Automated Depth Dataset Generation with Integrated Quality Metrics for Robotic Manipulation

Albert Dalmases
Inst. of Industrial and Control Eng.
Universitat Politècnica de Catalunya
Barcelona, Spain
albert.dalmases@estudiantat.upc.edu

Oriol Ruiz-Celada
Inst. of Industrial and Control Eng.
Universitat Politècnica de Catalunya
Barcelona, Spain
oriol.ruiz.celada@upc.edu

Jan Rosell
Inst. of Industrial and Control Eng.
Universitat Politècnica de Catalunya
Barcelona, Spain
jan.rosell@upc.edu

Isiah Zaplana
Inst. of Industrial and Control Eng.
Universitat Politècnica de Catalunya
Barcelona, Spain
isiah.zaplana@upc.edu

***Abstract*—This work introduces a fully automatic and adaptable pipeline for synthetic depth dataset generation that later on can be used for the training of deep learning algorithms for robotic manipulation. From any available set of 3D object model meshes, the pipeline outputs rendered depth image data with labeled grasp candidates represented as a set of grasping points relative to the camera with associated metrics. The proposed pipeline allows adaptability in various characteristics such as the input dataset of objects, the sampling method, the gripper type, or the grasp evaluation metrics to allow the generation of a more customized, task-oriented collection of labeled grasps relevant for different robotic applications. The implementation is done using Blender's Python API workspace, avoiding the use of multiple software tools or libraries, reducing the pipeline complexity, facilitating the extensibility, and providing benefits in terms of visualization and debugging.**

I. Introduction

Robotic manipulation research relies significantly on the ability to identify feasible grasping points or grasping poses from objects in images. Recent studies [1]–[4] suggest deep learning and the use of depth data for training as the preferred solution for predicting successful grasps on previously unseen objects. This is due to the ability of deep learning models to identify complex patterns within known data to generalize it to unknown data.

However, the performance of these methods is heavily dependent on the quality and quantity of training data, which might not always be readily available or require time-consuming preparation [5]. Using real-world data is more representative but it is costly and laborious to collect. In contrast, synthetic data can be easily generated but may lack real-world applicability. To bridge this gap, domain randomization [6] or domain adaptation [7] are often used.

In this line, this paper contributes with the proposal of a versatile, automated pipeline for generating synthetic data that later on can be used for training learning models for manipulation-oriented grasping, trying to make the pipeline as adaptable as possible and with ease to be expanded so it can handle a variety of robotic grippers, objects and manipulation scenarios. This has great relevance to industries where robots have a diverse range of applications and require different grasping strategies for a set of objects and gripper designs. Lastly, the fact that everything is integrated into a single workspace simplifies potential modifications, updates or enhancements.

This work was supported by the European Commission's Horizon Europe Framework Programme with the project SmartHandle (Resilient manufacturing lines based on smart handling systems) under Grant Agreement ID 101091792.

II. Related Work And Approach Overview

Some approaches [8] focus on the dataset generation relying on GraspNet [9] for defining grasps and their associated quality metrics. This could potentially decrease the diversity of grasps, as it utilizes a 3-DOF pose representation for parallel jaw-grippers only. Other approaches employ GraspIt! [10], which also has limitations in terms of grasp pose diversity due to its naive search strategy, and its grasping evaluation is limited to pick-and-place motions. Alternatively, Andries et al. [11] present a pipeline for generating the object set from scratch, but the provided analytical quality metrics turned out to be unreliable.

In an effort to address the aforementioned constraints, a more versatile and adaptable pipeline is proposed. This pipeline is capable of accommodating changes in the object sampling method, gripper type, characteristics, and grasping metrics. The latter can be used to evaluate and rank grasping candidates. Representing grasping candidates as a gripper pose might be adequate for suction cups and parallel-jaw grippers, but less effective for more complex, multi-fingered grippers. To better accommodate these, the use of contact point as grasping candidates is proposed, which should generalize and scale better for a variety of objects and grasping scenarios. As part of future work, optimization strategies will be used not only to determine the optimal gripper pose in relation to the selected contact points, but also to ensure the feasibility of this pose with respect to the environment and the gripper's own configuration. This is particularly important given that we do not consider kinematic constraints and potential hardware collisions during the dataset generation. The reason for this approach is to prevent the need to manage high-dimensional configuration spaces typical of these tasks, which would

result in an exponential increase in data, computational costs, simulation complexity and make the learning process more difficult.

