SingleDemoGrasp: Learning to Grasp From a Single Image Demonstration

Amir Mehman Sefat[1], Alexandre Angleraud[1], Esa Rahtu[2] and Roel Pieters[1]

***Abstract*— Learning-based grasping models typically require a large amount of training data and training time to generate an effective grasping model. Alternatively, small non-generic grasp models have been proposed that are tailored to specific objects by, for example, directly predicting the object's location in 2/3D space, and determining suitable grasp poses by post processing. In both cases, data generation is a bottleneck, as this needs to be separately collected and annotated for each individual object and image. In this work, we tackle these issues and propose a grasping model that is developed in four main steps: 1. Visual object grasp demonstration, 2. Data augmentation, 3. Grasp detection model training and 4. Robot grasping action. Four different vision-based grasp models are evaluated with industrial and 3D printed objects, robot and standard gripper, in both simulation and real environments. The grasping model is implemented in the OpenDR toolkit at: `https://github.com/opendr-eu/opendr/tree/master/projects/control/single_demo_grasp`.**

***Index Terms*— Grasping, Deep Learning in Grasping and Manipulation, Perception for Grasping and Manipulation**

I. INTRODUCTION

Collaborative robots have gained popularity in industry as they are designed to be safe, particularly where human and robot share the workspace. Accompanied by intuitive programming interfaces, robot tasks can be programmed efficiently [1]. Despite the benefits, the application of cobots in industrial settings are mainly limited to offline tasks where the actions and targets are defined to the system beforehand [2]. For example, in the majority of pick and place tasks, object poses are fixed, and the robotic arm should reach a predefined grasp pose. Although there is great interest in the generation of object grasp models from visual data, [3], limitations still exist, for example, in terms of object type coverage, grasp success, training complexity, model inference time, etc. In particular, while grasp models have reported high success rate (e.g., Dex-Net 4.0 [4] achieves above 95% accuracy), this typically only holds for the task at hand, i.e., bin picking with generic household items. Evaluating such grasping model on objects that exhibit different properties (e.g., industrial parts) might result in unsuccessful grasp attempts and an overall lower accuracy. In addition, grasp modelling requires vast amounts of training data and considerable training time on high-performance computing clusters. Consequently, state of the art grasping models can be large in size and slow to execute [5].

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

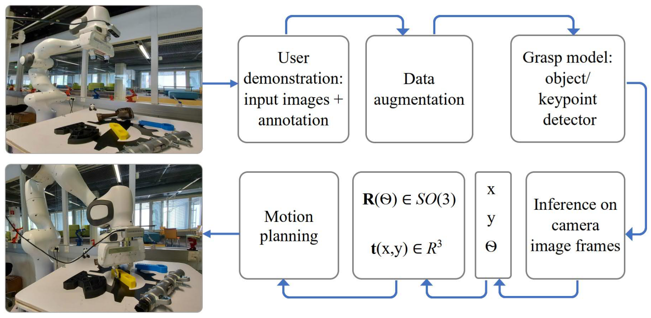

User demonstration: input images + annotation
Data augmentation
Grasp model: object/ keypoint detector
Inference on camera image frames
x
y
Θ
$\mathbf{R}(\Theta) \in SO(3)$
$\mathbf{t}(x,y) \in R^3$
Motion planning

Fig. 1: Overview of the proposed grasping model.

Extending an existing dataset and retraining a grasping model is in most cases not an option, due to unavailable data or limitations in resources and computation power. These problems exist in particular for small and medium sized enterprises (SME), which typically don't have the knowledge and resources available for data collection, model training and fine-tuning.

Our main observation to motivate this work is that collecting or generating training data for a grasp detection model is a tedious, time-consuming and costly task, which is often out of reach in industrial environments. Even though plenty datasets can be found [6], [7], each are limited (to some extend) to the objects they contain. Industrial SMEs require the handling of objects that, in most cases, do not resemble objects in these datasets, or the objects themselves can change depending on a customer's requirement. Moreover, as the handling of such objects requires a human pre-selected grasp pose, a single generic model for all objects is unfeasible.

