CAD-based Grasp and Motion Planning for Process Automation in Fused Deposition Modelling

Andreas Wiedholz[1], Michael Heider[2][a], Richard Nordsieck[1], Andreas Angerer[1], Simon Dietrich[3] and Jörg Hähner[2][b]

[1]*XITASO GmbH IT & Software Solutions, Augsburg, Germany*

[2]*Organic Computing Group, University of Augsburg, Germany*

[3]*Faculty of Electrical Engineering, University of Applied Sciences Augsburg, Germany*

{*andreas.wiedholz, richard.nordsieck, andreas.angerer*}*@xitaso.com*, {*michael.heider, joerg.haehner*}*@uni-a.de, simon.dietrich@hs-augsburg.de*

Keywords: grasp planning, motion planning, planning, robotic pick, automation, ROS, digital twin

Abstract: Planning the right grasp pose and motion into it has been a problem in the robotic community for more than 20 years. This paper presents a model-based approach for a *Pick* action of a robot that increases the automation of FDM based additive manufacturing by removing a produced object from the build plate. We treat grasp pose planning, motion planning and simulation-based verification as separate components to allow a high exchangeability. When testing a variety of different object geometries, feasible grasps and motions were obtained for all objects. We also found that the computation time is highly dependent on the random seed, leading us to employ a system of budgeted runs for which we report the estimated success probability and expected running time. Within the budget, some objects never found feasible picks. Thus, we rotated these objects by 90° which lead to a substantial improvement in success probabilities.

1 INTRODUCTION

Planning grasps and motions are widely researched problems in the robotic community, often addressed independently due to their great complexity (Fontanals et al., 2014; Ghalamzan E. et al., 2016). However, this renders the consideration of mutual influences impossible and results in a missing plausibility check. For example, if grasping is planned, the resulting pose might be unavailable due to a restricted kinematic. A grasp planner determines a suitable pose for the robot gripper in relation to an object it is supposed to grasp. This pose should lead to a grasp forming either a positive or non-positive connection which secures the object during movement. The motion planner, in contrast, determines a motion from a start pose to a target pose. If the robot has to do a *Pick and Place* manoeuvre, the robot first adopts the target pose before being able to grasp the object. Subsequently, the robot moves it to the desired location before releasing it.

In this paper, we present a new architecture to determine grasp poses as well as motions to perform a Pick manoeuvre. Since initial experiments have shown that in our case study *Place* manoeuvres are trivial, we focused our evaluation on the *Pick* manoeuvre. We consider both robot and object location as well as its orientation to be known and the environment to be free of unexpected obstacles. Although object geometries can vary widely, they are known prior to planning through exact CAD models. We situate our approach at a **CA**D based g**r**asp and m**o**tion p**l**anning system (CAROL) in the context of fused deposition modelling (FDM)—an additive manufacturing (AM) technique. Here, we encounter a large variety of objects that need to be grasped differently. However, as producing an object takes at least 15-20 minutes (and sometimes substantially longer), sufficient time remains for planning anew. Additionally, in FDM, the exact CAD-models are always available as well as the information about the position and orientation of the object on the build plate and the target position. Furthermore, the position and orientiation of the object on the build plate can both be set freely.

In AM objects are rarely produced in large quantities and the variety of objects is considerable. There-

[a] https://orcid.org/0000-0003-3140-1993

[b] https://orcid.org/0000-0003-0107-264X

fore, programming the robot manually for each object (or even type of object) is not feasible. While in recent years numerous approaches using machine learning for grasp planning have been studied (Zelch et al., 2020), they share difficulties of generalizing to previously unknown objects and are typically trained on domain specific datasets as shown by Zunjani et al. (2018) and Depierre et al. (2018). Since all information is known a priori in our use case, we are able to circumvent these limitations by relying on a completely model-based approach.

In our evaluation we combine an explicit grasp planning with an explicit motion planning based on the exact 3D models of robot, environment and graspable object. Explicit in this context means that we do not estimate poses (Chen et al., 2018) or execute random motions (Garg et al., 2020) towards a desired pose. In theory, however, our approach does not rely on this explicitness as we are able to validate the planned motions and grasps through our simulation which constitutes a digital twin.

2 RELATED WORK

In the past years, the state of the art in grasp planning changed more and more from CAD-based (Miller et al., 2003; Gatrell, 1989) to computer vision (Zhao et al., 2019; Huang et al., 2020) and machine learning (ML) (Zunjani et al., 2018; Depierre et al., 2018) based approaches. One possible reason for this trend is the shifting of robot applications and their use cases in recent years. In many current applications, the robot has to recognize and pick up unsorted objects lying on a desk in front of the robot (Tanwani et al., 2019; Lee and Lee, 2020). In these scenarios, available information about the object geometry or position is insufficient to manipulate it using a robot arm. Therefore, a common approach is to use cameras and computer vision to acquire the missing information.