III. The Pipeline

The proposed pipeline offers an adaptable framework capable of processing each object mesh individually, treating them as free-flying objects and generating grasping candidates across all feasible surfaces. The pipeline supports grippers such as vacuum, parallel-jaw, and three-fingered grippers, using known basic geometries to generate the feasible candidates for the dataset. The main stages of the pipeline include: a sampling stage where the selected mesh is sampled using diverse methods, a selection of potential grasping candidates stage where the sets of points on the object's surface are filtered to those that might successfully interact with a specific type of gripper, an evaluation stage where grasping metrics are set to the candidates, and the generation of depth images using domain randomization to provide a more diverse set of training data. Note that the idea is to process each mesh once and then generate depth images with different perspectives to label the visible grasping candidates to each perspective. This method ensures a diverse set of potential grasps, covering all possible orientations and angles, thereby extending beyond these typical scenario of an object placed on a surface. Finally, all the relevant data from the previous steps are compiled into a dataset format.

The output labeled data can be represented as a set of samples. Each sample S_i can be expressed as $S_i = (Y_i, \mathbf{g_i}, \mathbf{m_i})$, where $Y_i \in \mathbb{R}^{H \times W}$ is a matrix representing the point cloud depth image with height H and width W. The point cloud is constituted by three-dimensional points expressed with respect to the camera. In addition, $\mathbf{g_i}$ is the set of all grasping point candidates on the image i. Each grasp j of the set $\mathbf{g_i}$ can be expressed as $g_{ij} = (\mathbf{p_1})$, $g_{ij} = (\mathbf{p_1}, \mathbf{p_2})$ or $g_{ij} = (\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3})$, for a vacuum, a parallel-jaw and a three fingered grasp, respectively, each $\mathbf{p_k} = (x, y, z) \in \mathbb{R}^3$ being the point location relative to the camera and $\mathbf{m_i} \in \mathbb{R}^{j \times n}$ an array of n quality metrics associated to each grasping candidate.

Notably, the proposed pipeline leverages existing technologies, such as BlenderProc[1], useful for the rendering and physics simulation, point-cloud-utils[2] and igl[3] libraries that provide functionalities for mesh sampling and other geometric computations, and blender-plot[4] library to simplify the visualization process within blender by providing a matplotlib-like API.

A. Parameter Setup

The parameter setup of the pipeline provides users with substantial customization capacity, allowing them to tailor the pipeline and resulting dataset to their unique requirements.

[1]https://github.com/DLR-RM/BlenderProc
[2]https://github.com/fwilliams/point-cloud-utils
[3]https://github.com/libigl/libigl-python-bindings
[4]https://github.com/Linusnie/blender-plots

Users can adjust the sampling parameters, which include the choice of sampling method—random, poisson disk, or curvature-based sampling—and the desired number of samples per mesh. The selection of the sampling method has a significant impact on the distribution and richness of the generated grasp candidates, with different methods offering varying trade-offs between computation time and sampling quality. In addition, the parameter setup also enables users to control the size and diversity of the dataset through the number of depth output scenes to be generated per object. Each scene provides a different perspective of the object and potential grasping points, increasing the variety of data. Furthermore, the gripper parameters allows to choose vacuum, parallel-jaw, or three-fingered grippers, each of which carries specific additional parameters.

For parallel-jaw grippers, parameters such as the jaw width and depth can be set, as well as the conditions for generating the grasping candidates, such as including squeezing and/or expanding grasps, and set specific geometric criteria through angle thresholds between the candidate points and gripper. Such parameters allow the inclusion of a broader range of scenarios and directly impact the diversity and complexity of the generated grasps. Similarly, vacuum or three-fingered grippers include their own tunable parameters.