In this work, we aimed to tackle this issue by investigating visual learning-based approaches for object grasp detection, with human annotation of a desired object grasp pose. For this, different variants of the R-CNN architecture from Detectron2 [8] are evaluated for the fast generation of a grasping model. Single or multiple image demonstrations with human annotations of an object grasp are collected and utilized to generate an augmented object training dataset, from which a detection model is trained. Object grasp detection results (object grasp position and orientation on a plane) are transformed to a 3D grasp pose and given as input for robot motion planning (see Fig. 1). Four different networks are developed and evaluated in simulation (Webots) with eight different objects. The grasp detection model with best performance was then implemented and evaluated in real robot experiments (Franka robot with standard gripper).

The main contributions of our work are as follows:

- **Four different planar grasp models** based on pre-trained object detection models.
- **Data generation by image augmentation** of image demonstrations with human annotation.
- **Training of grasp detection networks** in short time.
- **Evaluation of the grasping approach** in simulation and with real experiments.

The paper is organized as follows: Section II reviews related works and state of the art in computer vision and grasping methods. Section III and Section IV define the considered problem statement and research methodology, respectively. Section V describes the implementation details of the proposed grasping model, and Section VI reports the results and provides an analysis. Finally, Section VII concludes the work.

II. Related Work

In the context of robotics, object detection, pose estimation and grasp detection are closely related, as grasp poses or grasp actions can be directly generated from an object pose. This section presents a brief overview of related approaches.

A. Object Detection and Pose Estimation

Traditionally, object detection and pose estimation algorithms have utilized classical 2D features that exploit local salient details, such as corners, edges and ridges. Well-known detectors like SIFT [9] can extract robust keypoints from a scene by relying on texture on objects or of the scene itself. Texture-less keypoint detection, on the other hand, utilizes geometrical primitives as features in methods such as BIND [10]. In addition, alternatives to traditional keypoints are template matching, where a image patch provides the template to localize within an image, or deep features that extract keypoints based on high-level cues captured by convolutional neural networks. The latter is a recent development that has gained popularity due to their data-driven property and promising performance [11], as compared to hand-crafted features. Analogous to 2D keypoints for RGB images, 3D keypoints can be extracted from 3D data representations, such as pointclouds or volumetric images [12]. Following the detection of keypoints from a raw image, follow-up steps include the description of the keypoint and the matching of them over two or multiple images. In a similar manner, Convolutional Neural Network (CNN) based detectors, such as Faster R-CNN [13], could be utilized to detect objects, after which a grasp pose needs to be be extracted.

Object pose estimation on the other hand directly estimates the 6D pose of an object. Similar to object detection, different approaches exist, such as correspondence-based methods 3DMatch [14], template-based methods such as PoseCNN [15] and voting based methods such as DenseFusion [16]. Again, once an object pose is extracted, this needs to be converted to a grasp pose suitable for a robot to hold an object.

B. Grasp Detection

Object grasp detection aims to derive a grasp pose directly from sensor measurements 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. 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, where the center of the box is the grasp position [17], [18]. On the other hand, in case a complete 3D pose is required for grasping, detection should return the full 3D position and 3D orientation [19]. In context of learning-based grasp detection, typical data-driven approaches differentiate between the utilization of RGB [20], depth (in form of pointclouds [21], [22]) or a combination of both (RGB-D, [23], [24]). 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 [25].

The methods explained generate a grasp pose and require motion planning to execute a grasping action. Such motion planning approaches can be generally listed as motion primitive-based methods, imitation learning and reinforcement learning methods [5], [25].

C. Datasets

Existing datasets for 2D object detection, such as Pascal VOC [26], COCO [27] and, more recently, Objectron [28] for 3D objects, are widely available, including common objects that are present in everyday scenes. There are also datasets designed specifically for grasping such as EGAD! that contains 3D meshes with diverse properties [6] to cover variations in object properties, and datasets that utilize simulation for the grasp data collection, e.g., Jacquard [29] and ACRONYM [7]. These publicly available datasets include different categories of objects enabling a reasonable comparison and performance evaluation of grasping models. However, they are not suitable for applications where the target objects are not included in the dataset, simply because no success rate can be guaranteed.