Also, new technologies, big databases and well trained networks help to improve the performance and success rate of ML-based robot applications. Neural networks, in particular Convolutional Neural Networks (CNNs), are being used to predict grasp configurations in robot applications. Mahler et al. (2018) developed Dex-Net, which relies on CNNs to plan robust grasps. With the help of this network, it was possible to execute grasp planning with a cellphone and an augmented reality app (Zhang et al., 2020). Furthermore, Kumra and Kanan (2017) present a robotic grasp detection system that uses an RGB-D image of a scene to extract features of the environment and then uses a CNN to predict the grasp configuration for the object to be grasped. Another approach that uses learning systems is from Bohg and Kragic (2010). The authors train a machine learning model from several examples and test it on novel objects. With their experiments, they show that a non-linear classification algorithm with shape context can lead to a stable detection of grasping points for many different objects.

Model-based approaches require exact knowledge of the environment, including all possible obstacles if the robot is not equipped with collision avoidance, which is a cost factor in real-world problems. Hou et al. (2018) tried to achieve a more stable grasp by pivoting and compliant rolling (rolling a grasped object over the surface to improve the grasp quality) after the gripper first grasped the object. To achieve lower computation times Goldfeder et al. (2007) simplify CAD models with the help of decomposition trees and superquadrics. Superquadrics are objects with a simple geometry such as a cylinder or cuboids. The authors first abstract the complex object with only one superquadric. The abstraction is iteratively refined until a satisfactory precision is reached by adding more superquadrics. Then they start to plan and execute the grasp.

For real world applications not only grasp planning but also motion planning is required. Depending on the use case you may dispense with an exact grasp planning and use a grasp pose estimation instead. Huang et al. (2020) implemented a moving object tracking algorithm that detects and grasps an object that moves on a conveyor belt setup or with a random motion. One possibility for planning a movement is to use a Voronoi diagram as seen by Dorn et al. (2020): In their paper, they present the generation of Voronoi diagrams using a voxelization of the surface with a GPU render approach. The accuracy of their diagram for subsequent path planning meets requirements of only a few millimetres tolerance.

Many approaches use point clouds to acquire information about the object and use it to calculate a feasible target pose (Gonçalves and Lima, 2019; Rosa et al., 2016). These sorts of approaches often use RGB-D cameras to generate these point clouds of the object for planning. As shown by Chen et al. (2018), some information about the object is lost when only using point clouds even if there are two cameras installed. Garg et al. (2020) describe a system that helps one-armed people to grasp objects from their environment by marking an object the robot is supposed to grasp with a laser pointer. With a stereo-camera and OpenCV, information about the object is gathered in the form of point clouds and used in the motion planning with ROS (Quigley et al., 2009) and MoveIt!

(Görner et al., 2019). Huh et al. (2018) present a sampling-based motion planning method for *Pick and Place* manoeuvres with high DoF manipulators. Their approach efficiently selects the optimal grasp pose for manipulators with redundant degrees of freedom, thus achieving much lower computation times than other sampling-based motion planners. In an ROS context, MoveIt! is a widely used tool for CAD and sampling based motion planning. The latest version of MoveIt! offers a continuous collision detection[1] and a grasp planner[2] which uses Dex-Net as described by Mahler et al. (2018). We provide the same functionality but enable exchangeability of specific components e. g. if a non-integrated component is anticipated to lead to better results. In addition, we use the exact 3D models of all objects involved in the process for grasp as well as for motion planning.

As the additive manufacturing scenario provides us with exact models, we have more detailed information available than computer vision based estimation methods (Chen et al., 2018), which we utilize to generate better grasps. Another method that utilizes 3D models to improve picking and detaching target objects in an additive manufacturing scenario is shown by Becker et al. (2020). They use CAD models (i.e. STL files) and combine them with GCODE—FDM printer instructions that are used to print each layer of the object—to generate collision free trajectories. In contrast to this approach we first have to calculate target robot poses because we have to grasp objects we do not have precalculated good grasp poses for. Other approaches like (Aroca et al., 2017) use a 2 DOF manipulator connected to a computer vision system to push the printed object from the build plate. Their approach has limitations (e.g. obstacles in the environment and lighting variations) which we addressed by using more DOF and incorporating the 3D model. In contrast to Fontanals et al. (2014), we separate grasp and motion planing, rather than integrating them. This provides the benefit that if either grasp planner, motion planner or simulation do not work perfectly for a specific use case or scenario, they are easily exchangeable. This is aided by defining abstract interfaces, each of which aims to address individual sub-problems.

