Automatic Dataset Generation From CAD for Vision-Based Grasping

Saad Ahmad[1,†], Kulunu Samarawickrama[1,†], Esa Rahtu[2] and Roel Pieters[1]

Abstract*— Recent developments in robotics and deep learning enable the training of models for a wide variety of tasks, from large amounts of collected data. Visual and robotic tasks, such as pose estimation or grasping, are trained from image data (RGB-D) or point clouds that need to be representative for the actual objects, to acquire accurate and robust results. This implies either generalized object models or large datasets that include all object and environment variability, for training. However, data collection is often a bottleneck in the fast development of learning-based models. In fact, data collection might be impossible or even undesirable, as physical objects are unavailable or the physical recording of data is too time-consuming and expensive. For example, when building a data recording setup with cameras and robotic hardware. CAD tools, in combination with robot simulation, offer a solution for the generation of training data that can be easily automated and that can be just as realistic as real world data. In this work, we propose a data generation pipeline that takes as input a CAD model of an object and automatically generates the required training data for object pose estimation and object grasp detection. The object data generated are: RGB and depth image, object binary mask, class label and ground truth pose in camera- and world frame. We demonstrate the dataset generation of several sets of industrial object assemblies and evaluate the trained models on state of the art pose estimation and grasp detection approaches. Code and video are available at: *https://github.com/KulunuOS/gazebo_dataset_generation

I. Introduction

Grasping and manipulation of objects by a robot manipulator is a common task, often executed without external sensors, such as cameras. Industrial processes, for example, can be designed in such way that recurring objects are always at the same location, where fixtures provide the support for grasping and following handling steps [1]. Variability, however, is a rising trend in agile production with reconfiguration and flexibility of the work cell a requirement. Computer vision can offer solutions for detecting objects and suitable grasp locations, on the condition that objects are known or similar to previously seen objects [2]. Deep learning approaches [3] utilize vasts amount of data to train a detection model, assuming that such data is representative for the actual task. In case data is unavailable, incorrect or difficult to obtain, a different solution has to be found. In addition, from an industrial perspective, objects might only be available physically and generating the required training data requires skilled and time-consuming preparation.

The automatic generation of training data, in the form of RGB-D images, object binary masks and ground truth poses is therefore of value, especially through Computer-Aided Design (CAD) and simulation (see Fig. 1). This paper describes these exact developments, for the robotic task of grasping. In particular, we propose a dataset generation pipeline that utilizes CAD models and standard open source software modules to generate the required training data.

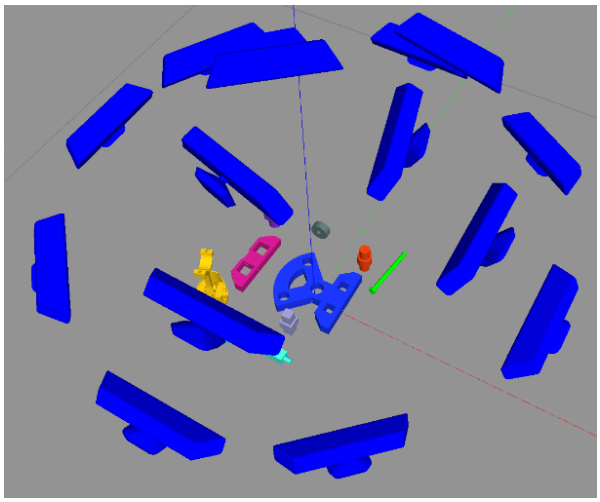


Fig. 1: Simulated objects (colored) and cameras (blue) to automatically generate an object dataset (RGB-D, object binary masks and ground truth poses) for vision-based grasping.

We demonstrate the data generation pipeline by three industrial assemblies. Each assembly has several parts and is designed in CAD or selected from open source CAD repositories (see Table I). The pipeline simulates the assemblies and a camera to generate RGB-D images, object binary masks, labels and ground truth poses. Evaluation of the datasets on a pose estimation model and grasp detection model verifies that the training data is suitable for vision-based grasping. Summarizing, the contributions of this work are:

- Automatic object dataset generation pipeline from CAD
- Simulation of objects and RGB-D camera in Gazebo
- Variation in parameters (e.g., yaw, pitch, scale, resolution) captures dataset variability
- Object binary masks and labels obtained via depth information

The paper is organized as follows. We introduce the paper in Section I. A brief overview of the current developments in grasp detection and existing datasets is given in Section II, which describes the state of the art and their limitations. The proposed automatic dataset generation pipeline is described in Section III and its results are explained in Section IV. Section V concludes the work.