III. Problem Statement

The robot object grasping scenario considers a robot manipulator with standard gripper and objects that are located on a planar table in front of it (see Fig. 2). Objects of interest are unknown beforehand (e.g., industrial objects) and can have both simple or complex geometry. All objects should allow for a stable grasp, without alteration to the gripper or object pose and be light enough to be lifted (< 1 kg). As general rule, we denote that each object can be represented by a 2D planar position and 1D orientation $\{x, y, \theta\}$, from which a grasp pose is extracted, with rotation and translation defined as $\mathbf{R}(\theta) \in SO(3)$ and $\mathbf{t}(x, y) \in \mathbb{R}^3$. Our observation is that one common grasp model for a selection of objects is difficult to generate. Instead, our approach aims to generate

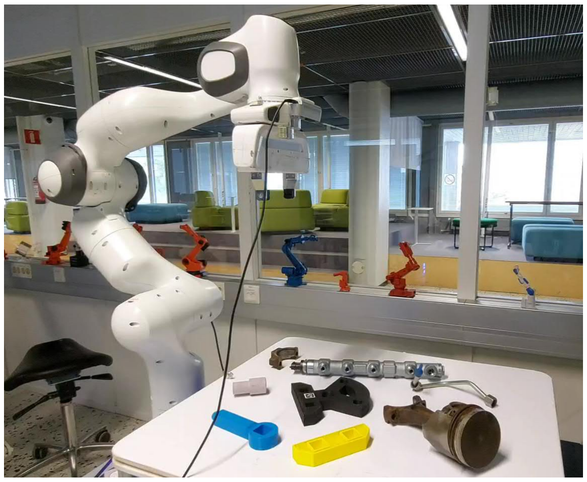


Fig. 2: Object grasping scenario.

a grasp model for individual objects, thereby avoiding modelling conflicts with objects that have different properties. In addition, grasp models need to be generated and deployed fast, without large computational resources, thus restricting the data generation and model training process. This implies that for perception only RGB images are used, with a camera located on the end-effector of the robot.

In summary, the grasping problem can then be stated as follows: from a single image demonstrations of an object annotated with its grasp, generate suitable training data and train a grasp detection model that can run in real-time to successfully grasp the object.

IV. Methodology

Four different grasp detection modules are developed and implemented to find the most robust approach for extracting planar grasp poses. In all cases, the grasping approach consists of the following four distinct steps (Fig. 1):

1) **Human input** - captures and annotates the object in the field of view of the camera.
2) **Training data** - is generated automatically by applying data augmentation techniques.
3) **Object grasp pose** - is estimated based on different state of the art neural networks.
4) **Grasping action** - is done after converting planar grasp to 3D Cartesian pose.

Following, we describe the four different grasp detection modules and their required image annotations. An overview of the models is depicted in Fig. 3 and described in Table I.

A. *Faster R-CNN-based Grasp Detection*

Model A separates the object grasp location and orientation estimation into two different detection models, i.e., Faster R-CNN and CNN, respectively. The Faster R-CNN network takes images and their corresponding annotation as input for training, and generates a bounding box around the objects if they are present in the image scene. The center location of the bounding box is then used as grasp position. In order to predict the grasp orientation, a CNN network is implemented where the final layer consists of 360 output nodes to represent the object's orientation. The first layer of this CNN network accepts image arrays with a size of $(224 \times 224 \times 3)$ to extract features and classifies the object based on the highest score to predict the corresponding orientation. Input annotation on the image is done by defining a bounding box around the object. One additional step is required for the orientation, by using the bounding box to crop and resize the region of interest that are labeled with the corresponding orientation.

B. *Keypoint R-CNN-based Grasp Detection by Bounding box*

Model B utilizes the Keypoint R-CNN network to detect both object, represented as a bounding box, and keypoints of an object. For human annotation, a bounding box and two keypoints on the object need to be defined. The keypoints represent the reference orientation for the grasp, from which the augmentation will add ten more keypoints. The estimated bounding box center can then be used as the robot's intermediate hover position, before a relative orientation is extracted from the keypoints to form the grasp pose.

C. *Keypoint R-CNN-based Grasp Detection*

Model C is an improvement of Model B by retrieving the grasp position and orientation directly from the keypoints. Therefore, the same network as in model B is used, and the bounding box information is not utilized. Annotation follows the same approach as Model B, with 12 keypoints used in total for grasp detection.

