius of the circle is

$$r = \frac{\max((x_{max} - x_{min}), (y_{max} - y_{min}))}{2}$$

and the area thus

$$A = \pi \cdot r^2.$$

Finding Z_{mass} and combining it with (X, Y) coordinates of the cluster center, $(X_{mass}, Y_{mass}, Z_{mass})$ provides the predicted coordinate for the TCP of the robot arm.

Determining the Dimension of the Object on the Grasping Plane After Z_{mass} is found, the closing width for the gripper is determined. When the software stack is started, one of the parameters passed to the G-code analysis is the height of the gripper's fingers h .

Let d be the width of the object on the plane $Z = Z_{mass}$ and h be the height of the fingers of the gripper. Let d_1 be the width of the object on $Z_1 = Z_{mass} - \frac{h}{2}$ and d_2 be the width of the object on $Z_2 = Z_{mass} + \frac{h}{2}$. Let the widths d_3, \dots, d_n be the widths on all Z_n between Z_1 and Z_2 ($Z_n \in Z_{obj}$). The final closing width for the gripper is then $width = \max(d_1, \dots, d_n)$. Fig. 4 and 5 illustrate the need for this calculation. Fig. 5 also shows why the center of mass should be preferred to the object center.

The width d_i for $Z = Z_i$ can then be determined as follows: $d_i = X_{max,i} - X_{min,i}$, where $(X_{max,i}, Y, Z_i)$ and $(X_{min,i}, Y, Z_i)$ are points of the object on the plane $Z = Z_i$ and $Y \leq Y_{mass}$ holds. If the width of the object is allowed, it does not immediately mean that the center of mass of the object can be approached by the TCP. If the distance between y_{center} and y_{min} is greater than the length of the gripping surfaces of the gripper fingers, the point to be approached must be moved along the Y axis.

Determination of the Possibility to Grasp There are mainly the following situations in the approach, where the removal of an object is not possible:

- If the required closing width $width_{closed}$ of the gripper minus 16 mm is larger than the maximum gripper opening, the object cannot be removed. When approaching the object, the gripper jaws must be opened further than the object width to avoid collision with the object. A distance of $2 \cdot 8 \text{ mm} = 16 \text{ mm}$ is included to prevent damage.
- If the closing width of the gripper is smaller than the minimum gripper opening, the object cannot be removed. The minimum gripper opening is 0 mm, but the distance between the gripper jaws in this situation is not necessarily 0. For the gripper jaws used here, the distance between the gripper jaws in the closed state is 6.5 mm.

- If the height of the object is less than 20 mm, the object cannot be removed as the robot could damage the printer by hitting the build plate.
- If the object A with allowed width in the projection on (X, Z) plane is covered by another object B that cannot be gripped, the object A cannot be picked either.

If the current grasping scenario does is none of the cases above, the robot is able to unload the object automatically.

Determining the Order of Object Removal Since the objects are extracted parallel to the X -axis, the order of extracting the objects based on Y -coordinates of the object centers can be determined very easily. For this purpose, the Y -coordinates including labels of the cluster centers are extracted and sorted in ascending order with respect to Y -coordinates. The resulting sequence of labels is the order of extraction of the objects.

Creation of the Object Hypothesis After the G-code analysis is finished, the hypothesis is calculated, which is further verified by the analysis of the point cloud algorithm. For each object, the following information are merged into one structure:

- $X_{min}, X_{max}, Y_{min}, Y_{max}, Z_{max}$ - array of the object's edge coordinates in each of the three dimensions;
- Coordinates of the center of mass of the object in mm - a tuple $(X, Y, Z)_{mass}$;
- Decision whether the object can be grasped *can_grasp* - *true* or *false*;

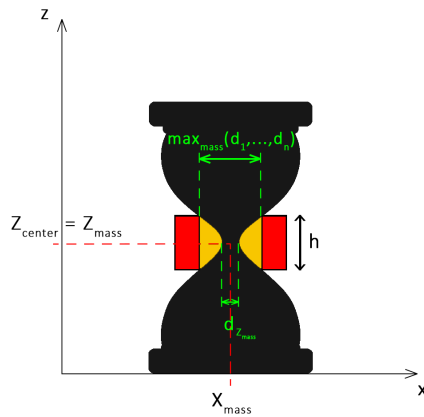


Fig. 4: Example: No possibility to move the grasp towards dz_{mass} as the gripper brackets are too wide for the calculated position