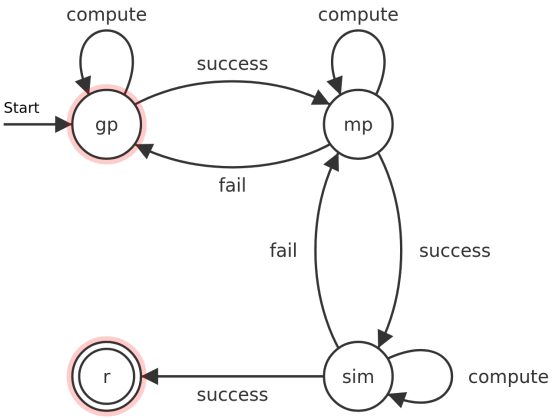


Figure 1: Finite state machine of the process for planning a movement for the robot

3 METHODOLOGY

In this section, we introduce our approach of a **CA**D based g**r**asp and m**o**tion p**l**anning system (CAROL). Figure 1 shows the general workflow as a state machine. The states are named *gp* for grasp planning, *mp* for motion planning, *sim* for simulation and *r* for robot. The exchange of data between grasp and motion planning and simulation is limited to the necessary information (i.e. grasp poses, gripper positions or desired joint positions required during pathing). As soon as the grasp planner has found good grasps, these grasps will be sent to the motion planner. The motion planner then tries to find a fitting motion and if it succeeds, the motion is validated in a simulation. If no motion could be found or if all motions found were rejected by the simulation, the grasp planner is executed to find new grasps. This is repeated until a motion is validated in the simulation and can be sent to the robot for execution.

We use GraspIt! (version 4.0.0) (Miller and Allen, 2004) for grasp planning, MoveIt! (version 1.0.7) for motion planning and Gazebo[3] (version 9.0) as simulation. GraspIt! tests thousands of possible grasp configurations, evaluates these non-positive connections according to Grasp Wrench Space quality metrics and returns the best ones. The kinematic of the robot is not considered in GraspIt!, which leads to the generation of non-feasible grasp poses. MoveIt! wraps the Open Motion Planning Library (OMPL) (Sucan et al., 2012). Of the sampling based motion planning algorithms implemented in OMPL we use

[1] https://ros-planning.github.io/moveit_tutorials/doc/bullet_collision_checker/bullet_collision_checker.html

[2] https://ros-planning.github.io/moveit_tutorials/doc/moveit_deep_grasps/moveit_deep_grasps_tutorial.html

[3] http://gazebosim.org/

RRT* (Noreen et al., 2016). Gazebo is an often used simulation for robotics applications, in particular in conjunction with the Robot Operating System (ROS). We use ROS for connecting all these components. In this ROS context, we implemented nodes, which use GraspIt! or MoveIt! and manage the reactions of the components to specific events e. g. failure in motion planning or rejection of a motion in the simulation. The huge advantage of implementing a separate node for every sub-problem is the easy interchangeability of the planners. This interchangeability is further reinforced by relying on standard message types of ROS for data exchange. Therefore, no component specific messages are needed. We found that MoveIt!'s standard hyperparameter settings were appropriate for our use case. The code, installation and usage instructions are freely available on GitHub[4].

4 CASE STUDY

CAROL is developed in the context of a research project with the aim to optimize the parameterization process of fused deposition modelling by studying environmental influences on the process (Nordsieck et al., 2019). To this end, we extract rules based on operator interactions (Nordsieck et al., 2021) and combine them with learning systems (Heider et al., 2020) which requires an accurate dataset. To build this dataset, we perform extensive series of test prints of different objects. After each print job, the printed object currently has to be removed manually from the printing plate before the next print job can start. To reduce the manual effort and increase throughput of print jobs, we want to automate this process by using a robot to remove constructed objects from the build plate. Thus, the research question for this case study can be formulated as follows:

Is it feasible to achieve robust robot-based automation in unloading varying printed parts from an FDM printer?

Figure 2 shows our real world setup with a 5 DoF manipulator Kuka YouBot placed in front of a Creality Ender 3 printer. An example showing the setup in motion executing CAROLs instructions can be found here[5]. A Raspberry Pi 2B with OctoPrint[6] (version 1.3.11) is connected to the Ender 3. OctoPrint is a web interface with which a user can control an FDM printer, upload files to print, access information from sensors and add plugins for additional functionality.