D. *Mask R-CNN-based grasp detection*

Model D utilizes Mask R-CNN to predict an object mask and returns a planar object position and orientation. This is possible, as the grasping approach only requires 2D object information, based on the object mask that separates the object from the background. One additional step is necessary to determine the grasp orientation, which is done by converting the mask over the object to a binary image followed by local feature extraction methods (SIFT) to estimate the relative orientation. Annotation of the input image requires a bounding box to be drawn around the object and also a set of keypoints to construct a mask/polygon around the object.

E. *Augmented Dataset Generation*

Dataset generation utilizes a single or multiple input images (RGB, $480 \times 640 \times 3$, see Fig. 4a) taken above the workspace in which the target object is visible. These images are then annotated by a person as explained in previous sections and as illustrated in Fig. 4b, depending on their corresponding model input format (see Table I). Then, a sequence of image augmentation techniques are performed to the input annotation, consisting of cropping, zooming, rotation, translation, etc. (see Fig. 4c). For each model, 1500 training samples and 200 validation samples are generated, for position and orientation data, respectively. For the CNN network of model A, the number of generated samples for training an orientation predictor is 5000. More details about each model can be found in Table I.

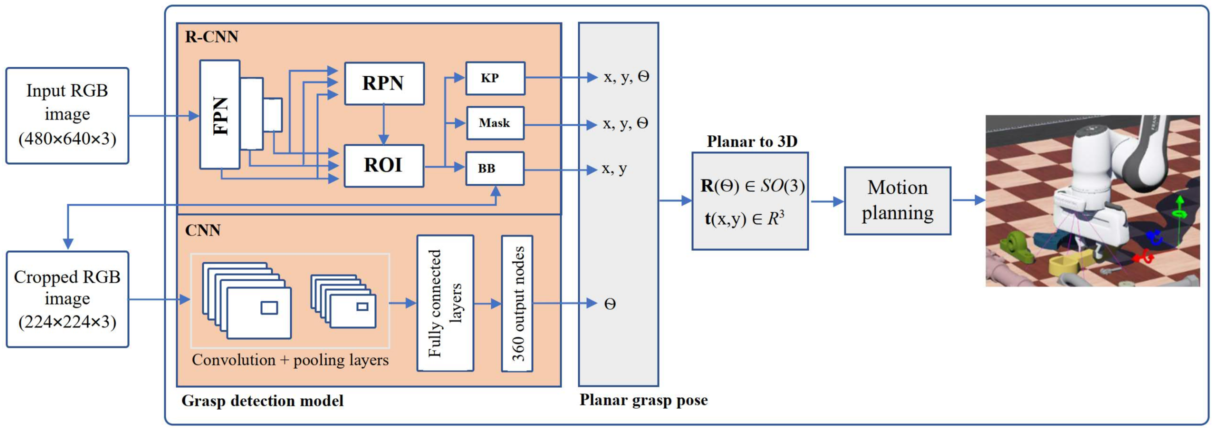


Fig. 3: Overview of the grasp detection approach from a single RGB image demonstration. The grasp detection models utilize the R-CNN architecture family from Detectron2 to generate keypoints, mask and bounding box and extract an object grasp pose. Separately, one CNN network is developed to obtain the object grasp orientation for model A. Abbreviations denote; FPN: Feature Pyramid Network, RPN: Region Proposal Network, ROI: Region Of Interest, KP: Keypoints, BB: bounding box.

TABLE I: Object grasp detection models. Abbreviations: BB - bounding box, KP - keypoints, CL - class label.

	Object position estimation		**Object orientation estimation**			
Grasping model	**Pre-trained model**	**Output format**	**Pre-trained model**	**Method**	**Human input annotation**	**Training data**
A	Faster R-CNN	BB	CNN	Classification	BB	BB, cropped box
B	Keypoint R-CNN	BB and KP	Keypoint R-CNN	KP	BB, 2 KP	BB, KP, CL
C	Keypoint R-CNN	KP	Keypoint R-CNN	KP	2 KP	KP, CL
D	Mask R-CNN	BB and object mask	Mask R-CNN	mask + SIFT	BB, mask	BB, mask, CL

V. Implementation

Implementation of the proposed grasping models include the overall architecture, the grasp detection networks, as well as simulation environment.

A. Architecture

The grasping approach is divided into two major computation nodes, which are explained as follows.