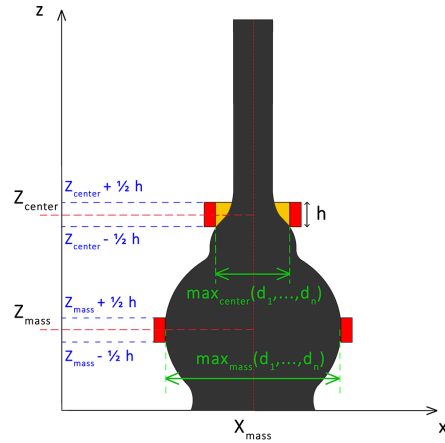


Fig. 5: Comparison of the grasping points on Z_{center} and Z_{mass} : the point on Z_{center} has fewer contact points and is therefore less stable

- Coordinates to approach with the Center Point tool in mm - a tuple $(X, Y, Z)_{grip}$;
- Object width on the grasping plane obj_width in mm;
- Closing width for the gripper $width_closed$ in mm.

The structures for each detected object are arranged in the order of extraction and, together with the number of objects and the gradient of the regression line, form a hypothesis message that is passed to the camera data analysis.

2.2 Verification of the Object Hypothesis Based on a Point Cloud

Determining the Number of Objects on the Build Plate The start position is approached with the camera. A single frame is captured and processed with the cropbox algorithm [19] which removes all points that are outside a fixed box. Further, the largest planar components that do not belong to the target objects are removed and all remaining points are clustered. The number of resulting cluster clouds is the measured number of objects on the printing plate. If this number is identical to the number from the hypothesis, we can proceed to checking the tangibility and removal of the objects.

Checking the Graspability and Removal of the Objects For checking the possibility to grasp, a camera position is approached for each object for which $can_grasp = true$. The X -axis of the camera is parallel to the X -axis of the build plate, coordinate origin of the camera is guided to the X -coordinate of the object center. Each object should be viewed individually, as smaller objects in the background may be obscured by the larger objects in the foreground.

After the camera has been moved to the inspection position, a single frame is again captured and processed as described. Further, only the cluster cloud containing the points whose X and Z coordinates are in the range of the object's boundary points is considered.

It may happen that there are protruding fine details on the proposed grasping plane, which could possibly break during grasping. To avoid this, the point cloud should be analyzed for the presence of such details. Experimentally, it was found that such fine details do not appear in the cluster cloud. They appear as reflections at the edge of the original point cloud and are thus already removed during filtering.

First, the width d of the object between $Z_{grip} - \frac{h}{2}$ and $Z_{grip} + \frac{h}{2}$ is calculated from the cluster cloud. Due to the incompleteness of the point cloud, this number will differ from the number from the G-code analysis. If $d \geq 0.8 \cdot obj_width$, the object can be grasped as proposed in the hypothesis. If $d < 0.8 \cdot obj_width$, the grasping plane is moved down or up one gripper finger width if the possibility exists. This is necessary as the gripper will not get in contact while closing otherwise. The object width for the new grasp position is determined from the cluster cloud. If the object can be gripped on the new gripping plane, it is removed and moved to the storage location. The procedure is repeated for each object. If all objects could be removed, it is checked whether the build plate is empty and then the printer receives the signal that a new print job can be started.

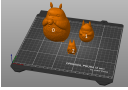
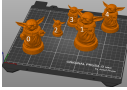
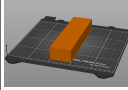
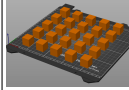
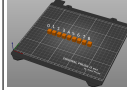
	Test #1	Test #2	Test #3	Test #4	Test #5
STL file					
Number of clusters	3	5	1	25	4
Gradient of regression line	-0.05070884	0.48840319	0.00567	0.003656	-
Successful	true	true	true	true	false

Table 2: Test files with multiple objects and different amounts of objects. Tests 1 to 4 where successful, whereas test 5 failed as the objects were too close to each other.

3 Experiments and Results

At first several tests were performed to verify the function of the G-code analysis module. The range of the number of objects in each G-code file was between 1 and 25, and the range of object sizes was between 10 mm x 10 mm and 80 mm x 80 mm (width x depth). The object heights were between 10 mm and 15 mm. In the following tables, results of some experimental runs are presented.