[4]https://github.com/XITASO/CAROL

[5]https://xitaso.com/wp-content/uploads/carol.mp4

[6]https://octoprint.org/

Figure 2: Real world setup with the robot (using a soft gripper) in front of the cartesian FDM printer.

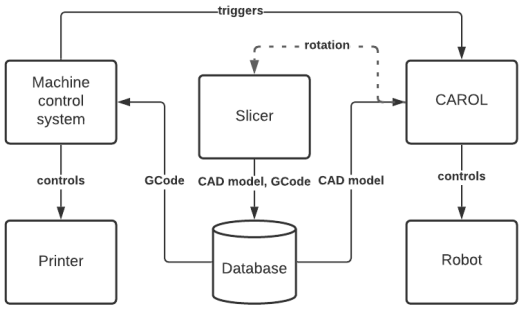


Figure 3: Our software architecture for integrating robotic automation into FDM

In Figure 3, an example architecture for automating object removal with a robot in an FDM scenario is shown. A slicer, e. g. Cura[7], turns STL files into machine readable GCode. The STL files and the GCode are uploaded to a database from which a machine control system (MCS)—in our case OctoPrint—can obtain the GCode and manage the FDM printer. The MCS also triggers CAROL, which fetches the required data from the database and starts the planning and execution of the robot movement. It is furthermore possible (but not yet implemented in our work) to let CAROL control the rotation of the printed object by specifying the desired rotation in the Slicer component.

Using the youBot poses significant challenges to

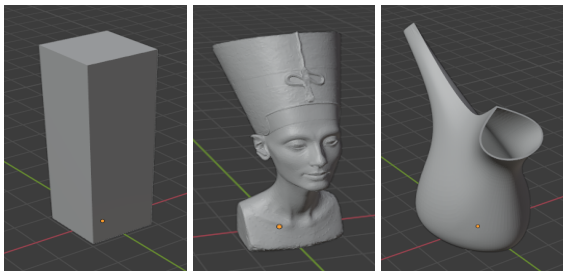
[7]https://ultimaker.com/de/software/ultimaker-cura

CAROL: aside from the kinematic restrictions caused by having only 5 DoF, the YouBot suffers from additional restrictions due to its rigid gripper, which struggles to grasp complex objects.

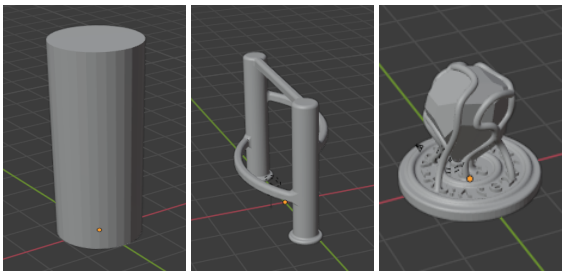
To address the grasping related restrictions, we also evaluated a contact-driven deformation based soft gripper[8], that is more likely to successfully grasp or manipulate a large variety of objects (Shintake et al., 2018). However, the soft gripper introduces new kinematic restrictions to be solved during planning due to its bigger size. Nevertheless, from our point of view these challenges will help the generality and transferability of our approach CAROL—if it can handle them, it should be easily adaptable to standard 6 DOF robots.

5 EVALUATION

To evaluate our approach, we designed experiments in which we placed the robot in different positions relative to the FDM printer, manipulated objects with different geometries and utilized two different grippers. The robot was placed to the left ($-90°$), in front ($0°$) or to the right ($+90°$) of the FDM printer. We suspected the difficulty of finding a grasp to also vary with object orientation. Thus, we utilized a two tiered experimental design. Firstly, we tested CAROL with the object's default orientation, which is determined by the way the slicer places the object on the build plate based on the information contained in the CAD model. Secondly, we rotated the object by $90°$ along the z-axis and re-applied CAROL. To conserve computation time, we only performed the second tier experiment series whenever the first tier showed sub par results. We ran each experiment 30 times to eliminate statistical influence. A run is successful (and completes) if CAROL was able to find a feasible motion into a pose that allows to grasp the desired object. This motion has to be collision-free and validated by the digital twin in the simulation. GraspIt! as well as the motion planning algorithms wrapped by MoveIt! are probabilistic. Thus, restarts of those components (unless initialised using the same seed) explore the search space differently. Runs are budgeted, i.e. after a certain amount of computation time was spent, the run is stopped even without finding a valid motion and grasp. As we expect the production process of a singular object to take at least 20 minutes (including the time to preheat the components and let the object cool down to solidify), we set that budget to 20 minutes, although feasible solutions are usually found much faster, cf. Table 2. The evaluation can easily run on typical office hardware without the need for high performance computing. In our case we utilised a Lenovo Thinkpad T470p with an Intel® Core™ i7-7700HQ@2.80 GHz and 16 GB RAM. The ROS interface[9] we used for GraspIt! does not support multi-threading. However, motion planning with MoveIt! has better multi-threading support and runs on at least two cores simultaneously.