Perception - feeds the neural network models with the input images, runs inference and generates an output. The raw output of the models as explained in Section IV are then used to calculate the planar grasp pose of the object with respect to the image plane. This 2D information is then transformed into 3D coordinates in the world frame and sent to the motion controller node. The transformation from 2D to 3D is done by utilizing a pin-hole camera model, as all the intrinsic and extrinsic parameters of the camera are available. The structure of the perception node is illustrated in Fig. 3.

Motion control - generates the actions and motions of the robot manipulator and gripper in order to execute a grasp. It receives input from the perception node, and directly commands a grasping action. Motion generation is done using ROS MoveIt[1], with a Cartesian position controller running on the robot at 1000 Hz.

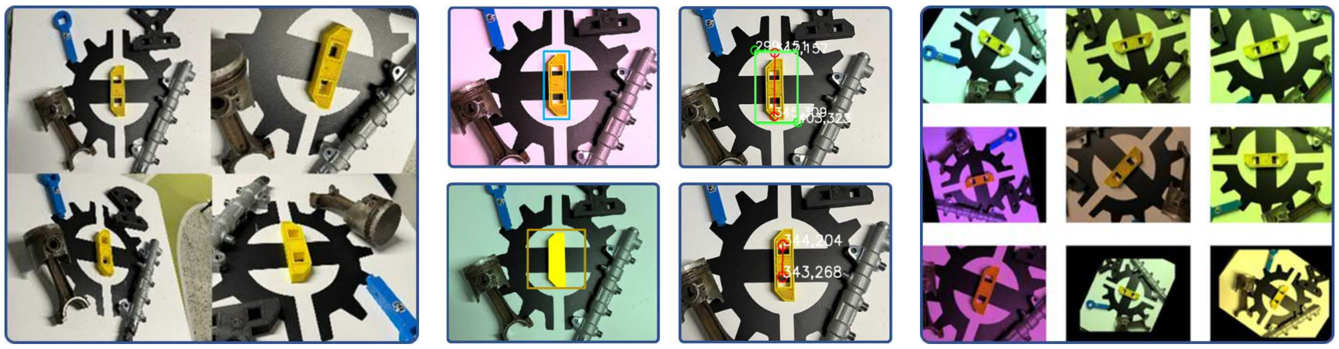
[1]https://moveit.ros.org

B. Grasp Detection Networks

All the models utilize Detectron2 [8] as their perception module. The generated training data and their corresponding labels are fed to the learner, according to the corresponding model input format (see Table I), resulting in one unique grasp detection model for each object. The object grasping approach is implemented in PyTorch, with ROS for communication, and integrated in the OpenDR toolkit [30]. As for the training hyperparameters, object detection for model A utilizes a learning rate of 0.005 and 8 images per batch to train for 500 iterations. For orientation prediction in model A training utilizes a batch size of 32, for 15 epochs with a categorical cross-entropy loss function. Models B, C and D utilize the same hyperparameters, with a learning rate of 0.0008, 2 images per batch for 1000 iterations.

C. Simulation Environment

For fast evaluation of the developed grasp detection models, the entire grasp detection and execution framework has been implemented in robotics simulation. This enables grasping models to be assessed without costly robotic hardware, speeding up developments considerably. For this purpose, 3D models of all objects are included and relevant object and physics parameters can be changed to understand the capabilities and limitations of the grasping models. Webots (see Fig. 5a) is utilized to demonstrate the functionalities, and is freely available to the research community.



(a) (b) (c)

Fig. 4: Training data generation for the planar grasp models. (a) depicts examples of input images for human annotation. (b) depicts the annotations for the four different grasp models (bounding box, keypoints and/or mask, see Table I). (c) depicts augmented images, with variation in brightness, translation, rotation and scale, generated from input and annotated images.

VI. Results and Comparison

Experimental results, their analysis and a comparison to other related work follows in this section.

A. *Object Grasping Scenario*

To evaluate the performance of the developed models, the grasping scenario is first performed in simulation, after which the best performing model is evaluated by experiments on a real robot. In both cases the scenario includes a collaborative robot (Franka Emika), RGB camera (Intel Realsense D-435) and standard gripper (see Fig. 2). The objects selected for evaluation are depicted in Fig. 5b and include parts from a real Diesel engine and several 3D-printed objects. Variations in object properties are thereby included in terms of mass distribution, texture, symmetry and scale, which is useful as human annotation input determines the object grasp pose. For example, the Diesel engine fuel line (curved pipe) has a small width and a non symmetric shape with even mass distribution, while the Diesel engine piston has a symmetric shape, low aspect ratio and an uneven mass distribution. All objects are placed at random configuration on the table in front of the robot and 10 robot grasp attempts are executed for each object from different robot starting configurations.