[1]Unit of Automation Technology and Mechanical Engineering, [2]Unit of Computing Sciences, Tampere University, 33720, Tampere, Finland; [†]both authors contributed equally, firstname.surname@tuni.fi

TABLE I: Overview of CAD Model datasets. ShapeNetCore and ShapeNetSem are a subset of ShapeNet. Most datasets listed cover a wide variety of categories. The abbreviations in the **License** column denote: C; the copyright of the CAD models is owned by their creators, P; Proprietary, NC; Non-commercial use, OS; open-source.

Dataset (**year**)	**Parts**	**Categories** (**number**)	**Parametric**	**Data source**	**License**
ABC [4] (2019)	1,000,000+	-	✓	Onshape [5]	C
Fusion360 Gallery [6] (2020)	8,625	-	✓	Autodesk Online Gallery [7]	P
MCB [8] (2020)	58,696	Mechanical components (68)	-	TraceParts [9] 3D Warehouse [10] GrabCAD [11]	-
ShapeNet [12] (2015) ShapeNetCore ShapeNetSem	3,000,000+ 51,300 12,000	(3,135) (55) (270)	-	3D Warehouse [10]	NC
Thingi10K [13] (2016)	10,000	3D printing models	-	Thingiverse [14]	OS

II. Related Work

A. Grasp detection

Object grasp detection is a popular topic in robotics and can be divided in several categories to differentiate between approaches and their assumptions. For example, the representation of a grasp is an important consideration and determines the complexity of the problem and its application [15]. When considering only a planar grasp pose representation, grasp detection is simplified to finding the object and its orientation on a planar surface, typically represented as an (oriented) bounding box. On the other hand, in case a complete 3D pose of the object is required for grasping, detection should return the full 3D position and 3D orientation of the object. In context of learning-based grasp detection [3], typical data-driven approaches [16] differentiate between the utilization of RGB, depth (in form of point clouds) [17], [18] or a combination of both (RGB-D) [19]. In addition, objects to be grasped can be known, similar (i.e., different instance of a known category) or novel, which should be considered when deciding (or developing) on the data representation, collection and training approach. A comprehensive survey of approaches is presented in [2]. In particular, PVN3D [19] is a recent, deep point-wise 3D keypoints-based voting network for 6-DOF pose estimation and utilizes single RGB-D images for inference. 6-DOF GraspNet [18], on the other hand, generates grasps directly as output of the network, from 3D point clouds observed by a depth camera.

B. Existing datasets

Existing datasets for 2D object detection, such as COCO [20] and Objectron [21] for 3D objects, are widely available, including common objects that are present in everyday scenes. Similarly, most of the recent and state of the art approaches in grasp detection utilize datasets with large variation in objects, thereby aiming to capture as much object geometries as possible. This inevitable leads to large datasets (gigabytes), long training times and heavy detection models [22], [23]. Alternative approaches are considered as well, with object models that cover a statistically large variety in geometry [24].

One additional differentiation between datasets and their generation is whether the collected data is from real [25], [26], [17] or simulated [27], [28], [29] objects. While real objects offer more realistic data to be captured, adding more object models and expanding the dataset is difficult when considering the utilized hardware for data similarity. In addition, most of the datasets only include common household objects, as these are freely available. Industrial parts, on the other hand, are not easy to obtain and too specific to generalize. One exception is the work in [28], which includes only industrial parts (gears, screws, brackets, etc.), obtained from an online CAD repository. Simulated or synthetically generated datasets have been considered as well, however, again focused mainly on household objects [27], [29] and are typically not integrated in a robotics simulator.

Even though most works provide their data and models openly available, reusing them is not an easy task. Different data formats, suitability of models and selecting a subset is not always possible or easy. This can be seen in Table I, which gives a brief overview of CAD model datasets. The most well-known CAD dataset, ShapeNet [12], provides the largest set of CAD models, is carefully categorized and models can be browsed and queried from its hosting website. The ABC [4] and Fusion360 Gallery [6] CAD datasets offer CAD models that are parametric, therefore focusing on the changing of models and their appearance, either by parametrized curves and surfaces [4], or by primitive operations [6]. Finally, the Mechanical Components Benchmark [8] and Thingi10K [13] offer datasets with mechanical and industrial parts, sourced from professional CAD repositories or from open source 3D printing website Thingiverse [14]. Due to these differences, licensing has to be carefully considered, as in some cases commercial use is prohibited.