Lastly, the metrics configuration section allows users to select from the available set of metrics to evaluate the grasping candidates. Metrics can substantially influence the criteria for selecting grasps and the usefulness for a specific grasping task.

B. Mesh Sampling

The mesh sampling process forms the initial step in the creation of grasping candidates. It involves sampling points on the mesh surface of the object as a foundation for the stages that follow.

At the start, the mesh data is loaded including its vertices and faces. Additional parameters such as the geometric center of mass, calculated as the average of all vertices, and the surface normals of the mesh are also determined during this stage. The algorithm presents several sampling methodologies. Random sampling samples mesh surface points randomly. The Poisson Disk sampling method, on the other hand, ensures a more uniform distribution of points. It provides a better coverage of the object's surface, resulting in a distribution commonly known as "blue noise". Curvature-based sampling, selects points based on the curvature of the object. This method may yield different results depending on the gripper type. For instance, vacuum grippers may benefit from an intensified sampling in areas of low curvature, whereas grippers with multiple fingers may find areas of high curvature more beneficial.

C. Grasping Candidates Selection

In the selection stage, grasping candidates are filtered based on conditions specific to each gripper type. This process highlights the flexibility of the candidate selection process for various robotic manipulation scenarios. Notice that, when different filter steps are applied, the order in which they are

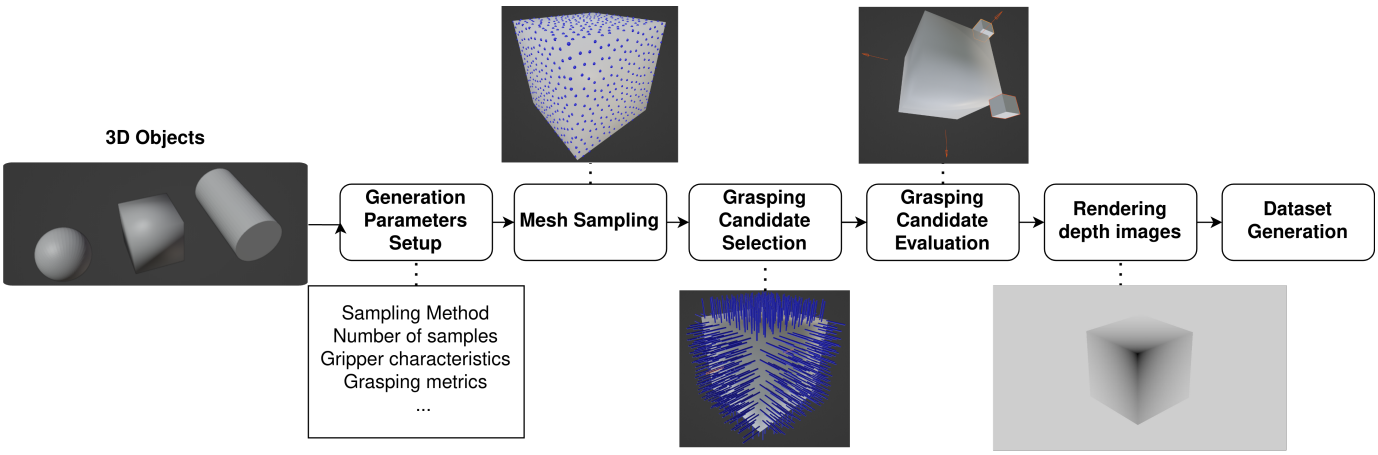

Fig. 1. Schematic representation of the stages - setup, sampling, selection, evaluation, rendering and generation - of the dataset generation pipeline.

applied is based on their computational speed, i.e., faster steps are prioritized so that it minimizes the number of candidates for subsequent steps that might require more computational effort. However, when implementing these filter steps, an early stop is included so that unfeasible candidates are discarded as soon as possible.