B. *Grasp Detection Results*

Table II lists the performance of each developed model for all objects, expressed as the percentage of successful grasps. For each object 10 grasp detections and grasp attempts are made, therefore, for each model 80 grasp attempts are made in total. While the success rate of all models are within a similar range (i.e., between 78%-93%), some crucial differences can be identified as follows.

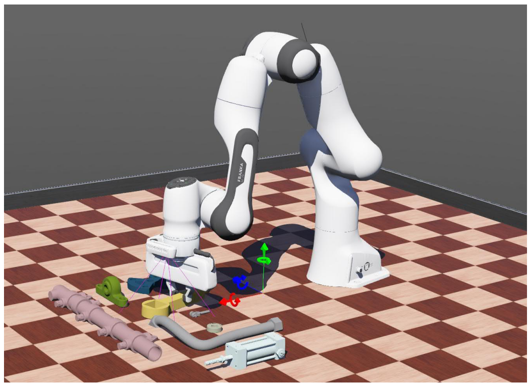
Model A is essentially a two-stage detector, with two separate training datasets and training steps, and, during inference, two consecutive predictions, to estimate the position and orientation of an object grasp. This, unfortunately, makes the model computationally and practically less efficient, compared to the other models. The high success rate is found to be due to a more robust bounding box detection by Faster R-CNN, compared to the keypoint detectors, in terms of the Minimum Area Rectangle (MAR).

Model B and C utilize keypoints for object grasp detection, with the difference that model B utilizes an estimated bounding box for the grasp orientation, while model C extracts this information from the keypoints themselves. While model C demonstrated the highest success rate among all models, keypoints have the limitation that a direct relation between detected keypoints and the actual grasp pose is difficult to realize. We discuss this further in Section VI-D. The high grasp success rate is in this case also achieved by increasing the number of input images and their annotations to seven.

Finally, model D achieved the lowest grasp success rate, partly due to the complexity of detecting the mask of an object. This requires a large number of features on the object, complicating the problem when objects share similarities with each other. In addition, the annotation of an input image is slightly more difficult as the user has to draw a mask/polygon over the object. Fig. 6 depicts several successful object grasping results with model C.

C. *Computational Performance*

All developed models could be trained to relatively high success rate (i.e., $\approx 80\%$) with a manageable dataset size. This implies around 1500 image samples, leading to a training time below ten minutes. The only exception is model A, where a slightly larger set of images was required for the object grasp orientation estimation and a longer overall training time. All trained models are light-weight (below 0.5GB), meaning they allow for training and real-time execution on a standard GPU (see Table II). In all, with the required data augmentation, dataset size and grasp model training time, it is possible to generate an object grasp model from single or multiple image demonstrations in under 15 minutes. This is very short, compared to other state of the art, e.g., 24hrs in case of [4]. However, it has to be noted that such comparison should take into account crucial differences between each method, such as grasp representation (planar vs. 6DOF) and image format (RGB vs. depth).


(a)

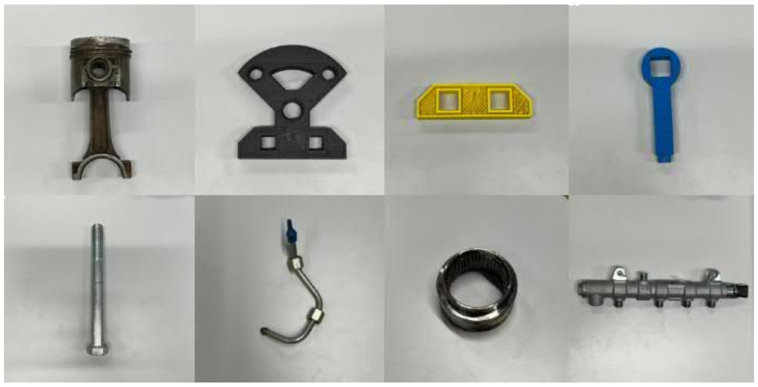

(b)

Fig. 5: Evaluation of the grasp detection models is done in Webots simulation environment (a) and by experiments with real objects (b). Objects used for evaluation include Diesel engine parts and 3D printed objects.