Test file #1 contains 3 objects with freeform shapes of different sizes that have a minimum distance of 20 mm from each other on the (X, Y) plane. The order of extraction was calculated to $[2, 0, 1]$ describing the labels of the objects. Data that the G-code analysis had yielded are listed in Table 2. For the object with label 2, the center of mass is at $Z = 14.45$ mm, which is below the threshold of 20 mm. However, the height of the object is 35.15 mm, which is above the threshold of 20 mm, so the object can be picked after all. It should be picked at $Z = 20$ instead of at $Z = 14.45$ mm. The analysis recognizes this. For the object with label 2 $(X, Y, Z)_{center} = (134.83, 72.21, 14.45)$, $(X, Y, Z)_{grip} = (134.83, 72.21, 20)$ was finally calculated.

Test file #2 contains 5 objects, all of which have the same model, but have been merged into the STL file at different sizes. On the (X, Y) plane, the minimum distance of 20 mm is also observed. The order of removal was calculated to be $[0, 1, 2, 3, 4]$.

Test file #3 contains one object of a simple shape (block) which depth is much greater than its width, so the distance between the y_{center} and y_{min} is greater than the length of the gripper jaws. For the object $75 \text{ mm} < (y_{center} - y_{min} = 102 \text{ mm})$ and $z_{center} < 20 \text{ mm}$, therefore y_{grip} as well as z_{grip} are adjusted.

The test file #4 contains 25 equal cubes with $20 \text{ mm} \times 20 \text{ mm}$ placed in 5 rows of 5 cubes each with a minimum distance of 20 mm. All 25 objects were detected by the G-code analysis. They should be grasped at $z_{grip} = 20 \text{ mm}$.

Test file #5 contains 9 cubes: $10 \text{ mm} \times 10 \text{ mm} \times 10 \text{ mm}$, placed in a row with the spacing of around 5 mm. The G-code analysis detects only 4 objects, clustering multiple cubes into one object. The wrong result of clustering leads to a wrong hypothesis.

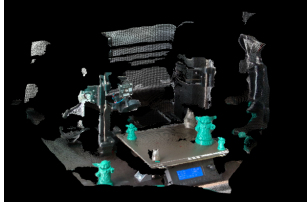
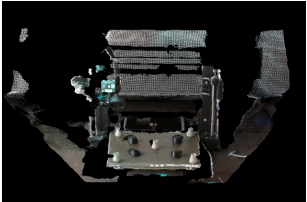
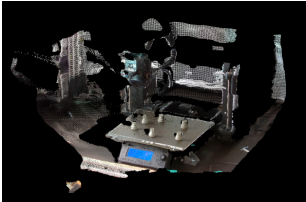
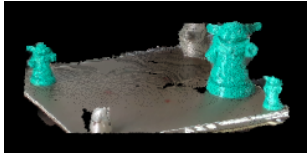

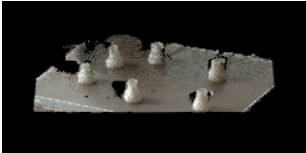
Left perspective	Front perspective	Right perspective
Original cloud		
		
Filtered cloud		
		
Number of detected objects		
5	9	6

Table 3: Tests objects of different form and amounts to check the filter results and the amount detection

Visual Data Analysis The verification of the hypothesis as well as the graspability were tested on several sets of objects. The test runs were performed on sets of objects whose arrangement on the printing plate require different starting positions. Table 3 shows the images of three test scenes and the corresponding filtered point clouds. In all tests the number of objects was successfully detected.

Examination of graspability In this test, object #1 from test #1 was printed and the information known from the G-code analysis were passed to the robot as well as to the CountObjects service. The test position was approached and the service was invoked. For clarity, resulting point clouds were saved and visualized. On the cloud filtered with CropBox (Fig. 6a) it is easy to see that the thin fine parts of the object cannot be found in the point cloud due to the sensor’s limitations. Next, the planar component of the point cloud was removed and the object was presented as a cluster cloud (Fig. 6b). The coordinates x_{tcp} as well as y_{tcp} , which were calculated by the G-code analysis, were used to determine if the cluster belonged to the object of interest. The cluster cloud was then cropped (Fig. 6c), and the width was calculated.