For vacuum grippers, grasping candidates are assessed for their compatibility with the suction cone, which is used to secure the object. The primary condition for a valid grasp is the availability of an adequate number of mesh points within the perimeter of the suction cone. To quantify this, the intersection points between the vacuum cone and the object mesh in the direction of the candidate surface normal is first computed. This essentially gives a subset of mesh points that should be in direct contact with the vacuum cone's perimeter. Then, an average plane is fit to these points ensuring that the distance from these and the average plane is below a user-defined threshold. This condition ensure that the grasp can be tight and secure. Furthermore, the angle between the average plane normal and the grasping candidate normal should also be below a threshold. Otherwise it might be an indicator of a potential irregular surface that could lead to an unstable grasp. Both aforementioned thresholds are defined according to the vacuum cone flexibility.

For parallel grippers several conditions have to be verified for a grasp to be valid. The first step focuses on geometric compatibility. This consists on verifying that the angles between the grasping points axes and their normals fall within a user-defined threshold. Additionally, the directionality of the normals is also checked to determine if grasps are squeezing or expanding. A squeezing grasp involves the robotic gripper applying inward pressure to securely hold an object, while an expanding grasp occurs when the gripper applies outward pressure, expanding to secure the object. Then the gripper physical width and depth constraints are evaluated. Firstly, the Euclidean distance between the pair of contact points is measured and checked against the gripper jaw width to ensure the grasp is physically achievable. Secondly, the depth is assessed by first defining the axis along the line connecting the two grasp points and generating a series of points along a circle in the plane perpendicular to this axis at a distance equivalent to the gripper's jaw depth. Then, by performing a ray-mesh intersection operation from the generated points in a direction tangent to the grasping axis, we can determine if the gripper would collide with the object's mesh. Furthermore it is also determined if it exists at least one approach direction at the maximum gripper width that would not collide with the object mesh. The final step verifies the force closure condition as specified in [12], considering a friction coefficient of 0.5.

For three-fingered grippers, conditions to ensure successful grasping are to be developed.

D. Grasping Candidates Evaluation

The evaluation of candidates is a key part of the whole pipeline since it has the greatest room for enhancements. The goal of this stage is to numerically evaluate each possible grasp considering specific metrics of interest. The current work is focused on the evaluation of parallel-jaw grippers, for which a *simulated success ratio* metric within Blender physics environment is implemented as follows. An initial arbitrary orientation of the object is considered. Then, for each candidate $(\mathbf{p_1}, \mathbf{p_2})$, small *grasping cubes* that simulate the gripper fingertips are created at $\mathbf{p_1}$ and $\mathbf{p_2}$ and used to simulate the force applied by the gripper by moving the cubes along the axis $\overline{\mathbf{p_1 p_2}}$. If after some time the object has not significantly moved, the grasp is successful, i.e. it is able to compensate for the gravity. The process is repeated for different gravity directions, and the final success ratio is computed as the proportion of successful grasps.

E. Rendering and Dataset Generation

This stage generates depth images of the objects and processes all the relevant data from the previous steps. The first part involves domain randomization, which randomizes the camera position inside a sphere around the object and facing towards the object, to generate depth images. In the previous

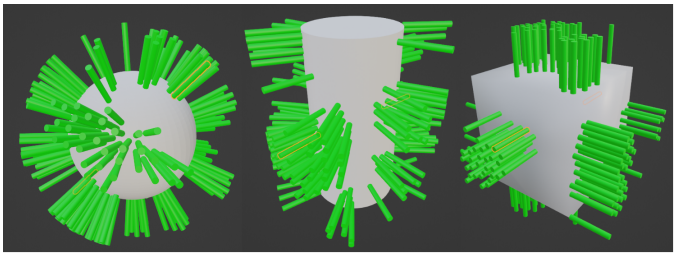

Fig. 2. Preliminary results of the synthetic depth dataset generation pipeline using three basic mesh shapes - sphere, cylinder, cube - to represent varying complexities and potential grasping scenarios for parallel-jaw grippers.

stages, a full set of potential grasps are identified. However, not all of these potential points will be visible in every image or scene due to changes in perspective and therefore, only the grasp positions that are actually visible in that scene are included. Furthermore, candidates are transformed from global to camera coordinates, retaining the grasping metrics associated with them from the evaluation stage.