TABLE II: Object grasping results are averaged for eight objects, with 10 attempts per object (80 attempts in total).

	Training			**Inference**		
Model	**Dataset size**	**Training time** hr:min:sec	**Model size**	GTX 1080 Ti (FPS)	GeForce 940mx (FPS)	**Success rate** (%)
A	Faster R-CNN: 1500 CNN: 5000	00:14:00 00:02:00	Faster R-CNN: 300 MB CNN: 8 MB	20	2.5	91
B	1500	00:07:30	450 MB			83
C (simulation)	1500	00:07:00	450 MB			**94**
C (real experiment)	2000	00:08:00	450 MB			**89**
D	1500	00:06:30	330 MB			78

D. Limitations

During the experiments, random objects were placed in the view of the camera to observe the effect of unseen and similar objects to the grasp detection models. Even though this did not pose any major issues (i.e., no false positives), typical challenges in visual detection, such as illumination effects and object overlap, remain. While such effects can be taken into account in the training dataset, this would increase the size of the grasp model. Similarly, since only a single view of an object is used for demonstration, situations might occur where a current viewpoint of the camera does not capture the object well. One solution to this is to include multiple views of an object, each with their individual annotations, which increases the robustness of grasp detection. In our experiments, the results for model C where obtained with seven different views and annotations of the object.

The proposed approach only utilizes RGB images, meaning depth information is not taken into account, as compared to other work [21], [22]. Therefore, the grasping height must be known or estimated prior to a grasp action. This can be solved either by calibrating the camera with respect to the robot and its work area and assuming a fixed grasp height above the table, or by hand-guiding the robot to a desired grasp height. In this work, the former approach was taken.

A further limitation of our approach is the choice of object grasp annotations. For all models, an object grasp is only defined by a bounding box around the object (similar to [18]) and/or several keypoints. In some cases, this does not represent well the grasp pose of an object, for example when a grasp position is not in the center of the bounding box or keypoint set. In such case, the grasp position should be offset by the required distance from the center.

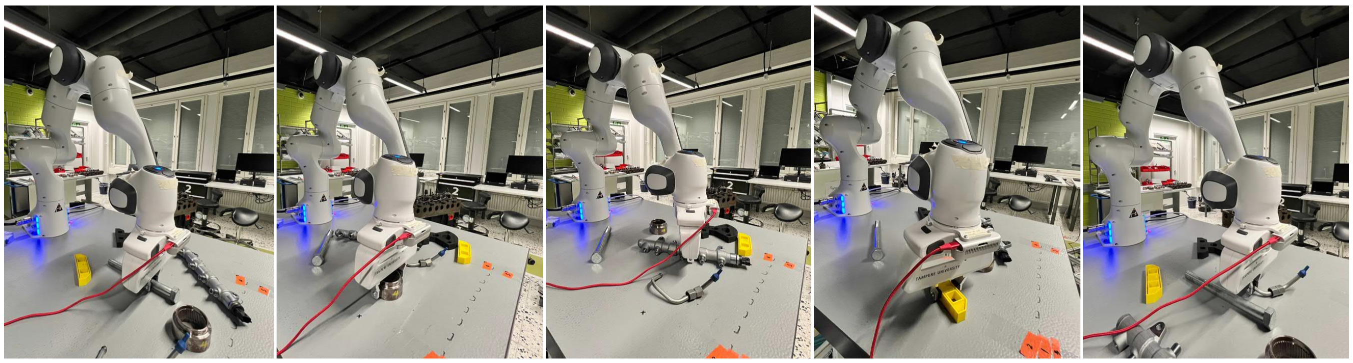
Finally, the planar grasp representation limits the approach to only top grasps with an end-effector pose perpendicular to the table (see Fig. 2). Other end-effector and/or grasp poses, would need to be modelled and integrated separately. One possible solution is to include depth sensing to extract the distance between object and gripper for 6D grasps.