The main observation to motivate our work is that collecting or generating training data for industrial parts is a tedious, time-consuming and costly task. Even though plenty datasets can found, each are limited (to some extend) to the objects they contain. Instead, we propose an automated pipeline that generates a dataset from CAD models and is integrated in a robotics simulator (Gazebo), where the variability and size of the dataset is controllable by different parameters.

III. Automatic Dataset Generation

A. Data capturing

Automatic data collection, as described in Algorithm 1, utilizes only simulation to generate a dataset, where different parameters are varied to recreate appropriately broad conditions as would be expected in a real use case. Different settings in the simulator influence the data generation, such as camera and image parameters (e.g., resolution, focal length, opening angle, etc.), environment conditions (e.g., lighting and background) and object properties (e.g., texture). A moderately dense clutter of all the objects in the dataset is created through CAD and loaded as mesh in the Gazebo simulation environment. Then, a 3D point cloud of each object is generated by sampling 3000 points from each object mesh and saving each as point cloud. This is required for generating the object binary masks, i.e., projection from 3D point cloud objects to 2D binary masks, and represented by function CAD2PCL() in Algorithm 1.

Hemisphere-sampling, as described in [25], is then carried out around the object-set origin and with the simulated camera's principle-axis pointing towards it (see Fig. 1). The camera pose is then varied with steps in:

- yaw increments, $\phi = \{0, \ldots, 360\}$ in degrees,
- pitch increments, $\theta = \{0, \ldots, 90\}$ in degrees,
- hemisphere radius increments, $s = \{0, \ldots, 3\}$ in meters.

The dataset then records the following data:

- RGB image I_{RGB}
- Depth image I_D
- Object binary mask I_M
- Encoded object class labels
- Ground truth object pose in camera- and world frame, T_O^C, T_O^W (4×4 homogeneous transformation matrices)

Object binary masks are represented as a grey-scale image I_M with the mask of each object in a different grey-scale value, such that they can be independently extracted. This is computed in Algorithm 1 by GenerateMask(I_D) as follows. First, as described earlier, a point cloud depth image I_D is projected onto the image plane and clipped to the correct simulated camera resolution. Gaps in the projected image are filled with a flood fill operation. Then, each projected point is mapped to a unique grey label and subsequently classified into a set, with size the number of objects in the scene. This leads to each projected object having a unique grey-scale value, representing the binary mask of the object.

B. Evaluation

The suitability of the generated datasets is evaluated by two state of the art models; PVN3D [19] for pose estimation and 6-DOF GraspNet [18] for grasp detection. PVN3D is a deep point-wise 3D keypoint voting network that takes both RGB and depth images as input. It consists of separate blocks for feature extraction, 3D keypoint detection, instance semantic segmentation and 6-DOF object pose estimation. A joint multi-task training is carried out for 3D keypoint detection and instance semantic segmentation blocks, and training utilizes the complete generated dataset as described in this work. 6-DOF GraspNet, on the other hand, has trained a grasping model by physics simulation on objects extracted from ShapeNet [12] and only takes point cloud images as input. Evaluation with 6-DOF GraspNet therefore serves to demonstrate that the object dataset can also be utilized for only grasp detection evaluation.

Algorithm 1: Automatic dataset generation

Parameters:
ϕ: yaw angle of the camera
θ: pitch angle of the camera
s: scale of the camera
Input : CAD model of object assembly
Output : I_{RGB}, I_D, I_M, labels, T_O^C
Functions : CAD2PCL()
GenerateMask()
Load CAD models in Gazebo
Set correct scale of the objects
Set simulation parameters
CAD2PCL(); (Convert CAD models to point cloud)
foreach ϕ **do**
  foreach θ **do**
    foreach s **do**
      Record I_{RGB}
      Record I_D
      Record $I_M \leftarrow$ GenerateMask(I_D)
      Record labels
      Record $T_O^C = \left(T_C^W\right)^{-1} T_O^W$
    end
  end
end

For evaluating object pose estimation, we utilize two widely used metrics: Average Distance of Model Points (ADD) and Average Closest Point Distance (ADD-S). ADD computes the average distance between two points using the ground-truth pose and the estimated pose. ADD-S computes the mean distance from a model point by the estimated pose to its closest neighbor on the target 3D model point transformed by the ground truth pose. Therefore, the ADD-S metric can solve the ambiguous problem of symmetrical objects. Evaluation of the grasp detection network is done by assessment of the generated grasps and their suitability.