Finally, in terms of data export, the depth images and their associated data are packaged into a usable format for deep learning training applications. Each depth image is saved in a standard image format like PNG or EXR, ensuring compatibility with common image processing tools and libraries. Finally a single JSON file is created containing all grasp candidates associated to the generated images and their metrics.

IV. Results

This section includes initial tests of the pipeline for parallel jaw grippers with the simulated success ratio. The sampling type used is set to Poisson with a target of approximately 1000 samples per object, and the number of depth output scenes per object is set to 10. Regarding the gripper, a parallel gripper is used with maximum jaw width set to 0.07 meters and maximum jaw depth set to 0.05 meters, allowing only squeezing grasps in the dataset. The maximum angle between the normal vectors of the candidates, as well as the maximum angle between each normal and the axis joining the points, is set to 15 degrees.

For the test dataset, three basic mesh shapes are included: a cylinder, a cube, and a sphere. These three shapes represent a range of different complexities and offer a wide array of grasping possibilities for parallel jaw grippers. As the preliminary results are based on these configurations, it is essential to note that they may differ when applied to different shapes, gripper types, or varying parameters within the pipeline.

At this stage, satisfactory initial results are observed, with the pipeline successfully identifying viable grasp points, associating metrics and effectively generating diverse depth images.

V. Conclusions

This work introduces an automated pipeline for generating datasets that can be used for the training of learning models for robotic manipulation capable of handling different types of grippers. Initial results showed promising results in the key stages of grasp candidate selection, evaluation, and dataset generation. Future work will focus on enhancing the pipeline with additional features and optimizing it for efficiency, with the ultimate goal of facilitating the development and training of deep learning models for manipulation-oriented grasping, potentially through the use of hardware acceleration or parallel processing techniques.

In terms of the pipeline specific stages, new selection criteria and evaluation metrics tailored to the different types of grippers will be introduced. Such metrics should be innovative and effective to consider aspects such as grasping adaptability or grasp orientation.

Future research will also focus on implementing the suitable neural network architectures compatible with the generated datasets, where the utilization of grasping quality measures might be explored in three different possible contexts - used to preprocess the dataset before training, learned during the training, or incorporated on the training algorithm through a multi-loss function approach.

References

[1] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-DOF grasp generation in cluttered scenes," *CoRR*, vol. abs/2103.14127, 2021.

[2] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.

[3] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems (RSS)*, 2017.

[4] A. I. Károly and P. Galambos, "Task-specific grasp planning for robotic assembly by fine-tuning gqcnns on automatically generated synthetic data," *Applied Sciences*, vol. 13, no. 1, 2023.

[5] J. Ruiz-del-Solar, P. Loncomilla, and N. Soto, "A survey on deep learning methods for robot vision," *CoRR*, vol. abs/1803.10862, 2018. [Online]. Available: http://arxiv.org/abs/1803.10862

[6] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, W. Zaremba, and P. Abbeel, "Domain randomization and generative models for robotic grasping," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3482–3489.

[7] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4243–4250.

[8] D. Morrison, P. Corke, and J. Leitner, "EGAD! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4368–4375, 2020.

[9] A. Mousavian, C. Eppner, and D. Fox, "6-DOF graspnet: Variational grasp generation for object manipulation," *CoRR*, vol. abs/1905.10520, 2019. [Online]. Available: http://arxiv.org/abs/1905.10520

[10] A. Miller and P. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics Automation Magazine, IEEE*, vol. 11, no. 4, pp. 110 – 122, dec. 2004.

[11] M. Andries, Y. Fleytoux, S. Ivaldi, and J.-B. Mouret, "AGOD-Grasp: an Automatically Generated Object Dataset for benchmarking and training robotic grasping algorithms," Mar. 2023, working paper or preprint. [Online]. Available: https://inria.hal.science/hal-03983079

[12] I.-M. Chen and J. Burdick, "Finding antipodal point grasps on irregularly shaped objects," *Robotics and Automation, IEEE Transactions on*, vol. 9, pp. 507 – 512, 09 1993.