(a) Box (b) Nefertiti[10] (c) Watering can[11]



(d) Cylinder (e) 3D print test[12] (f) Stone[13]

Figure 4: Objects of widely varying geometry tested with CAROL

For evaluation, we utilise the Estimated Success Probability (ESP) which describes the probability of finding a motion that is validated by the simulation in a single run. As we define finding a feasible solution as a successful run, the ESP is equivalent to the Feasibility Rate. ESP is determined by the ratio of successful runs to the total number of runs.

The six objects of varying geometrical complexity for which we evaluated CAROL are shown in Figure 4. In the depiction, they are rotated for the best visibility of their geometrical features.

The results of our experiments presented in Table 1 show that all objects are graspable from at least one position of the robot relative to the FDM printer. However, the rotation of the object by $90°$ (denoted

[8] http://www.youbot-store.com/developers/soft-two-finger-gripper-75

[9] https://github.com/JenniferBuehler/graspit-pkgs

[10] https://www.thingiverse.com/thing:1376105

[11] https://www.thingiverse.com/thing:770590

[12] https://www.thingiverse.com/thing:2397160

[13] copyright by: https://www.cgtrader.com/roolz

Table 1: Estimated Success Probability (ESP) [%] of a budgeted run to find a feasible solution.

	parallel gripper			soft gripper		
obj	left	front	right	left	front	right
4a	93	57	73	53	80	67
4b	3	80	0	0	50	0
4b *r*	97	3	90	30	13	40
4c	80	7	57	23	63	3
4c *r*	0	90	7	77	33	63
4d	20	17	17	47	23	23
4e	37	90	13	0	93	0
4e *r*	70	0	93	93	0	90
4f	97	67	13	27	20	13

Table 2: Expected running time (ERT) [min:sec] to find a feasible solution. ∞ is used to denote experiments for which no solution was found in the given computation budget.

	parallel gripper			soft gripper		
obj	left	front	right	left	front	right
4a	4:55	5:43	6:14	6:48	7:13	5:54
4b	7:42	5:35	∞	∞	7:21	∞
4b *r*	5:11	9:19	5:22	6:29	6:44	6:48
4c	5:48	2:14	8:04	10:47	7:15	16:51
4c *r*	∞	5:36	8:09	6:33	10:35	7:27
4d	5:53	8:50	7:46	11:45	10:20	6:00
4e	9:16	6:24	6:43	∞	4:54	∞
4e *r*	6:09	∞	8:32	3:00	∞	7:19
4f	4:03	9:19	8:33	10:26	5:43	19:28

by *r* behind the reference) achieved successful runs without the need to modify the robot's position. Thus, we conclude that when runs are not successful after a certain amount of computation budget was spent, the rotated object should be tested instead. In our experience, rotating the object did not negatively impact the production process. Note that the success rate is foremost dependent on the grasp poses provided by the grasp planner, with some poses not feasible in certain robot positions. For instance, in our experiments with objects 4b and 4e, the original success rates differ considerably from the rates of the rotated object.

Objects such as 4a and 4d, that should not necessitate a specific orientation of the robot due to their symmetry, have approximately the same success rates for at least two of the tested positions. The angular geometry of object 4a results in more stable grasps from certain poses than others, whereas for object 4d, the orientation of the grasp pose has no influence on the grasp quality. Object 4d is an especially interesting case as its seemingly simple geometry would indicate that it is easy to grasp. Nonetheless, neither computation times nor success rates reflect that assumption. A likely explanation is that, as it is round, the gripper connects to a limited contact area, which leads to unstable grasps.

We assume that the importance of the grasp pose is increased because we have a missing DoF in our manipulator that is fixed in a certain position. Therefore, we are only able to accept a limited range of orientations in our target pose. Additionally, in the case of the parallel gripper, we want to grasp objects that just fit in between the fingers. Consequently, there is not much space available to accept target poses with orientations slightly out of our range for desired poses.