C. Implementation

Gazebo was selected for the simulation environment, as it's a common tool in robotics research and provides fine-tuning of a variety of parameters, e.g., gravity, mass, friction, inertia and lighting (ambient, diffuse, directional, spot, etc.). This provides a good testing-ground for a complete robotic grasp and manipulation pipeline. A Kinect v1 camera is simulated in Gazebo which publishes color images, depth images and camera intrinsics for both over ROS. All objects are simulated using their standard polygon mesh files converted from CAD models, by Open3D [30]. Only ambient and diffuse lighting (i.e., no directional or spot light) is used

with a fixed color for each object. The background is set to a brightly-lit plain-gray room with no walls and objects always rest on the floor in their most stable equilibrium pose. For the sake of simplicity, no dense background clutter or non-dataset objects are added to the environment. However, these properties can be changed by altering the particular settings.

IV. Results

Results are reported towards dataset generation and its use to train the pose estimation and grasp detection models.

A. Dataset generation

CAD models of objects, as loaded in Gazebo, can be organized in different arrangements, as shown in Fig. 2. This depicts identical objects with different orientations and object sets in different configurations. Such setup can be utilized for the generation of training data and for the evaluation of the learned detection models. For the evaluation of the complete data generation pipeline, three different industrially-relevant object assemblies are selected, which are shown in Fig. 3. Assemblies comprise of Diesel engine parts, sourced from a local Diesel engine manufacturer (Fig. 3a) and an assembly benchmark set (Fig. 3a), which is designed by hand, a slider-crank mechanism (Fig. 3b), which is designed by hand and a planetary gearbox (Fig. 3c), sourced from the open source 3D printing repository Thingiverse [14]. Datasets are then generated as follows.

The parameters of the simulated camera are taken from the standard implementation of Kinect v1 in Gazebo, i.e., RGB-D resolution of 640 $\times$ 480. The steps in sampling are:

- 10° yaw increments, $\phi = \{0, 10, \ldots, 360\}$,
- 10° pitch increments, $\theta = \{0, 10, \ldots, 90\}$,
- $0.1m$ hemisphere radius scale increments, $s = \{0.65, 0.75, \ldots, 1.45\}$.

This procedure generates a total of 3000+ data samples (total number of yaw, pitch and scale configurations), where one sample includes the RGB and depth images, binary object masks, object labels and poses. Examples of this data can be seen in Fig. 3. This dataset has an approximate size of 200 MB, which is irrespective of the number of objects in the set. Dataset generation is implemented on a standard laptop with i7-9850H processor and quadro T1000 GPU, running Ubuntu 16.04 and ROS Kinetic, and takes around 2 hours for 3000+ samples ($\sim$0.25 seconds per object).

B. Pose estimation results

With a total of 3000+ collected data samples, a 75%-25% train-test split was used where every 4-th sample is used as a test sample, in order to evenly cover all possible pitches, yaws and scales in both test and training dataset. Input image size for training is 640 by 480 pixels and a total 12288 points are randomly sampled for PointNet++ feature extraction [31]. Only these points are further used for semantic labeling and keypoint-offset voting, which is an optimal number originally recommended and tested by the authors. If the number of points in the pointcloud are less than this number, the pointcloud is recursively wrap-padded around its edges until is has at least 12288 points.

For each object mesh, greedy farthest-point-sampling is used to sample keypoints that spread-out at a furthest possible distances from each other on the mesh surface. Two different versions i.e., 8 and 16 keypoints, are used for training two separate network checkpoints with a total of 25 epochs and batch size of 24, as recommended by the authors. Training of the pose estimation model [19] was carried out on 4 Nvidia V100 GPUs with 32 GB of memory simultaneously and takes around 2-3 hours for the given batch-size, number of epochs and training dataset.

Fig. 4 depicts the pose estimation results for the three object sets, where individual objects are overlaid with their estimated pose, represented by projected bounding boxes (light-blue). In addition, the accuracy of pose estimation is reported in terms of area under accuracy-threshold curve where the threshold for both ADD and ADD-S metric is set as 10% of object diameter. These results are reported in Table II, for a selection of parts. These results demonstrate that a highly accurate pose estimation model can be trained with our proposed dataset generation pipeline.

C. Grasp detection results

Grasp detection is evaluated on the generated dataset by the originally trained model [18]. As can be seen in Fig. 5, a large number of grasps are generated, which are ordered from red to green (i.e., worst to best), based on the authors' grasp quality metric [18]. Filtering of the generated grasp candidates, by removing impossible or unsuitable grasps, is still required to select a most suitable grasp. For example, from the faceplate (left most object in Fig. 5a), few grasps are generated, and all are unsuitable (red color) as grasping would require moving through the ground plane on which the object is placed. The Diesel engine piston (Fig. 5c) on the other hand, generates many suitable grasps (green color) due to its round shape. Nevertheless, these results demonstrate that the generated data is suitable for inference evaluation by a state of the art grasp detection model.