The width calculated by the G-code analysis was 40 mm. The width measured in the point cloud was 39 mm, which is 97.5% of the value from the G-code analysis. Thus, the object can be grasped as calculated in the hypothesis.

Full System Test To evaluate the whole approach all presented methods were tested together. After a successful generation of the initial hypothesis, this was verified with the point cloud and the calculated grasping point was given to the

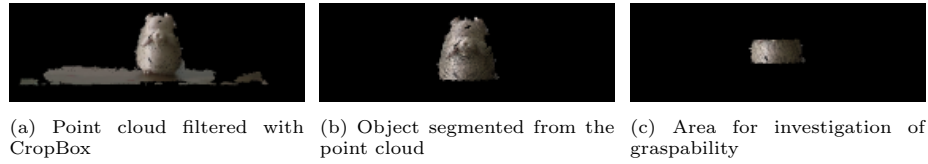


Fig. 6: The point cloud is pre-processed in multiple steps to get the best possible input for the graspability algorithm

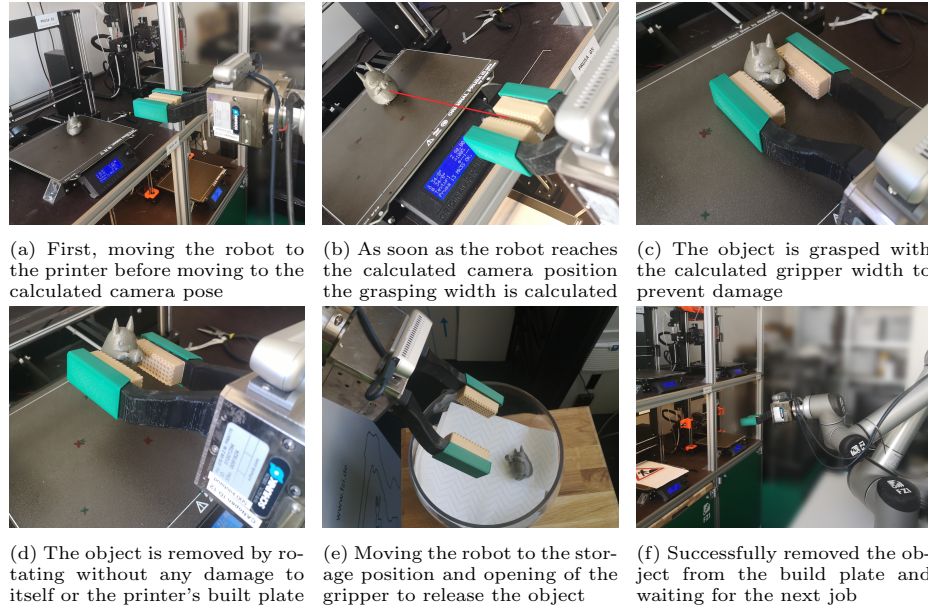


Fig. 7: Image sequence of the different intermediate steps of the presented approach to automatically remove printed objects

grasping mechanism. This moved the robot with the correct gripper opening width to the correct point and closed the gripper. Afterwards the object was placed at the storage position. The tests were successful and all objects were removed without any damage. Fig. 7 represents different steps of the whole system test.

4 Conclusion and Outlook

This work represents a method for removing multiple objects from a 3D printer using a robotic arm was developed. The method consists of two parts: a G-code analysis to generate the hypothesis for future removal, and a camera data analysis to verify this hypothesis. Both parts were tested extensively both separately and in collaboration. A robot is used to move the camera to an optimal viewing point as well as removing the printed object with its gripper.

During several tests, the G-code analysis proved to be stable and provided good results. The placement of the objects must also take into account the width of the gripper jaws and a distance for each gripper jaw.

Whether the analysis of the camera data for counting the objects gives correct results depends on the quality of the 3D point cloud. In most cases, the test results agreed with the expected results. However, if the objects are too close to each other and/or overlap from the camera's point of view, in some cases they are detected as a single object. This could be solved, for example, by using different hardware or a change of the perspective according to the current scenario. Calculating the width of the object gave mostly stable results, but again the quality of the point cloud plays a major role.