The expected running time (ERT) (Price, 1997) is the computation time we expect a successful run to take in order to find a validated motion. The ERT is calculated as follows, with n as number of successful planning attempts and t_m as the time to conclude a singular run $m \in \{0,...,n\}$:

$$\mathrm{ERT}_{\mathrm{object}} = \frac{\sum_{m=1}^{n} t_m}{n}$$

The ERT—in combination with the ESP—indicates whether the time to plan a grasp and motion is sufficient if we consider continuous production of objects. Note that unsuccessful runs always consume the allotted budget and that we assume a fixed and limited hardware.

The results of our experiments in regards to ERT are shown in Table 2. An ERT with the value ∞ is placed whenever we were not able to find a valid motion for an experiment. We observe an ERT of ∞ in experiments 6b, 6c r, 6e and 6e r. However, if the object was rotated, we were able to find a valid motion for all cases. Consequently, it can plausibly be assumed that a specific point exists at which it would be more efficient to restart the planning process with a rotated object. Finding this point would require extensive statistical analysis which we have not performed yet. However, employing this principle, CAROL can automatically find a valid motion for 100% of tested objects. From our experience, we can furthermore assume that users will require only moderate amounts of experience to estimate productive rotations from the object geometry alone.

Interestingly, ESP and ERT do not seem to inversely correlate. Hence, it seems likely that some random seeds used by our planning processes are inherently better suited for exploration than others. The presented times may seem very high in comparison to state-of-the-art evaluations shown by (Meijer et al., 2017; Yang et al., 2018). However, it has to be noted that their evaluations refer only to motion planning by itself. A target pose was given beforehand and did therefore not have to be computed. Of the presented times, our grasp planner takes about two thirds

(66%) of the total computation time, thus, exchanging GraspIt! with a faster grasp planner would reduce the total computation time considerably. This evaluation should show that our approach with separated grasp and motion planning is able to find a simulation-approved collision free trajectory to grasp every tested object even for a 5 DOF manipulator. In contrast to other approaches for grasp and motion planning (Fontanals et al., 2014; Vahrenkamp et al., 2010), our approach allows to easily exchange all components such as grasp planner, motion planner or the robot.

Coming back to the research question established in Section 4, we can now answer it as follows: Yes, robust robot-based automation in unloading FDM printers is feasible based on our approach and state-of-the-art automatic planning algorithms—at least if the rotation of the printed objects is actively controlled or the robot has a sufficiently large workspace. In the case of the YouBot, the workspace could be extended by placing it on its mobile robot base, which poses new challenges of course, e.g. regarding positioning accuracy.

6 FUTURE WORK

For our use case, the overall results are sufficient as computation times were lower than individual production times, which allows us to automate object removal. However, we intend to further evaluate the degree of certainty of the simulation by conducting more real world experiments.

Furthermore, we would like to evaluate the applicability of our approach to other domains such as CNC milling. If the approach is to be transferred to domains with stricter requirements for computational time, ERT should be improved to avoid increased hardware requirements. The biggest room for improvement lies within the generation of feasible grasps. Therefore, evaluating other grasp planners or at least multi-threading GraspIt! should be considered. To the best of our knowledge, there are no approaches that use a CAD-based ANN for grasp planning. Consequently, we want to develop one and evaluate its performance by comparing it with point cloud based ANN approaches such as (Staub et al., 2019). While we expect that improvements in regards to ESP should be marginal, the computation time is likely to be reduced. Another possible improvement could be provided by a grasp planner that is able to take the object's and gripper's geometry into consideration to produce grasps that form positive connections.

We also expect improvements by employing a grasp planner capable of considering previous successful grasps when generating new ones to increase the probability of generating a feasible grasp, in turn also decreasing computation times. As our current data set of successful grasps is quite small, training machine learning models for grasp estimation based on object geometry is not yet feasible. However, continued use of our setup or validation with GraspIt! and a simulation will help us create a suitable dataset. A possible candidate architecture for a machine learning model is Dex-Net (Mahler et al., 2018), which already contains a multitude of objects and feasible grasps for parallel jaw grippers.

Another improvement with regards to grasp planning would be better handling of the soft gripper with more contact points and a physics enabled simulation.

To the best of our knowledge, robots with 6 or 7 DoF are more common than those with 5 DoF. Consequently, the effect of the missing DoF should be evaluated. We expect it to negatively affect ESP and ERT since, with an additional DoF, more grasp poses determined by the grasp planning can be reached, hence resulting in a higher probability and lower computation times to find successful grasp-motion combinations. Given that CAROL is indeed able to work with only 5 DoF this can also help further validate our approach. Additionally, we want to extend CAROL by the capability to automatically rotate the object and reiterate grasp and motion planning if a valid motion could not be found (cf. Figure 3).