D. Discussion

The results in this work are only demonstrated in simulation and the object datasets are generated without object variations such as color and texture, environmental conditions such as lighting and background, and no noise was added to the simulated camera images. These variations are possible to be included and, naturally, would increase the size and generation time of the dataset, as well as the time to train a detection model. In addition, the dataset is limited with respect to the placement, pose and clutter of the objects on a flat surface. Only a single object pose is included, which might not necessarily be the pose that the object has, in a real situation, and objects might be in a more dense clutter than is trained on. Such wider variety of training data, again, can be included, yet leads to an increase in data size and training time.

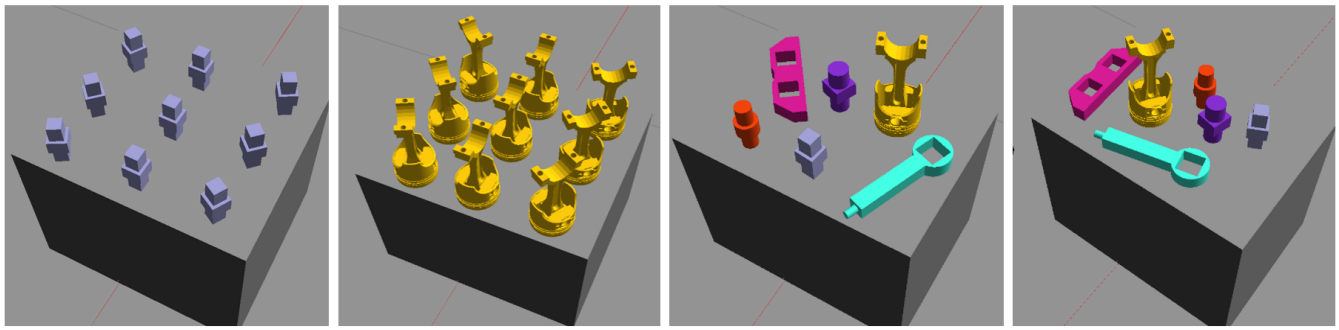
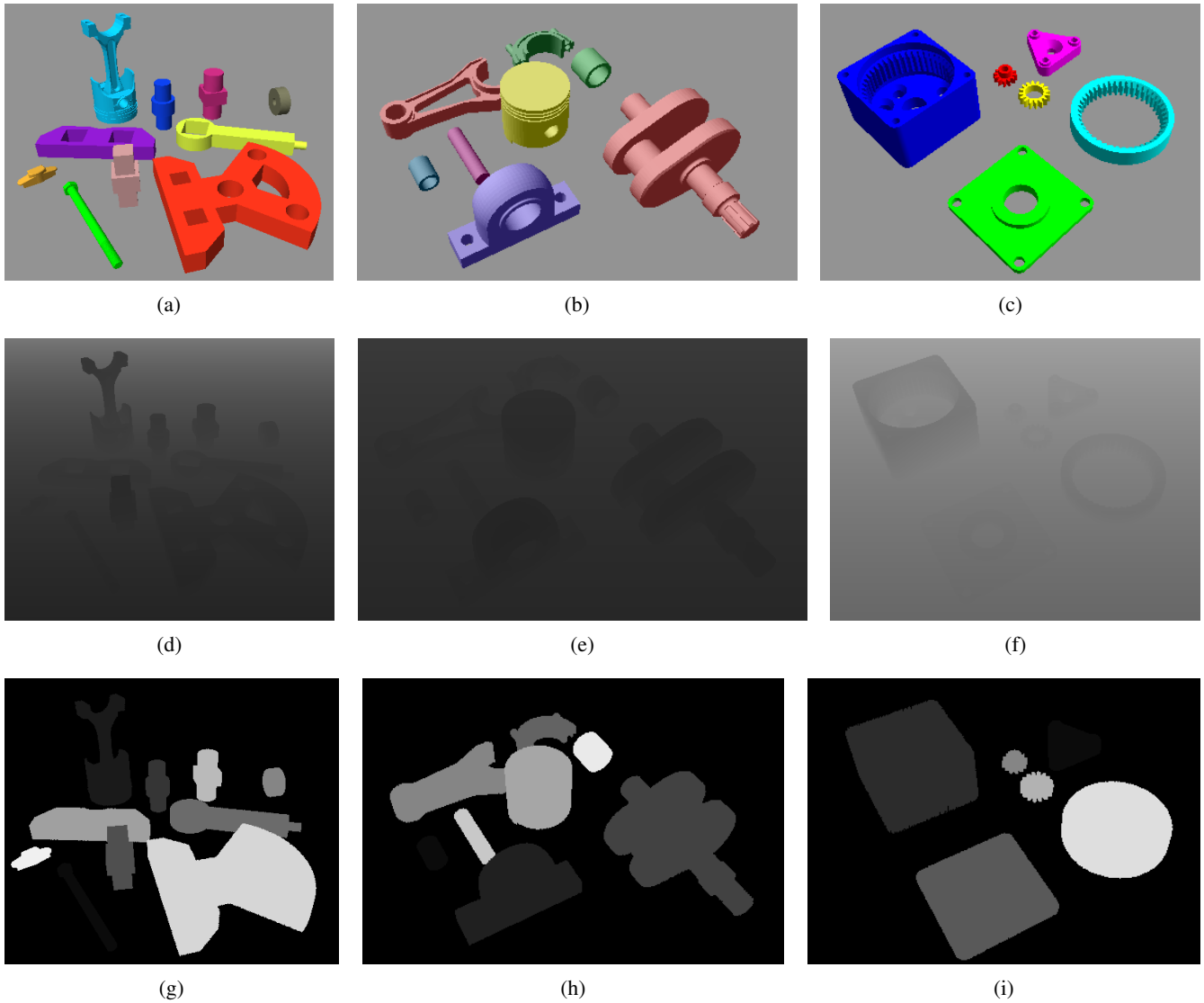