On the one hand, to get a better point cloud, in the future, one could take several shots of the printing plate with objects from different angles and merge the resulting point clouds to get a more complete point cloud. On the other hand, the use of better sensors can improve the quality of the point cloud. However, the object width check must be adjusted, otherwise there is a likelihood that fine parts that should be removed will be assigned to the object point cloud after all. The adjustment can be done by using morphological operations like erosion and dilation in 3D. The difference between the object widths before erosion and after dilation is then the basis of the statement whether the object can be gripped on the calculated grip plane.

Currently, the 3D image of the scene with the printer and objects is cropped with CropBox filters, leaving only the printing plate and objects in the point cloud to be analyzed. Although this is currently sufficient, it will be improved in the future to be able to set the filter correctly for all possible camera positions. Another possibility is to create a RANSAC [20] model for the body of the 3D printer so that SACSegmentation can capture and remove the points that belong to the printer (frame, axes, display, print head). If creating such a model is not possible, one can try to match the 3D model of the printer with the point cloud, identify the printer in the point cloud this way and remove it from the point cloud. Overall this work presents a chance of an easy integration of a service robot in an already existing additive manufacturing plant as there is almost no need to change the existing hardware.

References

1. A. Weber, "Additive Manufacturing: The Quest for Automation," 2019, [Online]. Available: <https://www.assemblymag.com/articles/95074-additive-manufacturing-the-quest-for-automation>.
2. A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour, and S. Schaal, "Learning of grasp selection based on shape-templates," *Autonomous Robots*, vol. 36, pp. 51–65, 2014.
3. J. Aleotti and S. Caselli, "A 3d shape segmentation approach for robot grasping by parts," *Robotics and Autonomous Systems*, vol. 60, pp. 358–366, 2012.
4. A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal, "Template-based learning of grasp selection," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 2379–2384.

5. D. Rao, Q. V. Le, T. Phoka, M. Quigley, A. Sudsang, and A. Y. Ng, "Grasping novel objects with depth segmentation," in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 18-22 2010.
6. P. Moreno, J. Hörnstein, and J. Santos-Victor, "Learning to grasp from point clouds," Department of Electrical and Computers Engineering, Instituto Superior Técnico, Portugal, Tech. Rep. Vislab-TR001/2011, September 2011, (technical report).
7. A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2901–2910.
8. C. Scott, "Voodoo Manufacturing Tells Us About Automating 3D Printing With Project Skywalker," 2017 (abgerufen am 16.07.2020), [Online]. Verfügbar: <https://3dprint.com/167938/project-skywalker-voodoo/>.
9. Form Labs, "Form Cell: Ein Blick in die Zukunft der Produktion," (2017), [Online]. Available: <https://formlabs.com/de/3d-printers/form-cell/>.
10. Stratasys, "Meet the Stratasys Continuous Build 3D Demonstrator," 2017, [Online]. Available; <https://www.stratasys.com/de/demonstrators>.
11. S. Schürmann, "Blackbelt 3D Printer," 2017, [Online]. Available: <https://blackbelt-3d.com/>.
12. S. Goehrke, "TRIDITIVE's Fully Automated 3D Printing System," 2018, [Online]. Available: <https://www.fabbaloo.com/blog/2018/12/27/triditives-fully-automated-3d-printing-system>.
13. P. Becker, E. Henger, A. Roennau, and R. Dillmann, "Flexible object handling in additive manufacturing with service robotics," in *2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)*, 2019, pp. 121–128.
14. M. Grebner, "NextGenAM Automated 3D Printing a Complete Success (press release)," 2019, [Online]. Available: <https://www.eos.info/en/press-releases/next-gen-am-serial-3d-printing-project-end>.
15. P. Nolis, "BMBF-Forschungsprojekt IDAM: Netzwerk bringt metallischen 3D-Druck auf automobilen Serienkurs (Pressemitteilung)," 2019, [Online]. Available: <https://www.research-news.org/2019/04/17/bmbf-forschungsprojekt-idam-netzwerk-bringt-metallischen-3d-druckauf-automobilen-serienkurs/>.
16. Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 17, no. 8, pp. 790–799, 1995.
17. M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.
18. M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.
19. R. B. Rusu and S. Cousins, "PCL API Documentation," [Online]. Available: <http://pointclouds.org/documentation/index.html>.
20. M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.