7 CONCLUSION

We presented CAROL, an entirely model-based grasp and motion planning approach to unload a print bed in FDM-based manufacturing. CAROL successfully integrates motion planning, grasp planning and a digital twin in simulation for verification before online execution. We designed our approach with a focus on low coupling. Every component is independent of the others and can therefore be exchanged comfortably without requiring code changes in other components. In our evaluation, we have shown that we are able to find both grasp and motion that were validated by the simulation for all objects. In preliminary real world tests, we found that, with a soft gripper, all pick manoeuvres were successful, although we plan on expanding these tests in the future as we further integrate CAROL into our additive manufacturing scenario. However, we note that the computation time differs between objects. One reason for the arguably large computation time is that we aim to use a 5 DoF robot with its kinematic limitations, which are not easily considered in grasp planning. This leads to the

generation of grasps that are not feasible for the robot restarting planning until feasible grasps are found.

The grasp planning component GraspIt! takes about two thirds (66%) of the total computation time. Thus, we identify the future improvement of this component as the most important. However, even now, grasps are usually found acceptably fast for the application in an additive manufacturing setting where individual production times are high. On standard office hardware, we can still operate multiple robot printer setups at once.

CAROL has three main advantages over vision guided approaches: (1) higher grasp precision, which is obtained by utilizing information about the object's geometry; (2) no sensory requirements (e. g. camera); (3) easy adaptability to other domains.

Thus, CAROL provides highly precise grasps as well as verified collision avoidance for geometrically complex objects while requiring neither expensive computing nor complex sensory hardware. In addition, the simple interchangeability of its major components allows for independent further optimization and adaptation as required in different settings and target domains.

ACKNOWLEDGEMENTS

This work is supported by the German Federal Ministry for Economic Affairs and Energy (BMWi).

REFERENCES

Aroca, Ventura, De Mello, and Pazelli (2017). Sequential additive manufacturing: Automatic manipulation of 3d printed parts. *Rapid Prototyping Journal*, 23(4):653–659.

Becker, P., Eichmann, C., Roennau, A., and Dillmann, R. (2020). Automation of post-processing in additive manufacturing with industrial robots. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1578–1583.

Bohg, J. and Kragic, D. (2010). Learning grasping points with shape context. *Robotics and Autonomous Systems*, 58(4):362–377.

Chen, Y., Sun, G., Lin, H., and Chen, S. (2018). Random bin picking with multi-view image acquisition and cad-based pose estimation. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2218–2223.

Depierre, A., Dellandréa, E., and Chen, L. (2018). Jacquard: A Large Scale Dataset for Robotic Grasp Detection. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3511–3516.

Dorn, S., Wolpert, N., and Schömer, E. (2020). Voxel-based General Voronoi Diagram for Complex Data with Application on Motion Planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 137–143.

Fontanals, J., Dang-Vu, B., Porges, O., Rosell, J., and Roa, M. A. (2014). Integrated grasp and motion planning using independent contact regions. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 887–893.

Garg, V., Sharma, T., Kumar, A., and Rastogi, V. (2020). Handaid: A seven DoF Semi-Autonomous Robotic Manipulator. In *2020 5th International Conference on Control and Robotics Engineering (ICCRE)*, pages 37–41.

Gatrell, L. B. (1989). CAD-based grasp synthesis utilizing polygons, edges and vertexes. In *Proceedings, 1989 International Conference on Robotics and Automation*, pages 184–189 vol.1.

Ghalamzan E., A. M., Mavrakis, N., Kopicki, M., Stolkin, R., and Leonardis, A. (2016). Task-relevant grasp selection: A joint solution to planning grasps and manipulative motion trajectories. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 907–914.

Goldfeder, C., Allen, P. K., Lackner, C., and Pelossof, R. (2007). Grasp Planning via Decomposition Trees. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4679–4684.

Gonçalves, J. and Lima, P. (2019). Grasp Planning with Incomplete Knowledge About the Object to be Grasped. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 1–6.

Görner, M., Haschke, R., Ritter, H., and Zhang, J. (2019). MoveIt! Task Constructor for Task-Level Motion Planning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 190–196.

Heider, M., Pätzel, D., and Hähner, J. (2020). Towards a Pittsburgh-Style LCS for Learning Manufacturing Machinery Parametrizations. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, GECCO '20, pages 127–128, New York, NY, USA. Association for Computing Machinery.