Fig. 2: Samples of objects simulated in Gazebo, in different arrangements and configurations, for the generation of training data and for evaluation of the pose estimation and grasp estimation models.

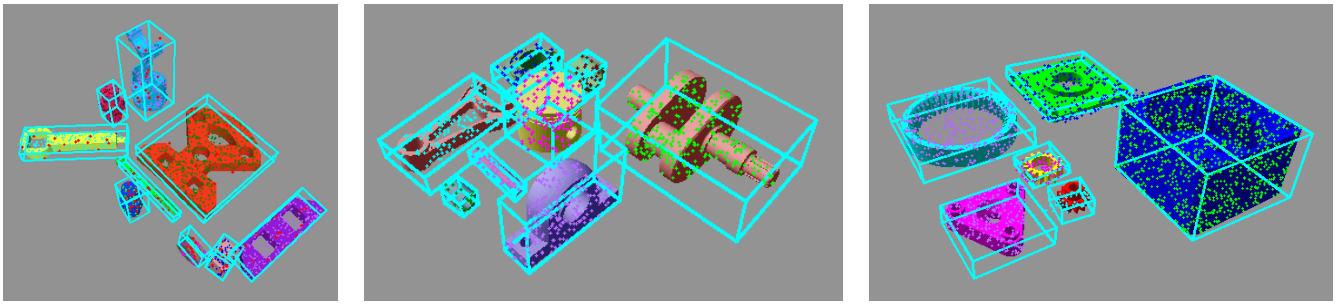


(a) (b) (c)

(d) (e) (f)

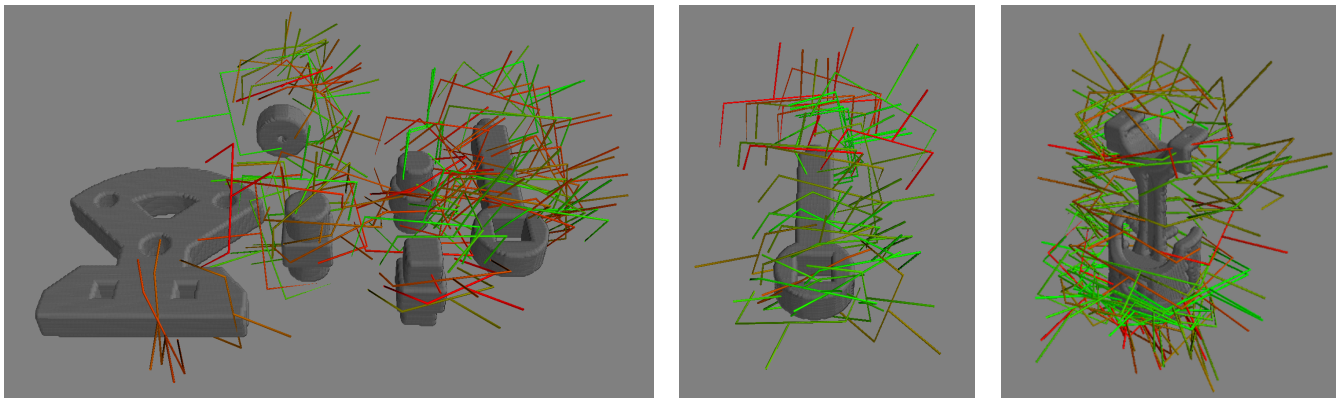
(g) (h) (i)