VII. CONCLUSIONS

This work proposed a fast modelling approach for vision-based object grasp detection. Based on a single human object grasp annotation, an augmented dataset of RGB training images is generated, to be utilized for training a grasp detection model. Four different planar grasp detection models, each with different human annotations and grasp detection approach, are evaluated and implemented in simulation. All models are light-weight (below 0.5GB), enabling real-time inference. Best results were obtained with a keypoint-based model, which was further demonstrated with real robot grasping experiments. In all, from a human object grasp annotation, the augmented dataset and grasp model training, the approach enables the generation of a planar object grasp model in under 15 minutes.

ACKNOWLEDGEMENTS

Project funding was received from European Union's Horizon 2020 research and innovation programme, grant no. 871449 (OpenDR) and no. 825196 (TRINITY).



(a) (b) (c) (d) (e)

Fig. 6: Successful object grasping results with model C for different parts: (a) bolt, (b), gear casing, (c) Diesel engine common rail, (d) 3D printed part and (e) Diesel engine fuel line.

References

[1] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, pp. 248–266, 2018.

[2] S. El Zaatari, M. Marei, W. Li, and Z. Usman, "Cobot programming for collaborative industrial tasks: An overview," *Robotics and Autonomous Systems*, vol. 116, pp. 162–180, 2019.

[3] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping," *Current Robotics Reports*, pp. 1–11, 2020.

[4] 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, 2019.

[5] 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.

[6] 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.

[7] C. Eppner, A. Mousavian, and D. Fox, "ACRONYM: A large-scale grasp dataset based on simulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6222–6227.

[8] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[9] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE international conference on computer vision (ICCV)*, vol. 2, 1999, pp. 1150–1157.

[10] J. Chan, J. Addison Lee, and Q. Kemao, "BIND: Binary integrated net descriptors for texture-less object recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2068–2076.

[11] J. Ma, X. Jiang, A. Fan, J. Jiang, and J. Yan, "Image matching from handcrafted to deep features: A survey," *International Journal of Computer Vision*, vol. 129, no. 1, pp. 23–79, 2021.

[12] F. Tombari, S. Salti, and L. Di Stefano, "Performance evaluation of 3D keypoint detectors," *International Journal of Computer Vision*, vol. 102, no. 1-3, pp. 198–220, 2013.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *International Conference on Neural Information Processing Systems*, 2015, p. 91–99.

[14] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1802–1811.

[15] C. Capellen, M. Schwarz, and S. Behnke, "ConvPoseCNN: Dense convolutional 6D object pose estimation," *arXiv preprint arXiv:1912.07333*, 2019.

[16] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6D object pose estimation by iterative dense fusion," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3343–3352.

[17] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from RGBD images: Learning using a new rectangle representation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3304–3311.

[18] E. De Coninck, T. Verbelen, P. Van Molle, P. Simoens, and B. Dhoedt, "Learning robots to grasp by demonstration," *Robotics and Autonomous Systems*, vol. 127, p. 103474, 2020.

[19] A. Mousavian, C. Eppner, and D. Fox, "6-DOF graspnet: Variational grasp generation for object manipulation," in *IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 2901–2910.

[20] H. Zhang, X. Lan, X. Zhou, Z. Tian, Y. Zhang, and N. Zheng, "Visual manipulation relationship network for autonomous robotics," in *IEEE International Conference on Humanoid Robots (Humanoids)*, 2018, pp. 118–125.

[21] 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.

[22] B. Zhao, H. Zhang, X. Lan, H. Wang, Z. Tian, and N. Zheng, "REGNet: Region-based grasp network for single-shot grasp detection in point clouds," *arXiv preprint arXiv:2002.12647*, 2020.

[23] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 444–11 453.

[24] P. Wang, F. Manhardt, L. Minciullo, L. Garattoni, S. Meier, N. Navab, and B. Busam, "DemoGrasp: Few-shot learning for robotic grasping with human demonstration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5733–5740.

[25] 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*, pp. 1–58, 2020.

[26] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *IEEE European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[28] A. Ahmadyan, L. Zhang, A. Ablavatski, J. Wei, and M. Grundmann, "Objectron: A large scale dataset of object-centric videos in the wild with pose annotations," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 7822–7831.

[29] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3511–3516.

[30] N. Passalis *et al.*, "OpenDR: An open toolkit for enabling high performance, low footprint deep learning for robotics," *arXiv preprint arXiv:2203.00403*, 2022.