Hou, Y., Jia, Z., and Mason, M. T. (2018). Fast Planning for 3D Any-Pose-Reorienting Using Pivoting. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1631–1638.

Huang, J., Zhang, F., Dong, X., Yang, R., Xie, J., and Shang, W. (2020). Vision-guided Dynamic Object Grasping of Robotic Manipulators. In *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 460–465.

Huh, J., Lee, B., and Lee, D. D. (2018). Constrained sampling-based planning for grasping and manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 223–230.

Kumra, S. and Kanan, C. (2017). Robotic grasp detection using deep convolutional neural networks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 769–776.

Lee, S. and Lee, Y. (2020). Real-Time Industrial Bin-Picking with a Hybrid Deep Learning-Engineering Approach. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 584–588.

Mahler, J., Matl, M., Liu, X., Li, A., Gealy, D., and Goldberg, K. (2018). Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5620–5627.

Meijer, J., Lei, Q., and Wisse, M. (2017). An empirical study of single-query motion planning for grasp execution. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1234–1241.

Miller, A. T. and Allen, P. K. (2004). Graspit! a versatile simulator for robotic grasping. *IEEE Robotics Automation Magazine*, 11(4):110–122.

Miller, A. T., Knoop, S., Christensen, H. I., and Allen, P. K. (2003). Automatic grasp planning using shape primitives. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 2, pages 1824–1829 vol.2.

Nordsieck, R., Heider, M., Angerer, A., and Hähner, J. (2019). Towards Automated Parameter Optimisation of Machinery by Persisting Expert Knowledge. pages 406–413.

Nordsieck, R., Heider, M., Winschel, A., and Hähner, J. (2021). Knowledge Extraction via Decentralized Knowledge Graph Aggregation. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, pages 92–99.

Noreen, I., Khan, A., Habib, Z., et al. (2016). Optimal path planning using RRT* based approaches: a survey and future directions. *Int. J. Adv. Comput. Sci. Appl*, 7(11):97–107.

Price, K. V. (1997). Differential evolution vs. the functions of the 2/sup nd/ ICEO. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, pages 153–157.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. (2009). ROS: an open-source Robot Operating System. volume 3.

Rosa, S., Russo, A., Saglinbeni, A., and Toscana, G. (2016). Vocal interaction with a 7-DOF robotic arm for object detection, learning and grasping. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 505–506.

Shintake, J., Cacucciolo, V., Floreano, D., and Shea, H. (2018). Soft robotic grippers. *Advanced Materials*, 30(29):1707035.

Staub, B., Tanwani, A. K., Mahler, J., Breyer, M., Laskey, M., Takaoka, Y., Bajracharya, M., Siegwart, R., and Goldberg, K. (2019). Dex-Net MM: Deep Grasping for Surface Decluttering with a Low-Precision Mobile Manipulator. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1373–1379.

Sucan, I. A., Moll, M., and Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics Automation Magazine*, 19(4):72–82.

Tanwani, A. K., Mor, N., Kubiatowicz, J., Gonzalez, J. E., and Goldberg, K. (2019). A Fog Robotics Approach to Deep Robot Learning: Application to Object Recognition and Grasp Planning in Surface Decluttering. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4559–4566.

Vahrenkamp, N., Do, M., Asfour, T., and Dillmann, R. (2010). Integrated grasp and motion planning. In *2010 IEEE International Conference on Robotics and Automation*, pages 2883–2888.

Yang, Y., Merkt, W., Ivan, V., and Vijayakumar, S. (2018). Planning in time-configuration space for efficient pick-and-place in non-static environments with temporal constraints. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9.

Zelch, C., Peters, J., and von Stryk, O. (2020). Learning control policies from optimal trajectories. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2529–2535.

Zhang, H., Ichnowski, J., Avigal, Y., Gonzales, J., Stoica, I., and Goldberg, K. (2020). Dex-Net AR: Distributed Deep Grasp Planning Using a Commodity Cellphone and Augmented Reality App. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 552–558.

Zhao, S., Xu, Y., and Tan, Y. (2019). Detecting grasping positions and postures in 3D point clouds by geometric constraints. In *2019 IEEE International Conference on Unmanned Systems (ICUS)*, pages 443–449.

Zunjani, F. H., Sen, S., Shekhar, H., Powale, A., Godnaik, D., and Nandi, G. C. (2018). Intent-based Object Grasping by a Robot using Deep Learning. In *2018 IEEE 8th International Advance Computing Conference (IACC)*, pages 246–251.