Fig. 3: Results of the automatic data generation pipeline. Objects in the left column (a, d and g) represent parts from a Diesel engine and assembly benchmark. Objects in the middle column (b, e and h) represent parts from a slider-crank mechanism. Objects in the right column (d, f and i) represent parts from a planetary gearbox. Top row (a, b and c) depicts RGB images. Middle row (d, e and f) depicts depth images. Bottom row (g, h and i) depicts object binary masks.



(a) (b) (c)

Fig. 4: Results of object pose estimation with the PVN3D [19] network, trained with the complete generated datasets, as proposed in this work. Individual objects are overlaid with their estimated pose, represented by projected bounding boxes (light-blue), demonstrating that a highly accurate pose estimation model is trained from the dataset.



(a) (b) (c)

Fig. 5: Results of object grasp detection with the 6-DOF GraspNet [18] network, in which grasps are ordered red to green (i.e., worst to best), based on the authors' grasp quality metric. Filtering of the generated grasps candidates is still required to select a most suitable grasp. These results are presented to demonstrate that the generated dataset can also be used for grasp detection evaluation. Parts are from an assembly benchmark (a) and (b), and a piston from a Diesel engine (c).

Practical considerations that should be taken into account when adopting this framework are as follows. The scale at which an object is loaded into the Gazebo simulation environment should be checked carefully for each object. If the correct size of an object is not utilized or included, the simulated camera will not observe the objects at the correct scale. This results in objects too big or too small for the camera, or objects that are not visible (outside the field of view). This recommendation also holds for the position of objects, which should be centered at the origin of the simulation world individually or as an object set.

Extension to other domains, such as physics-based simulation to train an object grasping model (e.g., by deep reinforcement learning), could be considered as well, as parts of the data generation pipeline could be utilized as input for the modelling of objects. Besides object shape, additional properties, such as object texture, mass or friction, or even environment conditions and different gripper and their parameters could be varied in the dataset and in the training framework.

The proposed data generation pipeline offers a structured way to control which object should be included or excluded in a dataset. This is possible by loading individual CAD object models sequentially and arranging them in a suitable configuration, or by loading a CAD object model set at once. This implies that such CAD models need to be available or designed beforehand, to be included (e.g., provided by companies, extracted from existing datasets or repositories; see Table I). Besides manual human effort, object model variations can also be integrated automatically. Most CAD programs offer some form of automated or parametric design via their API or internally by scripting, thereby enabling CAD design to be integrated in the automated data generation pipeline. For example, open source CAD design tools such as FreeCAD [32], provide a scripting interface for Python and OpenSCAD [33] enables scripting solid 3D CAD models without any interactive modelling interface. Geometric variations of the object itself, as is the case for standardized parts such as gears, can thus be automatically included without manually altering individual CAD files. Future work will explore this further.

TABLE II: Area under curve (AUC) for accuracy-threshold curve for the ADD and ADD-s metric on a selection of objects. These results show that the generated datasets are suitable for training the pose estimation model PVN3D [19].

	Piston	*Round peg*	*Square peg*	*Pendulum*	*Pendulum-head*	*Separator*	*Shaft*	*Face-plate*	*Valve-tappet*	*Shoulder-bolt*	*Bearing*	*cam*	*Conrod-lower*	*Conrod-upper*	*Piston-pin*
ADD															
8 keypoints	97.21	97.05	96.32	95.31	96.73	95.95	97.41	95.88	91.58	91.4	97.45	97.55	97.33	97.04	95.16
16 keypoints	97.76	97.35	97.12	96.09	97.27	96.4	97.93	96.16	91.7	93.5	89.99	88.46	86.29	76.06	87.09
ADD-S															
8 keypoints	97.85	97.05	96.32	95.31	96.73	95.95	97.41	95.88	91.58	91.4	98.12	98.13	98.39	97.04	95.16
16 keypoints	98.16	97.35	97.12	96.09	97.27	96.4	97.93	96.16	91.7	93.5	95.39	94.28	94.08	76.06	87.09

V. CONCLUSION

This work proposed an automatic data collection pipeline for robotic vision-based tasks such as grasping and pose estimation. The motivation of this work stems from freely available datasets that offer a wide variety of objects, yet are difficult to utilize and expand. Data collection is a cumbersome and time-consuming task, often limited by unavailable physical objects. With object CAD models as input, our pipeline generates data in the form of RGB-D images, object binary masks, labels and ground truth poses. Results with three different assemblies, containing varying parts, demonstrates the utilization of the generated datasets to train and evaluate a pose estimation and grasp detection model. The proposed pipeline is implemented in ROS with Gazebo as simulator and open source available.

ACKNOWLEDGEMENTS

Project funding was received from Helsinki Institute of Physics' Technology Programme (project; ROBOT) and European Union's Horizon 2020 research and innovation programme, grant agreement no. 871449 (OpenDR) and no. 871252 (METRICS). The authors wish to acknowledge CSC - IT Center for Science, Finland, for computational resources.

REFERENCES

[1] P. H. Joshi, *Jigs and fixtures*. Tata McGraw-Hill Education, 1998.
[2] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review," *Artificial Intelligence Review*, vol. 54, p. 1677–1734, 2021.
[3] S. Caldera, A. Rassau, and D. Chai, "Review of deep learning methods in robotic grasp detection," *Multimodal Technologies and Interaction*, vol. 2, no. 3, p. 57, 2018.
[4] S. Koch *et al.*, "ABC: A big CAD model dataset for geometric deep learning," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9601–9611.
[5] Onshape, https://www.onshape.com, accessed on 23.11.2020.
[6] K. D. Willis *et al.*, "Fusion 360 gallery: A dataset and environment for programmatic CAD reconstruction," *arXiv:2010.02392*, 2020.
[7] Autodesk Online gallery, https://gallery.autodesk.com/fusion360, accessed on 23.11.2020.
[8] S. Kim, H.-G. Chi, X. Hu, Q. Huang, and K. Ramani, "A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks," in *Eur. Conf. on Computer Vision (ECCV)*, 2020, pp. 175–191.
[9] Traceparts, https://www.traceparts.com, accessed on 23.11.2020.
[10] 3D Warehouse, https://3dwarehouse.sketchup.com, accessed on 23.11.2020.
[11] GrabCAD, https://grabcad.com, accessed on 23.11.2020.
[12] A. X. Chang *et al.*, "Shapenet: An information-rich 3D model repository," *arXiv:1512.03012*, 2015.
[13] Q. Zhou and A. Jacobson, "Thingi10K: A dataset of 10,000 3D-printing models," *arXiv:1605.04797*, 2016.
[14] Thingiverse, https://www.thingiverse.com, accessed on 23.11.2020.
[15] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012.
[16] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis - a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2013.
[17] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.
[18] A. Mousavian, C. Eppner, and D. Fox, "6-DOF graspnet: Variational grasp generation for object manipulation," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2019, pp. 2901–2910.
[19] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "PVN3D: A deep point-wise 3D keypoints voting network for 6DOF pose estimation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 632–11 641.
[20] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Eur. Conf. on Computer Vision (ECCV)*, 2014, pp. 740–755.
[21] A. Ahmadyan, L. Zhang, J. Wei, A. Ablavatski, and M. Grundmann, "Objectron: A large scale dataset of object-centric videos in the wild with pose annotations," *arXiv:2012.09988*, 2020.
[22] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2018, pp. 3511–3516.
[23] C. Eppner, A. Mousavian, and D. Fox, "ACRONYM: A large-scale grasp dataset based on simulation," *arXiv:2011.09584*, 2020.
[24] 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.
[25] S. Hinterstoisser *et al.*, "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *Asian Conference on Computer Vision (ACCV)*, 2012, pp. 548–562.
[26] B. Calli *et al.*, "The YCB object and model set: Towards common benchmarks for manipulation research," in *IEEE Int. Conf. on Advanced Robotics (ICAR)*, 2015, pp. 510–517.
[27] T. To *et al.*, "NDDS: NVIDIA deep learning dataset synthesizer," 2018, https://github.com/NVIDIA/Dataset_Synthesizer.
[28] J. Zhao, J. Liang, and O. Kroemer, "Towards precise robotic grasping by probabilistic post-grasp displacement estimation," *arXiv:1909.02129*, 2019.
[29] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam, "Blenderproc," 2019.
[30] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.
[31] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv:1706.02413*, 2017.
[32] FreeCAD, https://www.freecad.org/, accessed on 21.01.2020.
[33] OpenSCAD, The Programmers Solid 3D CAD Modeller, https://www.openscad.org/, accessed on 21.01.2020.